

---

# **django-e2ee-framework**

**Philipp S. Sommer**

**Oct 22, 2022**



**CONTENTS:**

- 1 Installation 3**
  - 1.1 Installation from PyPi . . . . . 3
  - 1.2 Install the Django App for your project . . . . . 3
  - 1.3 Installation for development . . . . . 4
- 2 Configuration options 5**
  - 2.1 Configuration settings . . . . . 5
- 3 API Reference 7**
  - 3.1 App settings . . . . . 7
  - 3.2 URL config . . . . . 7
  - 3.3 Models . . . . . 7
  - 3.4 Views . . . . . 17
  - 3.5 django\_e2ee package . . . . . 25
- 4 Contribution and development hints 47**
  - 4.1 Contributing in the development . . . . . 47
- 5 Indices and tables 49**
- Python Module Index 51**
- Index 53**



## An end-to-end encryption framework for Django

**Warning:** This package is work in progress, especially it's documentation. Stay tuned for updates and discuss with us at <https://gitlab.hzdr.de/hcdc/django/django-e2ee-framework>



## INSTALLATION

To install the *django-e2ee-framework* package for your Django project, you need to follow two steps:

1. *Install the package*
2. *Add the app to your Django project*

### 1.1 Installation from PyPi

The recommended way to install this package is via pip and PyPi via:

```
pip install django-e2ee-framework
```

Or install it directly from [the source code repository on Gitlab](https://gitlab.hzdr.de/hcdc/django/django-e2ee-framework) via:

```
pip install git+https://gitlab.hzdr.de/hcdc/django/django-e2ee-framework.git
```

The latter should however only be done if you want to access the development versions.

### 1.2 Install the Django App for your project

To use the *django-e2ee-framework* package in your Django project, you need to add the app to your *INSTALLED\_APPS*, configure your *urls.py*, run the migration, add a login button in your templates. Here are the step-by-step instructions:

1. Add the *django\_e2ee* app to your *INSTALLED\_APPS*
2. in your projects *urlconf* (see **setting: ROOT\_URLCONF**), add include *django\_e2ee.urls* via:

```
from django.urls import include, path

urlpatterns += [
    path("django-e2ee-framework/", include("django_e2ee.urls")),
]
```

3. Run `python manage.py migrate` to add models to your database
4. Configure the app to your needs (see *Configuration options*).

That's it! For further adaption to you Django project, please head over to the *Configuration options*. You can also have a look into the *testproject* in the [source code repository](https://gitlab.hzdr.de/hcdc/django/django-e2ee-framework) for a possible implementation.

## 1.3 Installation for development

Please head over to our [contributing guide](#) for installation instruction for development.



## CONFIGURATION OPTIONS

### 2.1 Configuration settings

The following settings have an effect on the app.



## API REFERENCE

### 3.1 App settings

This module defines the settings options for the `django-e2ee-framework` app.

### 3.2 URL config

URL patterns of the `django-e2ee-framework` to be included via:

```
from django.urls import include, path

urlpatterns = [
    path(
        "django-e2ee-framework",
        include("django_e2ee.urls"),
    ),
]
```

#### Data:

<code>app_name</code>	App name for the <code>django-e2ee-framework</code> to be used in calls to <code>django.urls.reverse()</code>
-----------------------	---

`django_e2ee.urls.app_name = 'e2ee'`

App name for the `django-e2ee-framework` to be used in calls to `django.urls.reverse()`

### 3.3 Models

Models for the `django-e2ee-framework` app.

#### Models:

<i>EncryptionKey</i> (*args, **kwargs)	A key that is used for encrypting content in the database.
<i>EncryptionKeySecret</i> (*args, **kwargs)	The secret of an encryption key, encrypted with a users master key.
<i>MasterKey</i> (*args, **kwargs)	A public key for a user that is used for encryption.
<i>MasterKeySecret</i> (*args, **kwargs)	The encrypted private key for a <i>MasterKey</i>
<i>SessionKey</i> (*args, **kwargs)	An encrypted private key for a <i>MasterKey</i> for one session.

#### Functions:

<i>delete_session_key</i> (request, **kwargs)	Deactivate the notification subscriptions for the session.
<i>validate_public_key</i> (value)	Try importing a public key with cryptography

**class** django\_e2ee.models.**EncryptionKey**(\*args, \*\*kwargs)

Bases: Model

A key that is used for encrypting content in the database.

#### Miscellaneous:

<i>DoesNotExist</i>	
<i>MultipleObjectsReturned</i>	

#### Model Fields:

<i>created_by</i>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<i>uuid</i>	A wrapper for a deferred-loading field.

#### Attributes:

<i>created_by_id</i>	
<i>encryptionkeysecret_set</i>	Accessor to the related objects manager on the reverse side of a many-to-one relation.
<i>master_keys</i>	Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.
<i>objects</i>	

**exception** DoesNotExist

Bases: ObjectDoesNotExist

**exception** MultipleObjectsReturned

Bases: MultipleObjectsReturned

**created\_by**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

**created\_by\_id****encryptionkeysecret\_set**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create\_forward\_many\_to\_many\_manager() defined below.

**master\_keys**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

Pizza.toppings and Topping.pizzas are ManyToManyDescriptor instances.

Most of the implementation is delegated to a dynamically defined manager class built by create\_forward\_many\_to\_many\_manager() defined below.

**objects = <django.db.models.manager.Manager object>**

**uuid**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**class django\_e2ee.models.EncryptionKeySecret(\*args, \*\*kwargs)**

Bases: Model

The secret of an encryption key, encrypted with a users master key.

**Miscellaneous:**


---

*DoesNotExist*

---

*MultipleObjectsReturned*

---

**Model Fields:**

<code>encrypted_with</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<code>encryption_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<code>id</code>	A wrapper for a deferred-loading field.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>signature</code>	A wrapper for a deferred-loading field.
<code>signed_by</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

#### Attributes:

<code>encrypted_with_id</code>
<code>encryption_key_id</code>
<code>objects</code>
<code>signed_by_id</code>

#### exception DoesNotExist

Bases: `ObjectDoesNotExist`

#### exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

#### encrypted\_with

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

#### encrypted\_with\_id

#### encryption\_key

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

**encryption\_key\_id****id**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**objects** = <django.db.models.manager.Manager object>

**secret**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**signature**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**signed\_by**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

**signed\_by\_id**

**class** django\_e2ee.models.MasterKey(\*args, \*\*kwargs)

Bases: Model

A public key for a user that is used for encryption.

**Miscellaneous:**

---

*DoesNotExist*

---

*MultipleObjectsReturned*

---

**Attributes:**

<code>encryptionkey_set</code>	Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.
<code>encryptionkeysecret_set</code>	Accessor to the related objects manager on the reverse side of a many-to-one relation.
<code>masterkeysecret_set</code>	Accessor to the related objects manager on the reverse side of a many-to-one relation.
<code>objects</code>	
<code>pubkey_loaded</code>	The pubkey loaded via cryptography
<code>sessionkey_set</code>	Accessor to the related objects manager on the reverse side of a many-to-one relation.
<code>signed_secrets</code>	Accessor to the related objects manager on the reverse side of a many-to-one relation.
<code>signing_pubkey_loaded</code>	The pubkey loaded via cryptography
<code>user_id</code>	

#### Model Fields:

<code>pubkey</code>	A wrapper for a deferred-loading field.
<code>signing_pubkey</code>	A wrapper for a deferred-loading field.
<code>user</code>	Accessor to the related object on the forward side of a one-to-one relation.

#### exception DoesNotExist

Bases: `ObjectDoesNotExist`

#### exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

#### encryptionkey\_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

#### encryptionkeysecret\_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.



**masterkeysecret\_set**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create\_forward\_many\_to\_many\_manager() defined below.

**objects** = <django.db.models.manager.Manager object>

**pubkey**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**property pubkey\_loaded**

The pubkey loaded via cryptography

**sessionkey\_set**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create\_forward\_many\_to\_many\_manager() defined below.

**signed\_secrets**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create\_forward\_many\_to\_many\_manager() defined below.

**signing\_pubkey**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**property signing\_pubkey\_loaded**

The pubkey loaded via cryptography

**user**

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

#### **user\_id**

**class** `django_e2ee.models.MasterKeySecret(*args, **kwargs)`

Bases: `Model`

The encrypted private key for a `MasterKey`

#### **Miscellaneous:**

---

`DoesNotExist`

---

`MultipleObjectsReturned`

---

#### **Model Fields:**

<code>identifier</code>	A wrapper for a deferred-loading field.
<code>iv</code>	A wrapper for a deferred-loading field.
<code>master_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via <code>ForwardOneToOneDescriptor</code> subclass) relation.
<code>salt</code>	A wrapper for a deferred-loading field.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>signing_secret</code>	A wrapper for a deferred-loading field.
<code>uuid</code>	A wrapper for a deferred-loading field.

