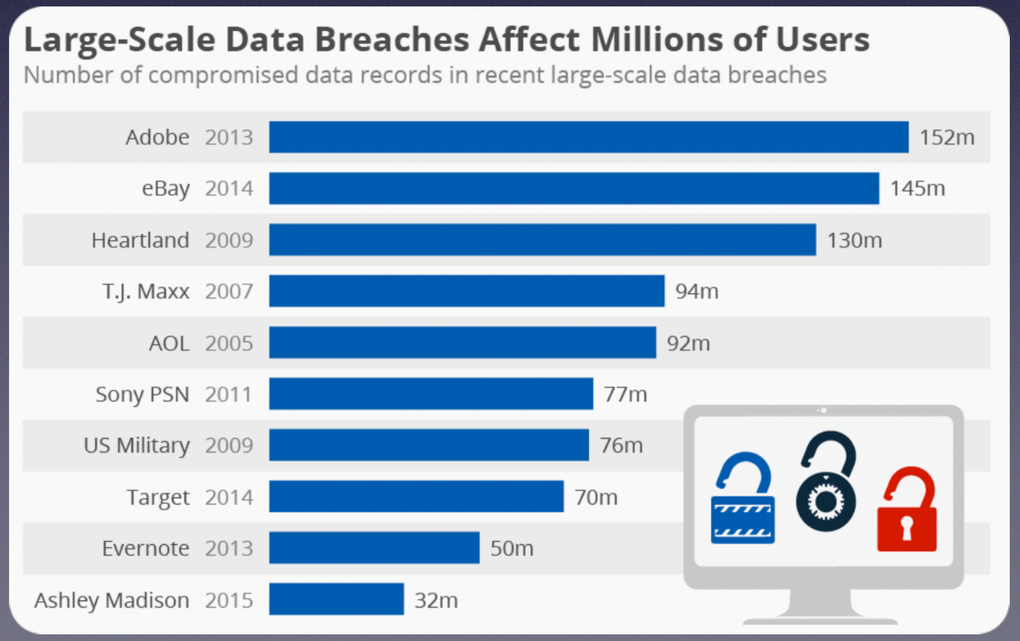
Data Security
&
Cybersecurity

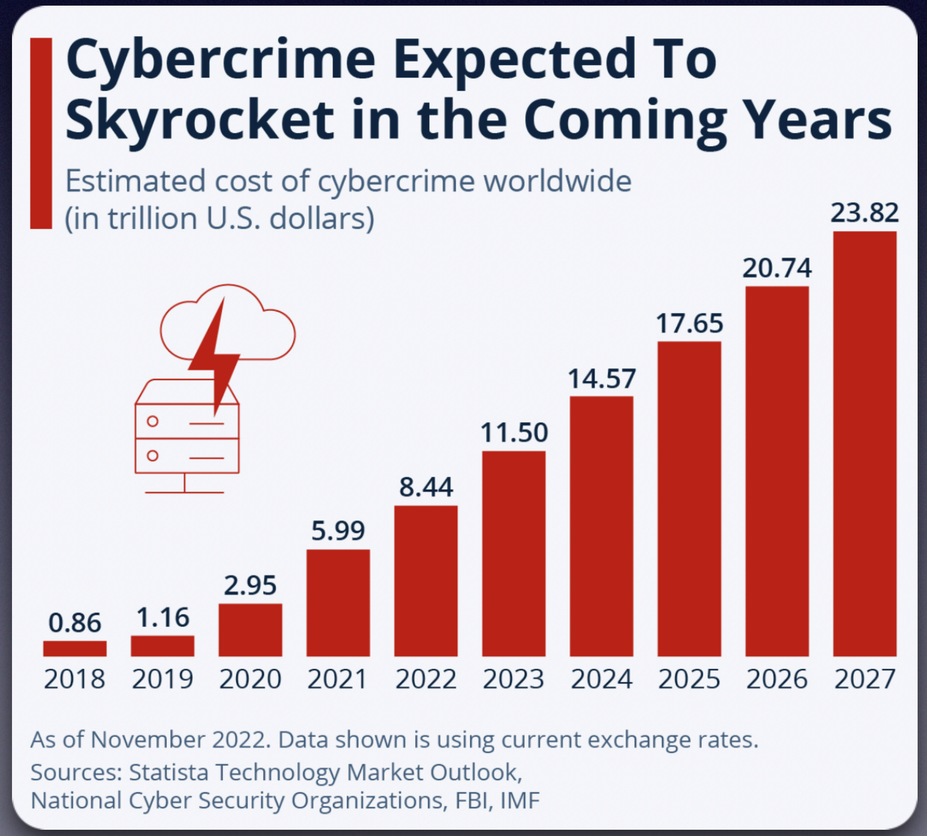
Blajan George

What is Cybersecurity ?

Cyber security is the application of technologies, processes, and controls to protect systems, networks, programs, devices and data from cyber attacks. It aims to reduce the risk of cyber attacks and protect against the unauthorised exploitation of systems, networks, and technologies.

Why is Cybersecurity important ?





Cybersecurity attacks

- Malware Attack
  - Adware
  - Spyware
  - Ransomware
- Phishing Attack
- Password Attack
- Man-in-the-Middle Attack
- SQL Injection Attack
- Spoofing
- Rainbow table attack
- Cross-Site Request Forgery (CSRF)
- Cross-Site Scripting (XSS) Attacks
- Distributed Denial of Service (DDos) Attacks

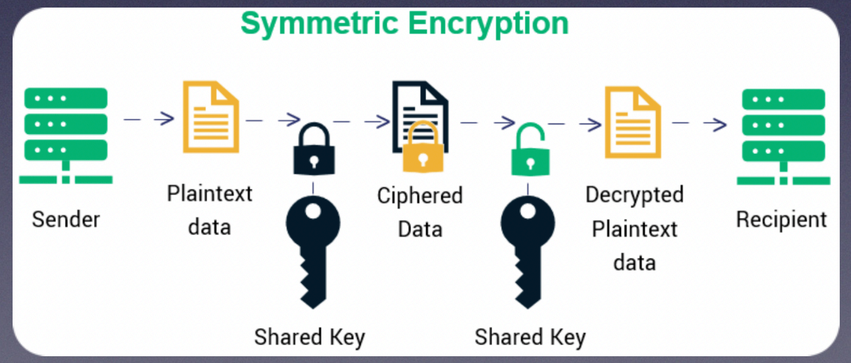
Solutions

Encryption & Hashing

