In Django Rest Framework (DRF), you can implement API key and secret authentication using a custom authentication class. Here's a sample code snippet to get you started:

1. Install Django Rest Framework if you haven't already:

```
pip install djangorestframework
```

2. Create a custom authentication class: In your Django project, create a new file named `custom_auth.py` inside one of your app's folders (or create a new app if you don't have one yet).

```
from rest_framework import authentication
from rest_framework import exceptions
from your_app.models import ApiKey  # Replace with your model for storing
API keys/secrets

class ApiKeyAuthentication(authentication.BaseAuthentication):
    def authenticate(self, request):
        api_key = request.META.get('HTTP_X_API_KEY')
        api_secret = request.META.get('HTTP_X_API_SECRET')

        if not api_key or not api_secret:
            return None

        try:
            api_key_obj = ApiKey.objects.get(key=api_key, secret=api_secret)
        except ApiKey.DoesNotExist:
            raise exceptions.AuthenticationFailed('Invalid API key or
secret')

        return (api_key_obj.user, None)
```

3. Update your Django settings: In your `settings.py` file, add the custom authentication class to the `REST_FRAMEWORK` settings:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'your_app.custom_auth.ApiKeyAuthentication',
    ],
}
```

4. Define your API key model: Create a model in your app's `models.py` to store API keys and secrets. You can customize this according to your needs.

```
from django.db import models

class ApiKey(models.Model):
    user = models.ForeignKey('auth.User', on_delete=models.CASCADE)
    key = models.CharField(max_length=64, unique=True)
    secret = models.CharField(max_length=128)

    def __str__(self):
        return self.key
```

5. Migrate your database: Run the following command to apply the changes to your database:

```
python manage.py makemigrations
python manage.py migrate
```

Now your DRF API is configured to use API key and secret authentication. When making requests, clients need to include the `X-API-KEY` and `X-API-SECRET` headers in the request with the corresponding values.

Please note that this is a basic example. Depending on your security and usage requirements, you might want to enhance this implementation further, possibly by adding rate limiting, expiration, or other security measures to your API keys and secrets.