#### **Attributes:**

---

`master_key_id`

---

`objects`

---

#### **exception DoesNotExist**

Bases: `ObjectDoesNotExist`

#### **exception MultipleObjectsReturned**

Bases: `MultipleObjectsReturned`

#### **identifier**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

#### **iv**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**master\_key**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

**master\_key\_id**

**objects** = <django.db.models.manager.Manager object>

**salt**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**secret**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**signing\_secret**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**uuid**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**class** django\_e2ee.models.**SessionKey**(\*args, \*\*kwargs)

Bases: Model

An encrypted private key for a *MasterKey* for one session.

**Miscellaneous:**


---

*DoesNotExist*

---

*MultipleObjectsReturned*

---

**Methods:**


---

*get\_absolute\_url()*

---

**Model Fields:**

<code>ignore</code>	A wrapper for a deferred-loading field.
<code>iv</code>	A wrapper for a deferred-loading field.
<code>master_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>session</code>	Accessor to the related object on the forward side of a one-to-one relation.
<code>session_secret</code>	A wrapper for a deferred-loading field.
<code>signing_secret</code>	A wrapper for a deferred-loading field.

#### Attributes:

<code>master_key_id</code>
<code>objects</code>
<code>session_id</code>

#### exception DoesNotExist

Bases: `ObjectDoesNotExist`

#### exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

#### get\_absolute\_url()

#### ignore

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

#### iv

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

#### master\_key

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

#### master\_key\_id

`objects = <django.db.models.manager.Manager object>`

#### secret

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**session**

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

**session\_id****session\_secret**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**signing\_secret**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`django_e2ee.models.delete_session_key(request, **kwargs)`

Deactivate the notification subscriptions for the session.

`django_e2ee.models.validate_public_key(value: str)`

Try importing a public key with cryptography

## 3.4 Views

Views of the `django-e2ee-framework` app to be imported via the url config (see [django\\_e2ee.urls](#)).

**Classes:**

<code>E2ELoginViewMixin()</code>	A mixin that gets a form to login or create an E2EE password.
<code>EncryptionKeySecretView(**kwargs)</code>	A view to create and get <code>e2ee.EncryptionKeySecret</code> .
<code>EncryptionKeyViewSet(**kwargs)</code>	A viewset for <code>e2ee.EncryptionKey</code>
<code>MasterKeyRetrieveView(**kwargs)</code>	Retrieve the master key of the user.
<code>MasterKeySecretViewSet(**kwargs)</code>	A viewset for a users <code>e2ee.MasterKeySecret</code>
<code>MasterKeyViewSet(**kwargs)</code>	A view to create and retrieve a users <code>e2ee.MasterKey</code>
<code>SessionKeyUpdateView(**kwargs)</code>	Retrieve the master key of the user.
<code>SessionKeyViewSet(**kwargs)</code>	A viewset for a users <code>e2ee.SessionKey</code>

**Functions:**

`dummy_view(request, *args, **kwargs)`

**class** `django_e2ee.views.E2ELoginViewMixin`

Bases: `object`

A mixin that gets a form to login or create an E2EE password.

This mixin provides an additional `e2ee_login_form` context variable for the template. Depending on the fact whether the user already created an E2EE password, this can be a [`django\_e2ee.forms.PasswordCreateForm`](#) to create a new E2EE password, or a [`django\_e2ee.forms.PasswordInputForm`](#) to enter the password.

Additionally, it adds a context variable `e2ee_login_url` that points to the URL where the form should be submitted.

**Methods:**

---

[`get\_context\_data\(\*\*kwargs\)`](#)

---

[`get\_context\_data\(\*\*kwargs\)`](#)**class** `django_e2ee.views.EncryptionKeySecretView(**kwargs)`

Bases: `RetrieveAPIView`, `CreateAPIView`

A view to create and get `e2ee.EncryptionKeySecret`.

**Methods:**

<a href="#"><code>get_object()</code></a>	Returns the object the view is displaying.
<a href="#"><code>get_queryset()</code></a>	Get the list of items for this view.
<a href="#"><code>get_serializer(*args, **kwargs)</code></a>	Return the serializer instance that should be used for validating and deserializing input, and for serializing output.

**Attributes:**

---

[`permission\_classes`](#)

---

[`queryset`](#)

---

**Classes:**

<a href="#"><code>serializer_class</code></a>	alias of <a href="#"><code>EncryptionKeySecretSerializer</code></a>
---	---

**get\_object()**

Returns the object the view is displaying.

You may want to override this if you need to provide non-standard queryset lookups. Eg if objects are referenced using multiple keyword arguments in the url conf.

**get\_queryset()**

Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using `self.queryset`.

This method should always be used rather than accessing `self.queryset` directly, as `self.queryset` gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

**get\_serializer(\*args, \*\*kwargs)**

Return the serializer instance that should be used for validating and deserializing input, and for serializing output.

**permission\_classes** = [`<class 'rest_framework.permissions.IsAuthenticated'>`, `<class 'django_e2ee.permissions.HasEncryptionSecretPermission'>`]

**queryset**

**serializer\_class**

alias of *EncryptionKeySecretSerializer* Classes:

---

**Meta()**

---

**Methods:**

---

<b>create(validated_data)</b>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------	--

---

**class** django\_e2ee.views.**EncryptionKeyViewSet**(\*kwargs)

Bases: `ModelViewSet`

A viewset for e2ee.EncryptionKey

**Attributes:**

---

*basename*

---



---

*description*

---



---

*detail*

---



---

*name*

---



---

*permission\_classes*

---



---

*queryset*

---



---

*suffix*

---

**Methods:**

---

<i>get_queryset()</i>	Get the list of items for this view.
-----------------------	--------------------------------------

---

**Classes:**

---

<i>serializer_class</i>	alias of <i>EncryptionKeySerializer</i>
-------------------------	---

---

**basename** = None

**description** = None

**detail** = None

**get\_queryset()**

Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

**name** = None

**permission\_classes** = [`<class 'rest_framework.permissions.IsAuthenticated'>`]

**queryset**

**serializer\_class**

alias of *EncryptionKeySerializer* Classes:

---

Meta()

---

**Methods:**

---

`create(validated_data)`

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

---

**suffix** = None

**class** `django_e2ee.views.MasterKeyRetrieveView`(\*\*kwargs)

Bases: `RetrieveAPIView`

Retrieve the master key of the user.

**Methods:**

---

`get_object()`

Returns the object the view is displaying.

---

**Attributes:**

---

`permission_classes`

---

**Classes:**

---

`serializer_class`

alias of *MasterKeySerializer*

---



**get\_object()**

Returns the object the view is displaying.

You may want to override this if you need to provide non-standard queryset lookups. Eg if objects are referenced using multiple keyword arguments in the url conf.

**permission\_classes** = [`<class 'rest_framework.permissions.IsAuthenticated'>`]

**serializer\_class**

alias of *MasterKeySerializer* Classes:

---

`Meta()`

---

**Methods:**


---

<code>create(validated_data)</code>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------------	--

---

**class** `django_e2ee.views.MasterKeySecretViewSet(**kwargs)`

Bases: `ModelViewSet`

A viewset for a users `e2ee.MasterKeySecret`

**Attributes:**


---

*basename*

---



---

*description*

---



---

*detail*

---



---

*name*

---



---

*permission\_classes*

---



---

*queryset*

---



---

*suffix*

---

**Methods:**


---

<i>get_queryset()</i>	Get the list of items for this view.
-----------------------	--------------------------------------

---

**Classes:**


---

<i>serializer_class</i>	alias of <i>MasterKeySecretSerializer</i>
-------------------------	---

---

**basename** = `None`

**description** = None

**detail** = None

**get\_queryset()**

Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

**name** = None

**permission\_classes** = [`<class 'rest_framework.permissions.IsAuthenticated'>`]

**queryset**

**serializer\_class**

alias of *MasterKeySecretSerializer* Classes:

---

Meta()

---

**Methods:**

---

create(validated\_data)

---

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

---

**suffix** = None

**class** django\_e2ee.views.**MasterKeyViewSet**(*\*\*kwargs*)

Bases: *ModelViewSet*

A view to create and retrieve a users *e2ee.MasterKey*

**Attributes:**

---

*basename*

---

*description*

---

*detail*

---

*name*

---

*permission\_classes*

---

*queryset*

---

*suffix*

---

**Classes:**

<code>serializer_class</code>	alias of <code>MasterKeySerializer</code>
-------------------------------	---

`basename = None`

`description = None`

`detail = None`

`name = None`

`permission_classes = [<class 'rest_framework.permissions.IsAuthenticated'>, <class 'rest_framework.permissions.DjangoModelPermissions'>]`

`queryset`

`serializer_class`

alias of `MasterKeySerializer` **Classes:**

`Meta()`

**Methods:**

<code>create(validated_data)</code>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------------	--

`suffix = None`

**class** `django_e2ee.views.SessionKeyUpdateView(**kwargs)`

Bases: `RetrieveAPIView`, `UpdateAPIView`

Retreive the master key of the user.