Encryption is the method by which information is converted into secret code that hides the information's true meaning. The science of encrypting and decrypting information is called cryptography. In computing, unencrypted data is also known as plaintext, and encrypted data is called ciphertext.

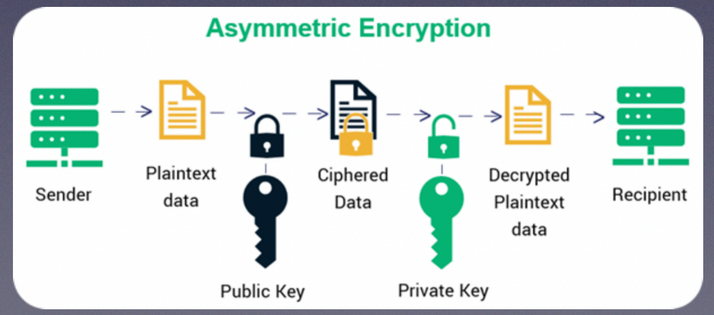
Symmetric encryption

In symmetric encryption, there is only one key, and all communicating parties use the same (secret) key for both encryption and decryption.



Asymmetric encryption

In asymmetric, or public key, encryption, there are two keys: one key is used for encryption, and a different key is used for decryption. The decryption key is kept private (hence the "private key" name), while the encryption key is shared publicly, for anyone to use (hence the "public key" name). Asymmetric encryption is a foundational technology for TLS (often called SSL).



