# API Keys

An Application Programming Interface (API) key is an authentication token that allows you to call an API. An application passes an API key to the API, which is then called to identify the person, programmer, or program trying to access a website. It is frequently accompanied by a set of access rights that are specific to the API the key is linked to. In this reading, you will delve into API keys, their role, their function in authentication and authorization, and how they are used.

The API key is usually randomly generated by the application and must be sent on every API call. It serves as a distinctive identifier and offers a secure token for authentication.

## Authentication and authorization

API keys may be used for both authentication, making sure you're who you say you are, and authorization, deciding which APIs you are allowed to call.

When you are authenticating with API keys, you are ensuring that malicious users or applications can't call an API and make unauthorized or authorized changes. With project authentication (application or site authentication), API keys help identify the project or application that makes the call. If you are using API keys for user authentication, the identity of the user is being verified.

When you are authorizing with API keys, you are also ensuring that you have the correct API call. Authorization will also check that the API key being used in the project is available.

## How they are used

When using APIs, the usage depends on the specific API. With most APIs, you are required to send the API key with every request. It can be sent in one of several ways:

1. As an HTTP parameter in the request URL. Example:
   `GET https://myapp.com/api/users/list?apikey=12345678`
2. As an HTTP header sent with the request. Example:
   `GET https://myapp.com/api/users/listX-API-Key: 12345678`
3. (Rarely) Posted to a specific authorization endpoint, which returns another token or a cookie to be sent with subsequent requests. Example:
   `POST https://myapp.com/api/auth{ "token": "12345678" }`

One last tip, do not hardcode API keys into your application code, especially if it will be posted in a public repository like Github. If you have hardcoded your API keys into your application code, anyone who wants to can make API calls with your authorization!

Unfortunately, it happens every day. For this reason, many applications are moving away from API keys and toward OAuth, which requires the user to manually authorize an application before using it. With being extra cautious, you can make sure this does not happen to you.

## Key takeaways

- **API keys facilitate secure interactions:** The API key serves as a crucial authentication token that not only permits API calls, but also plays a vital role in regulating access privileges and defining permissible actions. It's an essential tool in ensuring secure and controlled communication within digital ecosystems.
- **Authentication and authorization:** API keys serve a dual purpose: authentication and authorization. Authentication verifies the identity of users or applications making API calls, preventing unauthorized access or changes. Authorization, on the other hand, ensures that users have the appropriate rights to call specific APIs, promoting controlled usage and adherence to access policies.
- **Effective API key usage:** When using APIs, the API key can be included as an HTTP parameter in the URL or an HTTP header. Ensuring that API keys aren't hardcoded in application code is important in order to prevent unauthorized access. Many applications are transitioning from API keys to more secure methods where manual user authorization enhances security measures and minimizes risks associated with API misuse.

Go to next item 