**Methods:**

<code>get_object()</code>	Returns the object the view is displaying.
---------------------------	--

**Attributes:**

<code>permission_classes</code>
---------------------------------

**Classes:**

<code>serializer_class</code>	alias of <code>SessionKeySerializer</code>
-------------------------------	--

`get_object()`

Returns the object the view is displaying.

You may want to override this if you need to provide non-standard queryset lookups. Eg if objects are referenced using multiple keyword arguments in the url conf.

`permission_classes = [<class 'rest_framework.permissions.IsAuthenticated'>]`

`serializer_class`

alias of *SessionKeySerializer* Classes:

---

`Meta()`

---

**Methods:**

---

`create(validated_data)`

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

---

`class django_e2ee.views.SessionKeyViewSet(**kwargs)`

Bases: `ModelViewSet`

A viewset for a users `e2ee.SessionKey`

**Attributes:**

---

*basename*

---

*description*

---

*detail*

---

*name*

---

*permission\_classes*

---

*queryset*

---

*suffix*

---

**Methods:**

---

*get\_queryset()*

Get the list of items for this view.

---

**Classes:**

---

*serializer\_class*

alias of *SessionKeySerializer*

---

`basename = None`

`description = None`

`detail = None`

**get\_queryset()**

Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

**name** = None

**permission\_classes** = [`<class 'rest_framework.permissions.IsAuthenticated'>`]

**queryset**

**serializer\_class**

alias of *SessionKeySerializer* Classes:

---

Meta()

---

**Methods:**

---

create(validated\_data)

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

---

**suffix** = None

`django_e2ee.views.dummy_view(request, *args, **kwargs)`

## 3.5 django\_e2ee package

Django End-to-End Encryption Framework

An end-to-end encryption framework for Django

### 3.5.1 Subpackages

**django\_e2ee.templatetags package**

**Submodules**

**E2EE template tags**

Template tags for the django-e2ee-framework.

**Functions:**

<code>e2ee_enabled(session)</code>	Test if a session has e2e enabled.
<code>e2ee_ignored(session)</code>	Test if a session has e2e enabled.
<code>get_password_create_form()</code>	
<code>get_session_key(session)</code>	Get the <code>e2ee.SessionKey</code> for a session.
<code>get_session_key_form(context, session_key)</code>	Get a form to enable the session verification.
<code>get_session_keys(context)</code>	
<code>has_session_key(session)</code>	Test if a session has e2e enabled.

`django_e2ee.templatetags.e2ee.e2ee_enabled(session: Union[str, SessionStore, Session]) → bool`  
Test if a session has e2e enabled.

`django_e2ee.templatetags.e2ee.e2ee_ignored(session: Union[str, SessionStore, Session]) → bool`  
Test if a session has e2e enabled.

`django_e2ee.templatetags.e2ee.get_password_create_form()`

`django_e2ee.templatetags.e2ee.get_session_key(session: Union[str, SessionStore, Session]) → Optional[models.SessionKey]`  
Get the `e2ee.SessionKey` for a session.

`django_e2ee.templatetags.e2ee.get_session_key_form(context, session_key) → E2EESessionForm`  
Get a form to enable the session verification.

`django_e2ee.templatetags.e2ee.get_session_keys(context) → QuerySet[models.SessionKey]`

`django_e2ee.templatetags.e2ee.has_session_key(session: Union[str, SessionStore, Session]) → bool`  
Test if a session has e2e enabled.

### django\_e2ee.tests package

Tests for the `django_e2ee` app.

### Submodules

#### Tests for the django-e2ee-framework models

##### Functions:

<code>test_model()</code>	Place holder test to make sure pytest finds at least one.
---------------------------	---

`django_e2ee.tests.test_models.test_model()`  
Place holder test to make sure pytest finds at least one.  
Delete this once you really start to write some tests.

## Tests for the django-e2ee-framework views

### Classes:

<code>TestEncryptionKey()</code>	Unit tests for the EncryptionKey class in JavaScript.
<code>TestEncryptionKeyStore()</code>	Unit tests for the EncryptionKey class in JavaScript.

### Functions:

<code>test_encryption_decryption_post(e2e_browser, ...)</code>	Test encrypting a message through the service worker.
<code>test_master_key_creation(e2e_browser, test_user)</code>	Test the creation of the master key.
<code>test_master_key_secret_creation(e2e_browser, ...)</code>	Test the creation of another masterkey secret.
<code>test_remote_password_authentication(...)</code>	Test authentication of a new session via other session and password.
<code>test_remote_session_authentication(...)</code>	Test authentication of a new session via other session.
<code>test_session_authentication(e2e_browser, ...)</code>	Test authentication of a new session via password.

### class django\_e2ee.tests.test\_views.TestEncryptionKey

Bases: `object`

Unit tests for the EncryptionKey class in JavaScript.

#### Methods:

<code>get_encryption_key_js(driver)</code>	
<code>get_existing_encryption_key_js(driver, uuid)</code>	
<code>test_encrypt_decrypt(e2e_browser)</code>	Test encryption and decryption of a message.
<code>test_key_generation(e2e_browser)</code>	Test generating an encryption key.
<code>test_key_upload(e2e_browser)</code>	
<code>test_key_upload_for_multiple_users(...)</code>	Test the encryption key upload for multiple users.
<code>test_key_upload_for_other_user(...)</code>	

`get_encryption_key_js(driver: Remote)`

`get_existing_encryption_key_js(driver: Remote, uuid)`

`test_encrypt_decrypt(e2e_browser: Remote)`

Test encryption and decryption of a message.

`test_key_generation(e2e_browser: Remote)`

Test generating an encryption key.

`test_key_upload(e2e_browser: Remote)`

`test_key_upload_for_multiple_users(test_user_factory: Callable[[], User], e2e_browser_factory: Callable[[User], Remote])`

Test the encryption key upload for multiple users.

**test\_key\_upload\_for\_other\_user**(*test\_user\_factory: Callable[[], User], e2e\_browser\_factory: Callable[[User], Remote]*)

**class** django\_e2ee.tests.test\_views.**TestEncryptionKeyStore**

Bases: `object`

Unit tests for the EncryptionKey class in JavaScript.

**Methods:**

---

[\*get\\_encryption\\_key\\_store\\_js\*](#)(*driver*)

---

[\*test\\_decryption\*](#)(*e2e\_browser, test\_user*)      Test decrypting messages

[\*test\\_key\\_creation\*](#)(*e2e\_browser*)      Test the createKey method of an EncryptionKeyStore

[\*test\\_key\\_retrieval\*](#)(*e2e\_browser*)      Test the getKey method of an EncryptionKeyStore

---

**get\_encryption\_key\_store\_js**(*driver: Remote*)

**test\_decryption**(*e2e\_browser: Remote, test\_user: User*)

Test decrypting messages

**test\_key\_creation**(*e2e\_browser: Remote*)

Test the createKey method of an EncryptionKeyStore

**test\_key\_retrieval**(*e2e\_browser: Remote*)

Test the getKey method of an EncryptionKeyStore

django\_e2ee.tests.test\_views.**test\_encryption\_decryption\_post**(*e2e\_browser: Remote, test\_user: User, make\_request: Callable[[Remote, str, str], Optional[Dict]], Dict]*)

Test encrypting a message through the service worker.

django\_e2ee.tests.test\_views.**test\_master\_key\_creation**(*e2e\_browser: Remote, test\_user: User*)

Test the creation of the master key.

django\_e2ee.tests.test\_views.**test\_master\_key\_secret\_creation**(*e2e\_browser: Remote, make\_request: Callable[[Remote, str, str, Optional[Dict]], Dict], authenticated\_browser\_factory: Callable[[User], Remote], test\_user: User, dummy\_password: str)*)

Test the creation of another masterkey secret.

django\_e2ee.tests.test\_views.**test\_remote\_password\_authentication**(*e2e\_browser: Remote, test\_user: User, authenticated\_browser\_factory: Callable[[User], Remote], make\_request: Callable[[Remote, str, str, Optional[Dict]], Dict], dummy\_password: str)*)

Test authentication of a new session via other session and password.



```
django_e2ee.tests.test_views.test_remote_session_authentication(e2e_browser: Remote, test_user:
                                                                    User,
                                                                    authenticated_browser_factory:
                                                                    Callable[[User], Remote],
                                                                    make_request:
                                                                    Callable[[Remote, str, str,
                                                                    Optional[Dict]], Dict])
```

Test authentication of a new session via other session.

```
django_e2ee.tests.test_views.test_session_authentication(e2e_browser: Remote, test_user: User,
                                                                    authenticated_browser_factory:
                                                                    Callable[[User], Remote], make_request:
                                                                    Callable[[Remote, str, str,
                                                                    Optional[Dict]], Dict], dummy_password:
                                                                    str)
```

Test authentication of a new session via password.