Where ?

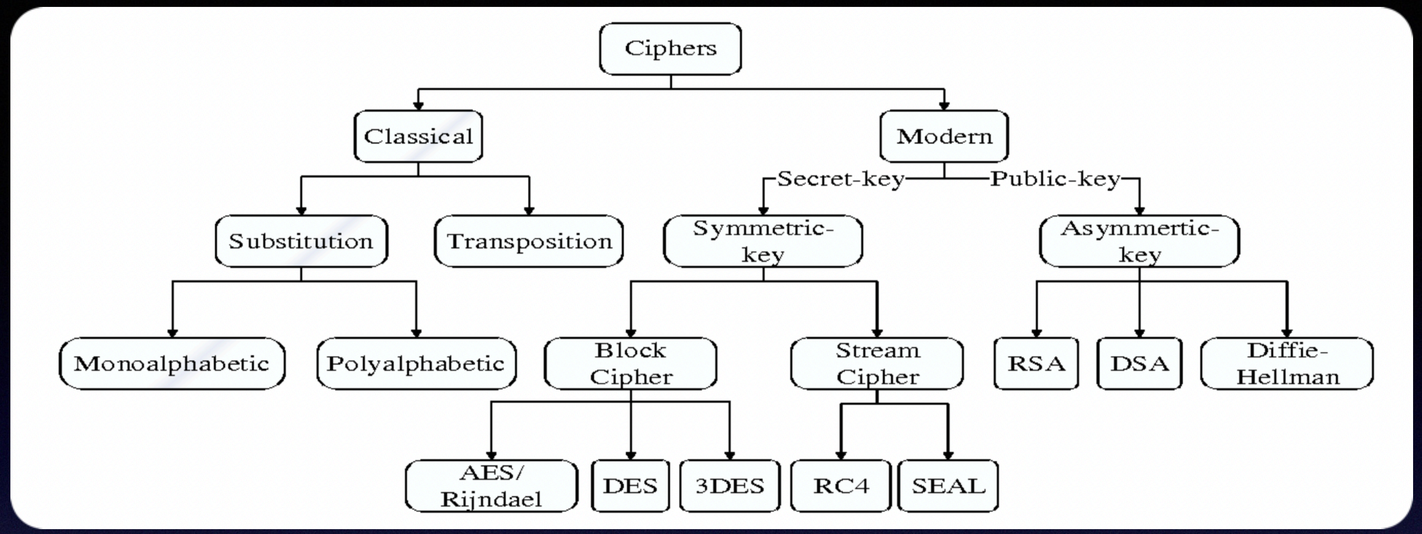
Encryption is foundational for a variety of technologies, but it is especially important for keeping HTTP requests and responses secure. The protocol responsible for this is called HTTPS (Hypertext Transfer Protocol Secure). A website served over HTTPS instead of HTTP will have a URL that begins with https:// instead of http://, usually represented by a secured lock in the address bar. HTTPS uses the encryption protocol called Transport Layer Security (TLS). In the past, an earlier encryption protocol called Secure Sockets Layer (SSL) was the standard, but TLS has replaced SSL. A website that implements HTTPS will have a TLS certificate installed on its origin server.

