Encrypting secrets for the REST API

In this article

In order to create or update a secret with the REST API, you must encrypt the value of the secret.

About encrypting secrets

Several REST API endpoints let you create secrets on GitHub. To use these endpoints, you must encrypt the secret value using libsodium. For more information, see the libsodium documentation.

In order to encrypt a secret, you need a Base64 encoded public key. You can get a public key from the REST API. To determine which endpoint to use to get the public key, look at the documentation for the `encrypted_value` parameter in the endpoint that you will use to create a secret .

Example encrypting a secret using Node.js

If you are using Node.js, you can encrypt your secret using the libsodium-wrappers library. For more information, see libsodium-wrappers.

In the following example, replace `YOUR_SECRET` with the plain text value that you want to encrypt. Replace `YOUR_BASE64_KEY` with your Base64 encoded public key. The documentation for the endpoint that you will use to create a secret will tell you which endpoint you can use to get the public key. `ORIGINAL` is not a placeholder; it is a parameter for the libsodium-wrappers library.

JavaScript

```
const sodium = require('libsodium-wrappers')

const secret = 'YOUR_SECRET'
const key = 'YOUR_BASE64_KEY'

//Check if libsodium is ready and then proceed.
sodium.ready.then(() => {
  // Convert the secret and key to a Uint8Array.
  let binkey = sodium.from_base64(key, sodium.base64_variants.ORIGINAL)
  let binsec = sodium.from_string(secret)

  // Encrypt the secret using libsodium
  let encBytes = sodium.crypto_box_seal(binsec, binkey)
```

```
  // Convert the encrypted Uint8Array to Base64
  let output = sodium.to_base64(encBytes, sodium.base64_variants.ORIGINAL)

  // Print the output
  console.log(output)
});
```

Example encrypting a secret using Python

If you are using Python 3, you can encrypt your secret using the PyNaCl library. For more information, see PyNaCl.

In the following example, replace `YOUR_SECRET` with the plain text value that you want to encrypt. Replace `YOUR_BASE64_KEY` with your Base64 encoded public key. The documentation for the endpoint that you will use to create a secret will tell you which endpoint you can use to get the public key.

Python

```
from base64 import b64encode
from nacl import encoding, public

def encrypt(public_key: str, secret_value: str) -> str:
  """Encrypt a Unicode string using the public key."""
  public_key = public.PublicKey(public_key.encode("utf-8"),
encoding.Base64Encoder())
  sealed_box = public.SealedBox(public_key)
  encrypted = sealed_box.encrypt(secret_value.encode("utf-8"))
  return b64encode(encrypted).decode("utf-8")

encrypt("YOUR_BASE64_KEY", "YOUR_SECRET")
```

Example encrypting a secret using C#

If you are using C#, you can encrypt your secret using the Sodium.Core package. For more information, see Sodium.Core.

In the following example, replace `YOUR_SECRET` with the plain text value that you want to encrypt. Replace `YOUR_BASE64_KEY` with your Base64 encoded public key. The documentation for the endpoint that you will use to create a secret will tell you which endpoint you can use to get the public key.

C#

```
var secretValue = System.Text.Encoding.UTF8.GetBytes("YOUR_SECRET");
var publicKey = Convert.FromBase64String("YOUR_BASE64_KEY");

var sealedPublicKeyBox = Sodium.SealedPublicKeyBox.Create(secretValue,
publicKey);

Console.WriteLine(Convert.ToBase64String(sealedPublicKeyBox));
```

Example encrypting a secret using Ruby

If you are using Ruby, you can encrypt your secret using the RbNaCl gem. For more information, see RbNaCl.

In the following example, replace `YOUR_SECRET` with the plain text value that you want to

encrypt. Replace `YOUR_BASE64_KEY` with your Base64 encoded public key. The documentation for the endpoint that you will use to create a secret will tell you which endpoint you can use to get the public key.

Ruby

```ruby
require "rbnacl"
require "base64"

key = Base64.decode64("YOUR_BASE64_KEY")
public_key = RbNaCl::PublicKey.new(key)

box = RbNaCl::Boxes::Sealed.from_public_key(public_key)
encrypted_secret = box.encrypt("YOUR_SECRET")

# Print the base64 encoded secret
puts Base64.strict_encode64(encrypted_secret)
```