### 3.5.2 Submodules

#### Admin interfaces

This module defines the django-e2ee-framework Admin interfaces.

##### Classes:

<i>EncryptionKeyAdmin</i> (model, admin_site)	An admin for encryption keys.
<i>EncryptionKeySecretInline</i> (parent_model, ...)	An inline for an encryption key secret.
<i>MasterKeyAdmin</i> (model, admin_site)	An admin for a master key
<i>MasterKeySecretInline</i> (parent_model, admin_site)	An inline for master key secrets.
<i>SessionKeyInline</i> (parent_model, admin_site)	An inline for session keys.

```
class django_e2ee.admin.EncryptionKeyAdmin(model, admin_site)
```

Bases: `ModelAdmin`

An admin for encryption keys.

##### Methods:

<i>has_add_permission</i> (request)	Return True if the given request has permission to add an object.
-------------------------------------	---

##### Attributes:

<i>inlines</i>
<i>list_display</i>
<i>media</i>

**has\_add\_permission**(*request*) → bool

Return True if the given request has permission to add an object. Can be overridden by the user in sub-classes.

**inlines** = [<class 'django\_e2ee.admin.EncryptionKeySecretInline'>]

**list\_display** = ['uuid', 'created\_by']

**property** media

**class** django\_e2ee.admin.**EncryptionKeySecretInline**(*parent\_model*, *admin\_site*)

Bases: StackedInline

An inline for an encryption key secret.

**Attributes:**

---

*extra*

---

*media*

---

*readonly\_fields*

---

**Methods:**

---

*has\_add\_permission*(*request*, *obj*)

Return True if the given request has permission to add an object.

---

**Models:**

---

*model*

alias of *EncryptionKeySecret*

---

**extra** = 0

**has\_add\_permission**(*request*, *obj*) → bool

Return True if the given request has permission to add an object. Can be overridden by the user in sub-classes.

**property** media

**model**

alias of *EncryptionKeySecret* **Miscellaneous:**

---

DoesNotExist

---

MultipleObjectsReturned

---

**Model Fields:**

<code>encrypted_with</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<code>encryption_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
<code>id</code>	A wrapper for a deferred-loading field.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>signature</code>	A wrapper for a deferred-loading field.
<code>signed_by</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

**Attributes:**

<code>encrypted_with_id</code>
<code>encryption_key_id</code>
<code>objects</code>
<code>signed_by_id</code>

```
readonly_fields = ['encrypted_with', 'secret', 'signature', 'signed_by']
```

```
class django_e2ee.admin.MasterKeyAdmin(model, admin_site)
```

Bases: `ModelAdmin`

An admin for a master key

**Methods:**

<code>has_add_permission(request)</code>	Return True if the given request has permission to add an object.
--	---

**Attributes:**

<code>inlines</code>
<code>list_display</code>
<code>media</code>
<code>readonly_fields</code>

```
has_add_permission(request) → bool
```

Return True if the given request has permission to add an object. Can be overridden by the user in subclasses.

```
inlines = [<class 'django_e2ee.admin.MasterKeySecretInline'>, <class  
'django_e2ee.admin.SessionKeyInline'>]
```

```
list_display = ['user']
```

```
property media
```

```
readonly_fields = ['pubkey', 'signing_pubkey']
```

```
class django_e2ee.admin.MasterKeySecretInline(parent_model, admin_site)
```

```
Bases: StackedInline
```

```
An inline for master key secrets.
```

**Attributes:**

---

*extra*

---

*media*

---

*readonly\_fields*

---

**Methods:**

---

*has\_add\_permission*(request, obj)

Return True if the given request has permission to add an object.

---

**Models:**

---

*model*

alias of *MasterKeySecret*

---

```
extra = 0
```

```
has_add_permission(request, obj) → bool
```

Return True if the given request has permission to add an object. Can be overridden by the user in sub-classes.

```
property media
```

```
model
```

alias of *MasterKeySecret* **Miscellaneous:**

---

DoesNotExist

---

MultipleObjectsReturned

---

**Model Fields:**

<code>identifier</code>	A wrapper for a deferred-loading field.
<code>iv</code>	A wrapper for a deferred-loading field.
<code>master_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via <code>ForwardOneToOneDescriptor</code> subclass) relation.
<code>salt</code>	A wrapper for a deferred-loading field.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>signing_secret</code>	A wrapper for a deferred-loading field.
<code>uuid</code>	A wrapper for a deferred-loading field.

**Attributes:**

<code>master_key_id</code>
<code>objects</code>

```
readonly_fields = ['uuid', 'secret', 'signing_secret', 'salt', 'iv']
```

```
class django_e2ee.admin.SessionKeyInline(parent_model, admin_site)
```

Bases: `StackedInline`

An inline for session keys.

**Attributes:**

<code>exclude</code>
<code>extra</code>
<code>media</code>
<code>readonly_fields</code>

**Methods:**

<code>has_add_permission(request, obj)</code>	Return True if the given request has permission to add an object.
---	---

**Models:**

<code>model</code>	alias of <code>SessionKey</code>
--------------------	----------------------------------

```
exclude = ['session']
```

```
extra = 0
```

```
has_add_permission(request, obj) → bool
```

Return True if the given request has permission to add an object. Can be overridden by the user in sub-classes.

**property media****model**alias of *SessionKey* Miscellaneous:

---

DoesNotExist

---

---

MultipleObjectsReturned

---

**Methods:**

---

get\_absolute\_url()

---

**Model Fields:**

ignore	A wrapper for a deferred-loading field.
iv	A wrapper for a deferred-loading field.
master_key	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
secret	A wrapper for a deferred-loading field.
session	Accessor to the related object on the forward side of a one-to-one relation.
session_secret	A wrapper for a deferred-loading field.
signing_secret	A wrapper for a deferred-loading field.

**Attributes:**

---

master\_key\_id

---

---

objects

---

---

session\_id

---

```
readonly_fields = ['session_secret', 'secret', 'signing_secret', 'iv']
```

**App config**

App config for the django\_e2ee app.

**Classes:**

---

*DjangoHelmholtzAaiConfig*(app\_name,  
app\_module)

---

```
class django_e2ee.apps.DjangoHelmholtzAaiConfig(app_name, app_module)
```

Bases: AppConfig

Attributes:

---

*default\_auto\_field*

---

*label*

---

*name*

---

```
default_auto_field = 'django.db.models.BigAutoField'
```

```
label = 'e2ee'
```

```
name = 'django_e2ee'
```

## Context processors

Context processors for the django\_e2ee app.

Functions:

---

*get\_e2ee\_login\_context\_data*(request)

---

django\_e2ee.context\_processors.get\_e2ee\_login\_context\_data(request) → Dict

Reusable forms of the django-e2ee-framework.

Classes:

<i>E2EESessionForm</i> ([data, files, auto_id, ...])	A form to setup E2EE for a session
<i>PasswordCreateForm</i> ([data, files, auto_id, ...])	A form for creting the E2EE-password.
<i>PasswordInputForm</i> ([data, files, auto_id, ...])	A form for entering the E2EE-password.

```
class django_e2ee.forms.E2EESessionForm(data=None, files=None, auto_id='id_%s', prefix=None,
    initial=None, error_class=<class
    'django.forms.utils.ErrorList'>, label_suffix=None,
    empty_permitted=False, field_order=None,
    use_required_attribute=None, renderer=None)
```

Bases: *PasswordInputForm*

A form to setup E2EE for a session

Methods:

<i>add_prefix</i> (field_name)	Return the field name with a prefix appended, if this Form has a prefix set.
--------------------------------	--

Attributes:

---

*base\_fields*

---

*declared\_fields*

---

<i>media</i>	Return all media required to render the widgets on this form.
--------------	---

---

**add\_prefix**(*field\_name*)

Return the field name with a prefix appended, if this Form has a prefix set.

Subclasses may wish to override.

```
base_fields = {'method': <django.forms.fields.CharField object>, 'password':
<django.forms.fields.CharField object>, 'session_secret':
<django.forms.fields.CharField object>, 'uuid':
<django.forms.models.ModelChoiceField object>, 'verification_number':
<django.forms.fields.IntegerField object>}
```

```
declared_fields = {'method': <django.forms.fields.CharField object>, 'password':
<django.forms.fields.CharField object>, 'session_secret':
<django.forms.fields.CharField object>, 'uuid':
<django.forms.models.ModelChoiceField object>, 'verification_number':
<django.forms.fields.IntegerField object>}
```

**property media**

Return all media required to render the widgets on this form.

```
class django_e2ee.forms.PasswordCreateForm(data=None, files=None, auto_id='id_%s', prefix=None,
initial=None, error_class=<class
'django.forms.utils.ErrorList'>, label_suffix=None,
empty_permitted=False, field_order=None,
use_required_attribute=None, renderer=None)
```

Bases: `Form`

A form for creting the E2EE-password.