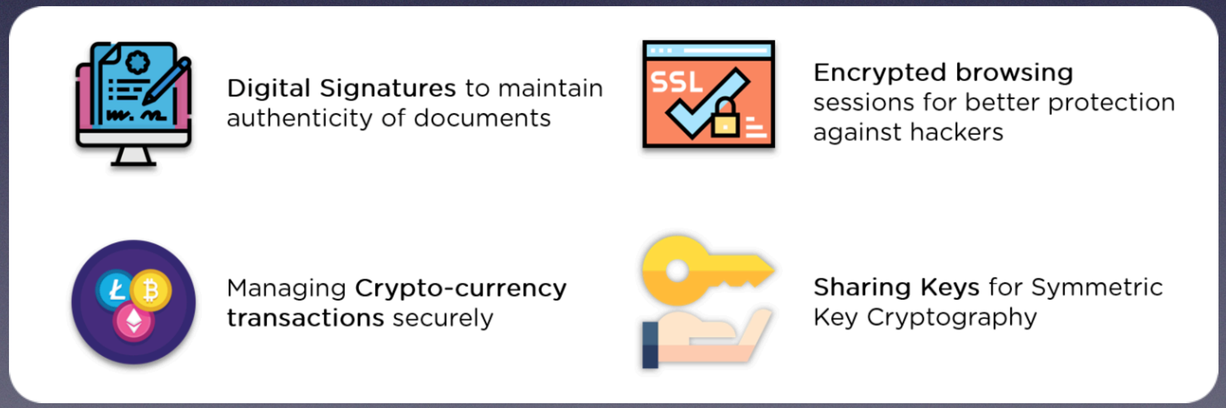
RSA Alg (Rivest-Shamir-Adleman)

1. **The 'client hello' message:** The client initiates the handshake by sending a "hello" message to the server. The message will include which TLS version the client supports, the cipher suites supported, and a string of random bytes known as the "client random."
2. **The 'server hello' message:** In reply to the client hello message, the server sends a message containing the server's SSL certificate, the server's chosen cipher suite, and the "server random," another random string of bytes that's generated by the server.
3. **Authentication:** The client verifies the server's SSL certificate with the certificate authority that issued it. This confirms that the server is who it says it is, and that the client is interacting with the actual owner of the domain.
4. **The premaster secret:** The client sends one more random string of bytes, the "premaster secret." The premaster secret is encrypted with the public key and can only be decrypted with the private key by the server. (The client gets the public key from the server's SSL certificate.)
5. **Private key used:** The server decrypts the premaster secret.
6. **Session keys created:** Both client and server generate session keys from the client random, the server random, and the premaster secret. They should arrive at the same results.
7. **Client is ready:** The client sends a "finished" message that is encrypted with a session key.
8. **Server is ready:** The server sends a "finished" message encrypted with a session key.
9. **Secure symmetric encryption achieved:** The handshake is completed, and communication continues using the session keys.

Diffie-Hellman Alg

1. **Client hello:** The client sends a client hello message with the protocol version, the client random, and a list of cipher suites.
2. **Server hello:** The server replies with its SSL certificate, its selected cipher suite, and the server random. In contrast to the RSA handshake described above, in this message the server also includes the following (step 3):
3. **Server's digital signature:** The server computes a digital signature of all the messages up to this point.
4. **Digital signature confirmed:** The client verifies the server's digital signature, confirming that the server is who it says it is.
5. **Client DH parameter:** The client sends its DH parameter to the server.
6. **Client and server calculate the premaster secret:** Instead of the client generating the premaster secret and sending it to the server, as in an RSA handshake, the client and server use the DH parameters they exchanged to calculate a matching premaster secret separately.
7. **Session keys created:** Now, the client and server calculate session keys from the premaster secret, client random, and server random, just like in an RSA handshake.
8. **Client is ready:** Same as an RSA handshake.
9. **Server is ready**
10. **Secure symmetric encryption achieved**