**Classes:**

---

*Media*()

---

**Attributes:**

---

*base\_fields*

---

*declared\_fields*

---

<i>media</i>	Return all media required to render the widgets on this form.
--------------	---

---

**class Media**

Bases: `object`

**Attributes:**



---

*js*


---

```
js = ('js/e2ee/submit_form_as_json.js', 'js/e2ee/password_generator.js',
      'js/e2ee/fill_random_password.js')
```

```
base_fields = {'identifier': <django.forms.fields.CharField object>, 'password':
<django.forms.fields.CharField object>, 'show_password':
<django.forms.fields.BooleanField object>}
```

```
declared_fields = {'identifier': <django.forms.fields.CharField object>,
'password': <django.forms.fields.CharField object>, 'show_password':
<django.forms.fields.BooleanField object>}
```

#### property media

Return all media required to render the widgets on this form.

```
class django_e2ee.forms.PasswordInputForm(data=None, files=None, auto_id='id_%s', prefix=None,
initial=None, error_class=<class
'django.forms.utils.ErrorList'>, label_suffix=None,
empty_permitted=False, field_order=None,
use_required_attribute=None, renderer=None)
```

Bases: Form

A form for entering the E2EE-password.

#### Classes:

---

*Media()*


---

#### Attributes:

---

*base\_fields*


---



---

*declared\_fields*


---



---

*media*


---

Return all media required to render the widgets on this form.

---

#### class Media

Bases: *object*

#### Attributes:

---

*js*


---

```
js = ('js/e2ee/submit_form_as_json.js',)
```

```
base_fields = {'password': <django.forms.fields.CharField object>, 'uuid':
<django.forms.models.ModelChoiceField object>}
```

```
declared_fields = {'password': <django.forms.fields.CharField object>, 'uuid':  
<django.forms.models.ModelChoiceField object>}
```

**property media**

Return all media required to render the widgets on this form.

## Permissions

Custom permissions for the restAPI.

**Classes:**

---

<code>HasEncryptionSecretPermission()</code>	Check if the user has access to the e2ee. EncryptionKey.
--	---

---

```
class django_e2ee.permissions.HasEncryptionSecretPermission
```

Bases: BasePermission

Check if the user has access to the e2ee.EncryptionKey.

**Methods:**

---

<code>has_object_permission(request, view, obj)</code>	Return <i>True</i> if permission is granted, <i>False</i> otherwise.
--	--

---

**has\_object\_permission**(*request, view, obj*)

Return *True* if permission is granted, *False* otherwise.

## Serializers

Serializers for the django-e2ee-framework app.

**Classes:**

---

<code>BulkCreateListSerializer(*args, **kwargs)</code>	
--	--

---

<code>EncryptionKeySecretSerializer(*args, **kwargs)</code>	A serializer for e2ee.EncryptionKeySecret.
<code>EncryptionKeySerializer(*args, **kwargs)</code>	A serializer for e2ee.EncryptionKey.
<code>MasterKeySecretSerializer(*args, **kwargs)</code>	A serializer for e2ee.MasterKeySecret.
<code>MasterKeySerializer(*args, **kwargs)</code>	A serializer for e2ee.MasterKey.
<code>SessionKeySerializer(*args, **kwargs)</code>	A serializer for e2ee.SessionKey.

---

```
class django_e2ee.serializers.BulkCreateListSerializer(*args, **kwargs)
```

Bases: ListSerializer

**Methods:**

---

<code>create(validated_data)</code>
-------------------------------------

---

**create**(*validated\_data*)

**class** django\_e2ee.serializers.**EncryptionKeySecretSerializer**(\*args, \*\*kwargs)

Bases: `ModelSerializer`

A serializer for `e2ee.EncryptionKeySecret`.

**Classes:**

---

*Meta()*

---

**Methods:**

---

<i>create</i> (validated_data)	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
--------------------------------	--

---

**class** `Meta`

Bases: `object`

**Attributes:**

---

*fields*

---



---

*read\_only\_fields*

---

**Classes:**

---

<i>list_serializer_class</i>	alias of <i>BulkCreateListSerializer</i>
------------------------------	--

---

**Models:**

---

<i>model</i>	alias of <i>EncryptionKeySecret</i>
--------------	-------------------------------------

---

**fields** = ['encryption\_key', 'encrypted\_with', 'secret', 'signature', 'signed\_by']

**list\_serializer\_class**

alias of *BulkCreateListSerializer* **Methods:**

---

*create*(validated\_data)

---

**model**

alias of *EncryptionKeySecret* **Miscellaneous:**

---

*DoesNotExist*

---



---

*MultipleObjectsReturned*

---

**Model Fields:**

encrypted_with	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
encryption_key	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
id	A wrapper for a deferred-loading field.
secret	A wrapper for a deferred-loading field.
signature	A wrapper for a deferred-loading field.
signed_by	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

**Attributes:**

encrypted_with_id
encryption_key_id
objects
signed_by_id

```
read_only_fields = ['encryption_key', 'signed_by']
```

```
create(validated_data)
```

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

```
return ExampleModel.objects.create(**validated_data)
```

If there are many to many fields present on the instance then they cannot be set until the model is instantiated, in which case the implementation is like so:

```
example_relationship = validated_data.pop('example_relationship') instance = Example-  
Model.objects.create(**validated_data) instance.example_relationship = example_relationship  
return instance
```

The default implementation also does not handle nested relationships. If you want to support writable nested relationships you'll need to write an explicit `.create()` method.

```
class django_e2ee.serializers.EncryptionKeySerializer(*args, **kwargs)
```

```
Bases: ModelSerializer
```

```
A serializer for e2ee.EncryptionKey.
```

**Classes:**

---

```
Meta()
```

---

**Methods:**

<code>create(validated_data)</code>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------------	--

#### class Meta

Bases: `object`

##### Attributes:

`fields`

##### Models:

`model`

alias of `EncryptionKey`

`fields = ['uuid']`

##### model

alias of `EncryptionKey` Miscellaneous:

`DoesNotExist`

`MultipleObjectsReturned`

##### Model Fields:

`created_by`

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

`uuid`

A wrapper for a deferred-loading field.

##### Attributes:

`created_by_id`

`encryptionkeysecret_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.

`master_keys`

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

`objects`

#### `create(validated_data)`

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

```
return ExampleModel.objects.create(**validated_data)
```

If there are many to many fields present on the instance then they cannot be set until the model is instantiated, in which case the implementation is like so:

```
example_relationship = validated_data.pop('example_relationship') instance = Example-
Model.objects.create(**validated_data) instance.example_relationship = example_relationship
return instance
```

The default implementation also does not handle nested relationships. If you want to support writable nested relationships you'll need to write an explicit `.create()` method.

**class** django\_e2ee.serializers.**MasterKeySecretSerializer**(\*args, \*\*kwargs)

Bases: `ModelSerializer`

A serializer for `e2ee.MasterKeySecret`.

**Classes:**

---

*Meta()*

---

**Methods:**

---

<i>create</i> (validated_data)	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
--------------------------------	--

---

**class** **Meta**

Bases: `object`

**Attributes:**

---

*fields*

---

**Models:**

---

<i>model</i>	alias of <i>MasterKeySecret</i>
--------------	---------------------------------

---

**fields** = ['identifier', 'secret', 'signing\_secret', 'salt', 'iv', 'uuid']

**model**

alias of *MasterKeySecret* **Miscellaneous:**

---

`DoesNotExist`

---



---

`MultipleObjectsReturned`

---

**Model Fields:**

<code>identifier</code>	A wrapper for a deferred-loading field.
<code>iv</code>	A wrapper for a deferred-loading field.
<code>master_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via <code>ForwardOneToOneDescriptor</code> subclass) relation.
<code>salt</code>	A wrapper for a deferred-loading field.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>signing_secret</code>	A wrapper for a deferred-loading field.
<code>uuid</code>	A wrapper for a deferred-loading field.

**Attributes:**

<code>master_key_id</code>
<code>objects</code>

**create(validated\_data)**

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

```
return ExampleModel.objects.create(**validated_data)
```

If there are many to many fields present on the instance then they cannot be set until the model is instantiated, in which case the implementation is like so:

```
example_relationship = validated_data.pop('example_relationship') instance = Example-
Model.objects.create(**validated_data) instance.example_relationship = example_relationship
return instance
```

The default implementation also does not handle nested relationships. If you want to support writable nested relationships you'll need to write an explicit `.create()` method.

```
class django_e2ee.serializers.MasterKeySerializer(*args, **kwargs)
```

Bases: `ModelSerializer`

A serialializer for `e2ee.MasterKey`.

**Classes:**

`Meta()`

**Methods:**

<code>create(validated_data)</code>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------------	--

**class Meta**

Bases: `object`

**Attributes:**

---

*fields*

---

*read\_only\_fields*

---

**Models:**

---

*model*alias of *MasterKey*

---

**fields = ['pubkey', 'signing\_pubkey', 'user']****model**alias of *MasterKey* **Miscellaneous:**

---

DoesNotExist

---

MultipleObjectsReturned

---

**Attributes:**

---

encryptionkey\_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

---

encryptionkeysecret\_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

---

masterkeysecret\_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

---

objects

---

pubkey\_loaded

The pubkey loaded via cryptography

---

sessionkey\_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

---

signed\_secrets

Accessor to the related objects manager on the reverse side of a many-to-one relation.

---

signing\_pubkey\_loaded

The pubkey loaded via cryptography

---

user\_id

---

**Model Fields:**

---

pubkey

A wrapper for a deferred-loading field.

---

signing\_pubkey

A wrapper for a deferred-loading field.

---

userAccessor to the related object on the forward side of a one-to-one relation.

---

**read\_only\_fields = ['user']****create(validated\_data)**

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:



```
return ExampleModel.objects.create(**validated_data)
```

If there are many to many fields present on the instance then they cannot be set until the model is instantiated, in which case the implementation is like so:

```
example_relationship = validated_data.pop('example_relationship') instance = Example-
Model.objects.create(**validated_data) instance.example_relationship = example_relationship
return instance
```

The default implementation also does not handle nested relationships. If you want to support writable nested relationships you'll need to write an explicit `.create()` method.

```
class django_e2ee.serializers.SessionKeySerializer(*args, **kwargs)
```

Bases: `ModelSerializer`

A serialializer for `e2ee.SessionKey`.

**Classes:**

---

`Meta()`

---

**Methods:**

---

<code>create(validated_data)</code>	We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:
-------------------------------------	--

---

```
class Meta
```

Bases: `object`

**Attributes:**

---

`fields`

---



---

`read_only_fields`

---

**Models:**

---

<code>model</code>	alias of <code>SessionKey</code>
--------------------	----------------------------------

---

```
fields = ['session', 'session_secret', 'secret', 'signing_secret', 'iv',
'ignore']
```

```
model
```

alias of `SessionKey` Miscellaneous:

---

`DoesNotExist`

---



---

`MultipleObjectsReturned`

---

**Methods:**

---

`get_absolute_url()`

---

**Model Fields:**

<code>ignore</code>	A wrapper for a deferred-loading field.
<code>iv</code>	A wrapper for a deferred-loading field.
<code>master_key</code>	Accessor to the related object on the forward side of a many-to-one or one-to-one (via <code>ForwardOneToOneDescriptor</code> subclass) relation.
<code>secret</code>	A wrapper for a deferred-loading field.
<code>session</code>	Accessor to the related object on the forward side of a one-to-one relation.
<code>session_secret</code>	A wrapper for a deferred-loading field.
<code>signing_secret</code>	A wrapper for a deferred-loading field.

**Attributes:**

---

`master_key_id`

---

---

`objects`

---

---

`session_id`

---

```
read_only_fields = ['session']
```

```
create(validated_data)
```

We have a bit of extra checking around this in order to provide descriptive messages when something goes wrong, but this method is essentially just:

```
    return ExampleModel.objects.create(**validated_data)
```

If there are many to many fields present on the instance then they cannot be set until the model is instantiated, in which case the implementation is like so:

```
    example_relationship = validated_data.pop('example_relationship') instance = Example-  
    Model.objects.create(**validated_data) instance.example_relationship = example_relationship  
    return instance
```

The default implementation also does not handle nested relationships. If you want to support writable nested relationships you'll need to write an explicit `.create()` method.

## CONTRIBUTION AND DEVELOPMENT HINTS

The django-e2ee-framework project is developed by the [Helmholtz-Zentrum Hereon](#). It is open-source as we believe that this package can be helpful for multiple other django applications, and we are looking forward for your feedback, questions and especially for your contributions.

- If you want to ask a question, are missing a feature or have comments on the docs, please [open an issue at the source code repository](#)
- If you have suggestions for improvement, please let us know in an issue, or fork the repository and create a merge request. See also *Contributing in the development*.

### 4.1 Contributing in the development

Thanks for your wish to contribute to this app!! The source code of the django-e2ee-framework package is hosted at <https://gitlab.hzdr.de/hcdc/django/django-e2ee-framework>. It's an open gitlab where you can register via GitHub, or via the Helmholtz AAI. Once you created an account, you can [fork](#) this repository to your own user account and implement the changes. Afterwards, please make a merge request into the main repository. If you have any questions, please do not hesitate to create an issue on gitlab and contact the developers.

**Warning:** For local development, you need a postgres and redis server available. They can be configured via environment variables (e.g. POSTGRES\_HOST REDIS\_HOST, see the `settings.py` file in the testproject).

Once you created you fork, you can clone it via

```
git clone https://gitlab.hzdr.de/<your-user>/django-e2ee-framework.git
```

and install it in development mode with the `[dev]` option via:

```
pip install -e ./django-e2ee-framework/[dev]
```

Once you installed the package, run the migrations:

```
cd django-e2ee-framework/  
python manage.py migrate
```

which will setup the database for you.

### 4.1.1 Fixing the docs

The documentation for this package is written in restructured Text and built with [sphinx](#) and deployed on [readthedocs](#).

If you found something in the docs that you want to fix, head over to the docs folder and build the docs with *make html* (or *make.bat* on windows). The docs are then available in `docs/_build/html/index.html` that you can open with your local browser.

Implement your fixes in the corresponding `.rst`-file and push them to your fork on gitlab.

### 4.1.2 Contributing to the code

We use automated formatters (see their config in `pyproject.toml` and `setup.cfg`), namely

- [Black](#) for standardized code formatting
- [blackdoc](#) for standardized code formatting in documentation
- [Flake8](#) for general code quality
- [isort](#) for standardized order in imports.
- [mypy](#) for static type checking on [type hints](#)

We highly recommend that you setup [pre-commit hooks](#) to automatically run all the above tools every time you make a git commit. This can be done by running:

```
pre-commit install
```

from the root of the repository. You can skip the pre-commit checks with `git commit --no-verify` but note that the CI will fail if it encounters any formatting errors.

You can also run the pre-commit step manually by invoking:

```
pre-commit run --all-files
```

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### d

- `django_e2ee`, 25
- `django_e2ee.admin`, 29
- `django_e2ee.app_settings`, 7
- `django_e2ee.apps`, 34
- `django_e2ee.context_processors`, 35
- `django_e2ee.forms`, 35
- `django_e2ee.models`, 7
- `django_e2ee.permissions`, 38
- `django_e2ee.serializers`, 38
- `django_e2ee.templatetags`, 25
- `django_e2ee.templatetags.e2ee`, 25
- `django_e2ee.tests`, 26
- `django_e2ee.tests.test_models`, 26
- `django_e2ee.tests.test_views`, 26
- `django_e2ee.urls`, 7
- `django_e2ee.views`, 17





## A

`add_prefix()` (*django\_e2ee.forms.E2EESessionForm* method), 36  
`app_name` (in module *django\_e2ee.urls*), 7

## B

`base_fields` (*django\_e2ee.forms.E2EESessionForm* attribute), 36  
`base_fields` (*django\_e2ee.forms.PasswordCreateForm* attribute), 37  
`base_fields` (*django\_e2ee.forms.PasswordInputForm* attribute), 37  
`basename` (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 19  
`basename` (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 21  
`basename` (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
`basename` (*django\_e2ee.views.SessionKeyViewSet* attribute), 24  
`BulkCreateListSerializer` (class in *django\_e2ee.serializers*), 38

## C

`create()` (*django\_e2ee.serializers.BulkCreateListSerializer* method), 38  
`create()` (*django\_e2ee.serializers.EncryptionKeySecretSerializer* method), 40  
`create()` (*django\_e2ee.serializers.EncryptionKeySerializer* method), 41  
`create()` (*django\_e2ee.serializers.MasterKeySecretSerializer* method), 43  
`create()` (*django\_e2ee.serializers.MasterKeySerializer* method), 44  
`create()` (*django\_e2ee.serializers.SessionKeySerializer* method), 46  
`created_by` (*django\_e2ee.models.EncryptionKey* attribute), 8  
`created_by_id` (*django\_e2ee.models.EncryptionKey* attribute), 9