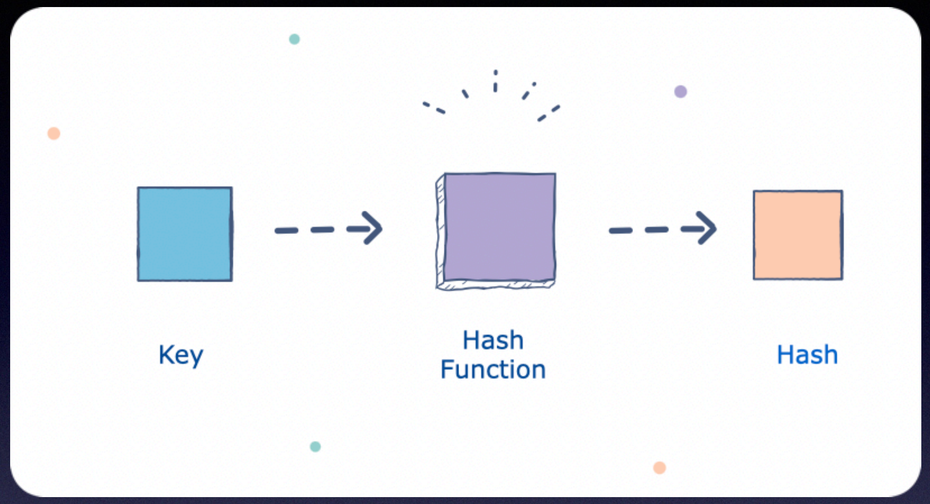


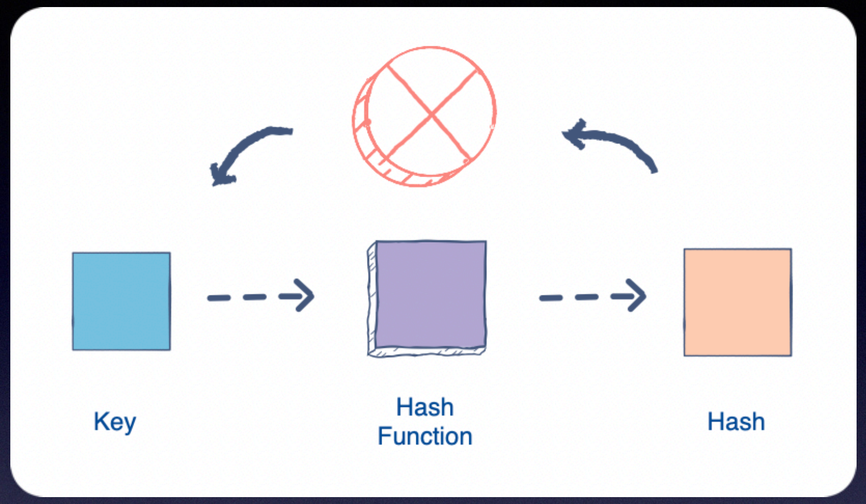
Key Differences	Symmetric Encryption	Asymmetric Encryption
Size of cipher text	Smaller cipher text compares to original plain text file.	Larger cipher text compares to original plain text file.
Data size	Used to transmit big data.	Used to transmit small data.
Resource Utilization	Symmetric key encryption works on low usage of resources.	Asymmetric encryption requires high consumption of resources.
Key Lengths	128 or 256-bit key size.	RSA 2048-bit or higher key size.
Security	Less secured due to use a single key for encryption.	Much safer as two keys are involved in encryption and decryption.
Number of keys	Symmetric Encryption uses a single key for encryption and decryption.	Asymmetric Encryption uses two keys for encryption and decryption
Techniques	It is an old technique.	It is a modern encryption technique.
Confidentiality	A single key for encryption and decryption has chances of key compromised.	Two keys separately made for encryption and decryption that removes the need to share a key.
Speed	Symmetric encryption is fast technique	Asymmetric encryption is slower in terms of speed.
Algorithms	RC4, AES, DES, 3DES, and QUAD.	RSA, Diffie-Hellman, ECC algorithms.

```java
/**
 * Method to encrypt private rsa keys stored in db
 *
 * @param data private rsa key
 * @return {@link Byte} encrypted rsa key
 */
2 usages  Blajan George *
public static byte[] encrypt(byte[] data, String encryptionSecret, byte[] iv)
        throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException,
        BadPaddingException, InvalidAlgorithmParameterException {
    Key key = new SecretKeySpec(encryptionSecret.getBytes(), AES);
    Cipher c = Cipher.getInstance(AES_GCM);
    GCMParameterSpec gcmParameterSpec = new GCMParameterSpec( tLen: 128, iv);
    c.init(Cipher.ENCRYPT_MODE, key, gcmParameterSpec);

    return c.doFinal(data);
}
```

```java
/**
 * Method to decrypt private rsa keys stored in db
 *
 * @param encryptedData private rsa key encrypted
 * @return {@link Byte} decrypted rsa key
 */
1 usage  Blajan George *
public static byte[] decrypt(byte[] encryptedData, String encryptionSecret, byte[] iv)
        throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException,
        BadPaddingException, InvalidAlgorithmParameterException {
    Key key = new SecretKeySpec(encryptionSecret.getBytes(), AES);
    Cipher c = Cipher.getInstance(AES_GCM);
    GCMParameterSpec gcmParameterSpec = new GCMParameterSpec( tLen: 128, iv);
    c.init(Cipher.DECRYPT_MODE, key, gcmParameterSpec);

    return c.doFinal(encryptedData);
}
```

Hashing is the process of converting a given key into another value. A hash function is used to generate the new value according to a mathematical algorithm. The result of a hash function is known as a hash value or simply, a hash.

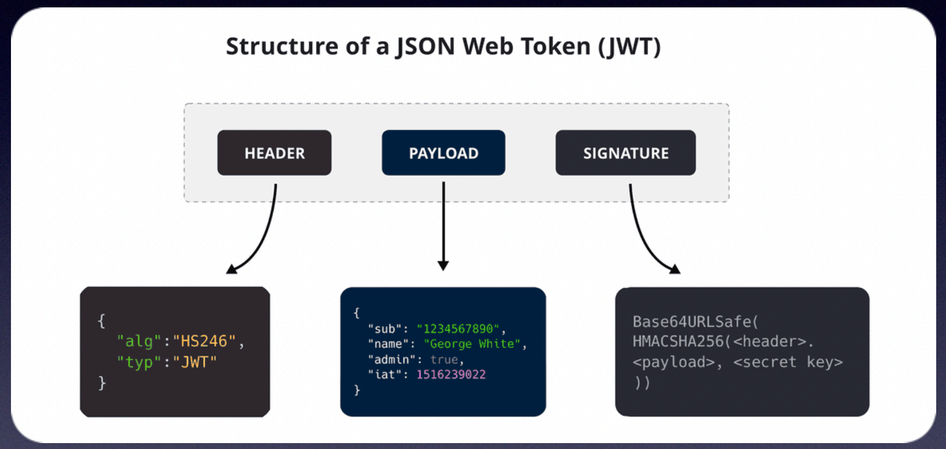




Salting is a concept that typically pertains to password hashing. Essentially, it's a unique value that can be added to the end of the password to create a different hash value. This adds a layer of security to the hashing process, specifically against brute force attacks.

JWT

JSON Web Token is an open industry standard used to share information between two entities, usually a client (like your app's frontend) and a server (your app's backend). They contain JSON objects which have the information that needs to be shared. Each JWT is also signed using cryptography (hashing) to ensure that the JSON contents (also known as JWT claims) cannot be altered by the client or a malicious party.



Registered Claims

- iss (issuer): Issuer of the JWT
- sub (subject): Subject of the JWT (the user)
- aud (audience): Recipient for which the JWT is intended
- exp (expiration time): Time after which the JWT expires
- nbf (not before time): Time before which the JWT must not be accepted for processing
- iat (issued at time): Time at which the JWT was issued; can be used to determine age of the JWT

An HttpOnly Cookie is a tag added to a browser cookie that prevents client-side scripts from accessing data. It provides a gate that prevents the specialized cookie from being accessed by anything other than the server.

The SameSite attribute of the Set-Cookie HTTP response header allows you to declare if your cookie should be restricted to a first-party or same-site context.

But what if we want to implement the Oauth2 policy ? Oauth2 forces us to send a Bearer token in the Authorization header.