## D

`declared_fields` (*django\_e2ee.forms.E2EESessionForm* attribute), 36  
`declared_fields` (*django\_e2ee.forms.PasswordCreateForm* attribute), 37  
`declared_fields` (*django\_e2ee.forms.PasswordInputForm* attribute), 37  
`default_auto_field` (*django\_e2ee.apps.DjangoHelmholtzAaiConfig* attribute), 35  
`delete_session_key()` (in module *django\_e2ee.models*), 17  
`description` (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 19  
`description` (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 21  
`description` (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
`description` (*django\_e2ee.views.SessionKeyViewSet* attribute), 24  
`detail` (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 20  
`detail` (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 22  
`detail` (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
`detail` (*django\_e2ee.views.SessionKeyViewSet* attribute), 24  
`django_e2ee` module, 25  
`django_e2ee.admin` module, 29  
`django_e2ee.app_settings` module, 7  
`django_e2ee.apps` module, 34  
`django_e2ee.context_processors` module, 35  
`django_e2ee.forms` module, 35  
`django_e2ee.models` module, 7  
`django_e2ee.permissions`

module, 38  
 django\_e2ee.serializers  
   module, 38  
 django\_e2ee.templatetags  
   module, 25  
 django\_e2ee.templatetags.e2ee  
   module, 25  
 django\_e2ee.tests  
   module, 26  
 django\_e2ee.tests.test\_models  
   module, 26  
 django\_e2ee.tests.test\_views  
   module, 26  
 django\_e2ee.urls  
   module, 7  
 django\_e2ee.views  
   module, 17  
 DjangoHelmholtzAaiConfig (class in  
   django\_e2ee.apps), 34  
 dummy\_view() (in module django\_e2ee.views), 25

## E

e2ee\_enabled() (in module  
   django\_e2ee.templatetags.e2ee), 26  
 e2ee\_ignored() (in module  
   django\_e2ee.templatetags.e2ee), 26  
 E2EESessionForm (class in django\_e2ee.forms), 35  
 E2ELoginViewMixin (class in django\_e2ee.views), 17  
 encrypted\_with (django\_e2ee.models.EncryptionKeySecret  
   attribute), 10  
 encrypted\_with\_id (django\_e2ee.models.EncryptionKeySecret  
   attribute), 10  
 encryption\_key (django\_e2ee.models.EncryptionKeySecret  
   attribute), 10  
 encryption\_key\_id (django\_e2ee.models.EncryptionKeySecret  
   attribute), 10  
 EncryptionKey (class in django\_e2ee.models), 8  
 EncryptionKey.DoesNotExist, 8  
 EncryptionKey.MultipleObjectsReturned, 8  
 encryptionkey\_set (django\_e2ee.models.MasterKey  
   attribute), 12  
 EncryptionKeyAdmin (class in django\_e2ee.admin), 29  
 EncryptionKeySecret (class in django\_e2ee.models),  
   9  
 EncryptionKeySecret.DoesNotExist, 10  
 EncryptionKeySecret.MultipleObjectsReturned,  
   10  
 encryptionkeysecret\_set  
   (django\_e2ee.models.EncryptionKey attribute),  
   9  
 encryptionkeysecret\_set  
   (django\_e2ee.models.MasterKey attribute),  
   12

EncryptionKeySecretInline (class in  
   django\_e2ee.admin), 30  
 EncryptionKeySecretSerializer (class in  
   django\_e2ee.serializers), 38  
 EncryptionKeySecretSerializer.Meta (class in  
   django\_e2ee.serializers), 39  
 EncryptionKeySecretView (class in  
   django\_e2ee.views), 18  
 EncryptionKeySerializer (class in  
   django\_e2ee.serializers), 40  
 EncryptionKeySerializer.Meta (class in  
   django\_e2ee.serializers), 41  
 EncryptionKeyViewSet (class in django\_e2ee.views),  
   19  
 exclude (django\_e2ee.admin.SessionKeyInline at-  
   tribute), 33  
 extra (django\_e2ee.admin.EncryptionKeySecretInline  
   attribute), 30  
 extra (django\_e2ee.admin.MasterKeySecretInline  
   attribute), 32  
 extra (django\_e2ee.admin.SessionKeyInline attribute),  
   33

## F

fields (django\_e2ee.serializers.EncryptionKeySecretSerializer.Meta  
   attribute), 39  
 fields (django\_e2ee.serializers.EncryptionKeySerializer.Meta  
   attribute), 41  
 fields (django\_e2ee.serializers.MasterKeySecretSerializer.Meta  
   attribute), 42  
 fields (django\_e2ee.serializers.MasterKeySerializer.Meta  
   attribute), 44  
 fields (django\_e2ee.serializers.SessionKeySerializer.Meta  
   attribute), 45

## G

get\_absolute\_url() (django\_e2ee.models.SessionKey  
   method), 16  
 get\_context\_data() (django\_e2ee.views.E2ELoginViewMixin  
   method), 18  
 get\_e2ee\_login\_context\_data() (in module  
   django\_e2ee.context\_processors), 35  
 get\_encryption\_key\_js()  
   (django\_e2ee.tests.test\_views.TestEncryptionKey  
   method), 27  
 get\_encryption\_key\_store\_js()  
   (django\_e2ee.tests.test\_views.TestEncryptionKeyStore  
   method), 28  
 get\_existing\_encryption\_key\_js()  
   (django\_e2ee.tests.test\_views.TestEncryptionKey  
   method), 27  
 get\_object() (django\_e2ee.views.EncryptionKeySecretView  
   method), 18

`get_object()` (*django\_e2ee.views.MasterKeyRetrieveView* inline method), 20  
`get_object()` (*django\_e2ee.views.SessionKeyUpdateView* inline method), 23  
`get_password_create_form()` (in module *iv* (*django\_e2ee.models.MasterKeySecret* attribute), 14  
*django\_e2ee.templatetags.e2ee*), 26 *iv* (*django\_e2ee.models.SessionKey* attribute), 16  
`get_queryset()` (*django\_e2ee.views.EncryptionKeySecretView* method), 18  
`get_queryset()` (*django\_e2ee.views.EncryptionKeyViewSet* method), 20  
`get_queryset()` (*django\_e2ee.views.MasterKeySecretViewSet* method), 22  
`get_queryset()` (*django\_e2ee.views.SessionKeyViewSet* method), 24  
`get_serializer()` (*django\_e2ee.views.EncryptionKeySecretView* method), 18  
`get_session_key()` (in module *django\_e2ee.templatetags.e2ee*), 26  
`get_session_key_form()` (in module *django\_e2ee.templatetags.e2ee*), 26  
`get_session_keys()` (in module *django\_e2ee.templatetags.e2ee*), 26  
**H**  
`has_add_permission()` (*django\_e2ee.admin.EncryptionKeyAdmin* method), 29  
`has_add_permission()` (*django\_e2ee.admin.EncryptionKeySecretInline* method), 30  
`has_add_permission()` (*django\_e2ee.admin.MasterKeyAdmin* method), 31  
`has_add_permission()` (*django\_e2ee.admin.MasterKeySecretInline* method), 32  
`has_add_permission()` (*django\_e2ee.admin.SessionKeyInline* method), 33  
`has_object_permission()` (*django\_e2ee.permissions.HasEncryptionSecretPermission* method), 38  
`has_session_key()` (in module *django\_e2ee.templatetags.e2ee*), 26  
`HasEncryptionSecretPermission` (class in *django\_e2ee.permissions*), 38  
**I**  
`id` (*django\_e2ee.models.EncryptionKeySecret* attribute), 11  
`identifier` (*django\_e2ee.models.MasterKeySecret* attribute), 14  
`ignore` (*django\_e2ee.models.SessionKey* attribute), 16  
`iv` (*django\_e2ee.models.MasterKeySecret* attribute), 14  
`iv` (*django\_e2ee.models.SessionKey* attribute), 16  
**J**  
`label` (*django\_e2ee.forms.PasswordCreateForm.Media* attribute), 37  
**L**  
`list_display` (*django\_e2ee.admin.EncryptionKeyAdmin* attribute), 30  
`list_display` (*django\_e2ee.admin.MasterKeyAdmin* attribute), 32  
`list_serializer_class` (*django\_e2ee.serializers.EncryptionKeySecretSerializer.Meta* attribute), 39  
**M**  
`master_key` (*django\_e2ee.models.MasterKeySecret* attribute), 14  
`master_key` (*django\_e2ee.models.SessionKey* attribute), 16  
`master_key_id` (*django\_e2ee.models.MasterKeySecret* attribute), 15  
`master_key_id` (*django\_e2ee.models.SessionKey* attribute), 16  
`master_keys` (*django\_e2ee.models.EncryptionKey* attribute), 9  
`MasterKey` (class in *django\_e2ee.models*), 11  
`MasterKey.DoesNotExist`, 12  
`MasterKey.MultipleObjectsReturned`, 12  
`MasterKeyAdmin` (class in *django\_e2ee.admin*), 31  
`MasterKeyRetrieveView` (class in *django\_e2ee.views*), 20  
`MasterKeySecret` (class in *django\_e2ee.models*), 14  
`MasterKeySecret.DoesNotExist`, 14  
`MasterKeySecret.MultipleObjectsReturned`, 14  
`masterkeysecret_set` (*django\_e2ee.models.MasterKey* attribute), 12  
`MasterKeySecretInline` (class in *django\_e2ee.admin*), 32  
`MasterKeySecretSerializer` (class in *django\_e2ee.serializers*), 42  
`MasterKeySecretSerializer.Meta` (class in *django\_e2ee.serializers*), 42