Rules of thumb for XSS

- Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

- Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

- Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

Rules of thumb for CSRF

- Double Submit Cookie

This technique is easy to implement and is stateless. In this technique, we send a random value in both a cookie and as a request parameter, with the server verifying if the cookie value and request value match. When a user visits (even before authenticating to prevent login CSRF), the site should generate a (cryptographically strong) pseudorandom value and set it as a cookie on the user's machine separate from the session identifier. The site then requires that every transaction request includes this pseudorandom value as a hidden form value (or as a request parameter/header). If both of them match at server side, the server accepts it as legitimate request and if they don't, it would reject the request.

```
1 usage   Blajan George
private String doGenerateToken(Map<String, Object> claims, String subject) {

    return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + timeToLive))
            .signWith(SignatureAlgorithm.HS512, secret.getBytes()).compact();
}
```

```
public boolean validateSignature(String token) {
    String[] chunks = token.split( regex: "\\.");

    String jwtWithoutSignature = chunks[0] + "." + chunks[1];
    String base64UrlEncodedSignature = chunks[2];

    SecretKeySpec secretKeySpec = new SecretKeySpec(secret.getBytes(), SignatureAlgorithm.HS512.getJcaName());
    DefaultJwtSignatureValidator validator = new DefaultJwtSignatureValidator(SignatureAlgorithm.HS512, secretKeySpec);

    return validator.isValid(jwtWithoutSignature, base64UrlEncodedSignature);
}
```

Rate limiter

Rate limiting is a strategy for limiting network traffic. It puts a cap on how often someone can repeat an action within a certain timeframe — for instance, trying to log in to an account. Rate limiting can help stop certain kinds of malicious bot activity. It can also reduce strain on web servers.

CDN (Content Delivery Network)

A content delivery network (CDN) is a geographically distributed group of servers that caches content close to end users. A CDN allows for the quick transfer of assets needed for loading Internet content, including HTML pages, JavaScript files, stylesheets, images, and videos. The popularity of CDN services continues to grow, and today the majority of web traffic is served through CDNs, including traffic from major sites like Facebook, Netflix, and Amazon.

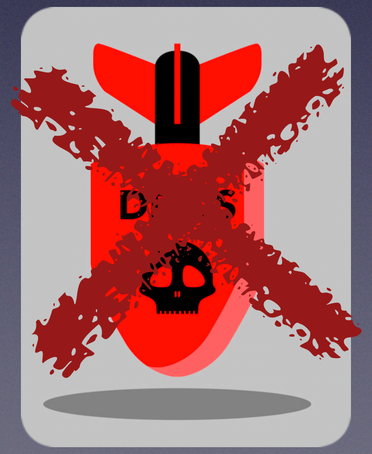
What are the benefits of using a CDN?

1. **Improving website load times** - By distributing content closer to website visitors by using a nearby CDN server (among other optimizations), visitors experience faster page loading times. As visitors are more inclined to click away from a slow-loading site, a CDN can reduce bounce rates and increase the amount of time that people spend on the site. In other words, a faster a website means more visitors will stay and stick around longer.

2. **Reducing bandwidth costs** - Bandwidth consumption costs for website hosting is a primary expense for websites. Through caching and other optimizations, CDNs are able to reduce the amount of data an origin server must provide, thus reducing hosting costs for website owners.

3. **Increasing content availability and redundancy** - Large amounts of traffic or hardware failures can interrupt normal website function. Thanks to their distributed nature, a CDN can handle more traffic and withstand hardware failure better than many origin servers.

4. **Improving website security** - A CDN may improve security by providing DDoS mitigation, improvements to security certificates, and other optimizations.



SQL Injection

```
...
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = '"
                + userName + "' AND itemname = '"
                + ItemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

```
SELECT * FROM items
WHERE owner =
AND itemname = ;
```

itemName = “name’ OR ‘a’=‘a”

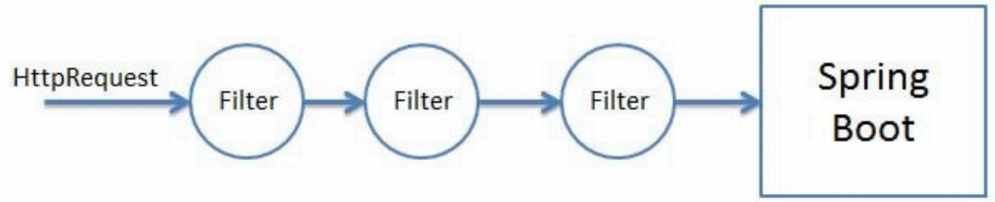
```
SELECT * FROM items
WHERE owner = 'wiley'
AND itemname = 'name' OR 'a'='a';
```

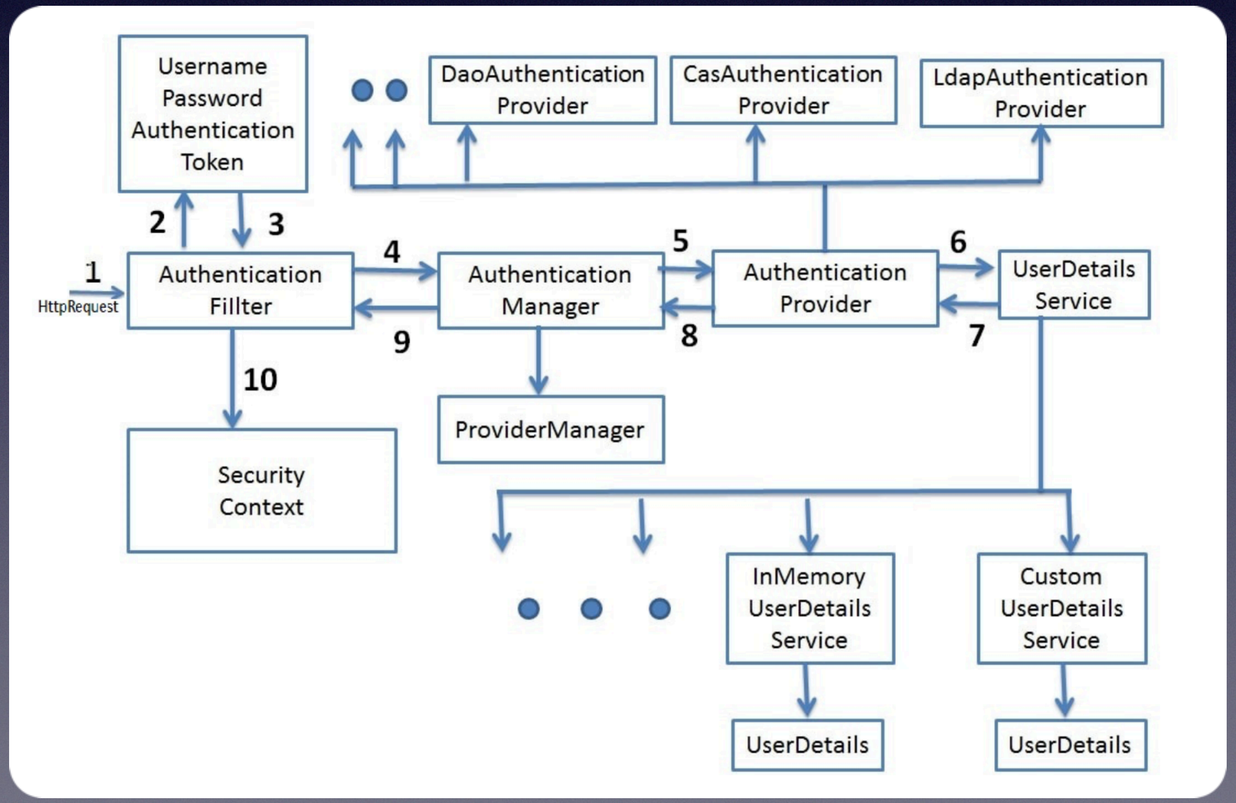
Rules of thumb for SQLI

- Use of Prepared Statements (with Parameterized Queries)
- Allow-list Input Validation
- Escaping All User Supplied Input

Spring Security Architecture

- **Filters -** Before the request reaches the Dispatcher Servlet, it is first intercepted by a chain of filters.





Q&A