MasterKeySecretViewSet	(class in <code>django_e2ee.urls</code> , 7 <code>django_e2ee.views</code> ), 21	
MasterKeySerializer	(class in <code>django_e2ee.serializers</code> ), 43	
MasterKeySerializer.Meta	(class in <code>django_e2ee.serializers</code> ), 43	
MasterKeyViewSet	(class in <code>django_e2ee.views</code> ), 22	
media	( <code>django_e2ee.admin.EncryptionKeyAdmin</code> property), 30	
media	( <code>django_e2ee.admin.EncryptionKeySecretInline</code> property), 30	
media	( <code>django_e2ee.admin.MasterKeyAdmin</code> property), 32	
media	( <code>django_e2ee.admin.MasterKeySecretInline</code> property), 32	
media	( <code>django_e2ee.admin.SessionKeyInline</code> property), 33	
media	( <code>django_e2ee.forms.E2EESessionForm</code> property), 36	
media	( <code>django_e2ee.forms.PasswordCreateForm</code> property), 37	
media	( <code>django_e2ee.forms.PasswordInputForm</code> property), 38	
model	( <code>django_e2ee.admin.EncryptionKeySecretInline</code> attribute), 30	
model	( <code>django_e2ee.admin.MasterKeySecretInline</code> attribute), 32	
model	( <code>django_e2ee.admin.SessionKeyInline</code> attribute), 34	
model	( <code>django_e2ee.serializers.EncryptionKeySecretSerializer.Meta</code> attribute), 39	
model	( <code>django_e2ee.serializers.EncryptionKeySerializer.Meta</code> attribute), 41	
model	( <code>django_e2ee.serializers.MasterKeySecretSerializer.Meta</code> attribute), 42	
model	( <code>django_e2ee.serializers.MasterKeySerializer.Meta</code> attribute), 44	
model	( <code>django_e2ee.serializers.SessionKeySerializer.Meta</code> attribute), 45	
module	<code>django_e2ee</code> , 25	
	<code>django_e2ee.admin</code> , 29	
	<code>django_e2ee.app_settings</code> , 7	
	<code>django_e2ee.apps</code> , 34	
	<code>django_e2ee.context_processors</code> , 35	
	<code>django_e2ee.forms</code> , 35	
	<code>django_e2ee.models</code> , 7	
	<code>django_e2ee.permissions</code> , 38	
	<code>django_e2ee.serializers</code> , 38	
	<code>django_e2ee.templatetags</code> , 25	
	<code>django_e2ee.templatetags.e2ee</code> , 25	
	<code>django_e2ee.tests</code> , 26	
	<code>django_e2ee.tests.test_models</code> , 26	
	<code>django_e2ee.tests.test_views</code> , 26	
		<b>N</b>
		name ( <code>django_e2ee.apps.DjangoHelmholtzAaiConfig</code> attribute), 35
		name ( <code>django_e2ee.views.EncryptionKeyViewSet</code> attribute), 20
		name ( <code>django_e2ee.views.MasterKeySecretViewSet</code> attribute), 22
		name ( <code>django_e2ee.views.MasterKeyViewSet</code> attribute), 23
		name ( <code>django_e2ee.views.SessionKeyViewSet</code> attribute), 25
		<b>O</b>
		objects ( <code>django_e2ee.models.EncryptionKey</code> attribute), 9
		objects ( <code>django_e2ee.models.EncryptionKeySecret</code> attribute), 11
		objects ( <code>django_e2ee.models.MasterKey</code> attribute), 13
		objects ( <code>django_e2ee.models.MasterKeySecret</code> attribute), 15
		objects ( <code>django_e2ee.models.SessionKey</code> attribute), 16
		<b>P</b>
		<code>PasswordCreateForm</code> (class in <code>django_e2ee.forms</code> ), 36
		<code>PasswordCreateForm.Media</code> (class in <code>django_e2ee.forms</code> ), 36
		<code>PasswordInputForm</code> (class in <code>django_e2ee.forms</code> ), 37
		<code>PasswordInputForm.Media</code> (class in <code>django_e2ee.forms</code> ), 37
		permission_classes ( <code>django_e2ee.views.EncryptionKeySecretViewSet</code> attribute), 19
		permission_classes ( <code>django_e2ee.views.EncryptionKeyViewSet</code> attribute), 20
		permission_classes ( <code>django_e2ee.views.MasterKeyRetrieveView</code> attribute), 21
		permission_classes ( <code>django_e2ee.views.MasterKeySecretViewSet</code> attribute), 22
		permission_classes ( <code>django_e2ee.views.MasterKeyViewSet</code> attribute), 23
		permission_classes ( <code>django_e2ee.views.SessionKeyUpdateView</code> attribute), 23
		permission_classes ( <code>django_e2ee.views.SessionKeyViewSet</code> attribute), 25
		pubkey ( <code>django_e2ee.models.MasterKey</code> attribute), 13
		pubkey_loaded ( <code>django_e2ee.models.MasterKey</code> property), 13
		<b>Q</b>
		queryset ( <code>django_e2ee.views.EncryptionKeySecretViewSet</code> attribute), 19

queryset (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 20  
 queryset (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 22  
 queryset (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
 queryset (*django\_e2ee.views.SessionKeyViewSet* attribute), 25

## R

read\_only\_fields (*django\_e2ee.serializers.EncryptionKeySecretSerializer.Meta* attribute), 40  
 read\_only\_fields (*django\_e2ee.serializers.MasterKeySerializer.Meta* attribute), 44  
 read\_only\_fields (*django\_e2ee.serializers.SessionKeySerializer.Meta* attribute), 46  
 readonly\_fields (*django\_e2ee.admin.EncryptionKeySecretInline* attribute), 31  
 readonly\_fields (*django\_e2ee.admin.MasterKeyAdmin* attribute), 32  
 readonly\_fields (*django\_e2ee.admin.MasterKeySecretInline* attribute), 33  
 readonly\_fields (*django\_e2ee.admin.SessionKeyInline* attribute), 34

## S

salt (*django\_e2ee.models.MasterKeySecret* attribute), 15  
 secret (*django\_e2ee.models.EncryptionKeySecret* attribute), 11  
 secret (*django\_e2ee.models.MasterKeySecret* attribute), 15  
 secret (*django\_e2ee.models.SessionKey* attribute), 16  
 serializer\_class (*django\_e2ee.views.EncryptionKeySecretViewSet* attribute), 19  
 serializer\_class (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 20  
 serializer\_class (*django\_e2ee.views.MasterKeyRetrieveView* attribute), 21  
 serializer\_class (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 22  
 serializer\_class (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
 serializer\_class (*django\_e2ee.views.SessionKeyUpdateView* attribute), 24  
 serializer\_class (*django\_e2ee.views.SessionKeyViewSet* attribute), 25  
 session (*django\_e2ee.models.SessionKey* attribute), 16  
 session\_id (*django\_e2ee.models.SessionKey* attribute), 17  
 session\_secret (*django\_e2ee.models.SessionKey* attribute), 17  
 SessionKey (class in *django\_e2ee.models*), 15  
 SessionKey.DoesNotExist, 16

SessionKey.MultipleObjectsReturned, 16  
 sessionkey\_set (*django\_e2ee.models.MasterKey* attribute), 13  
 SessionKeyInline (class in *django\_e2ee.admin*), 33  
 SessionKeySerializer (class in *django\_e2ee.serializers*), 45  
 SessionKeySerializer.Meta (class in *django\_e2ee.serializers*), 45  
 SessionKeyUpdateView (class in *django\_e2ee.views*), 23  
 SessionKeyViewSet (class in *django\_e2ee.views*), 24  
 signature (*django\_e2ee.models.EncryptionKeySecret* attribute), 11  
 signed\_by (*django\_e2ee.models.EncryptionKeySecret* attribute), 11  
 signed\_by\_id (*django\_e2ee.models.EncryptionKeySecret* attribute), 11  
 signed\_secrets (*django\_e2ee.models.MasterKey* attribute), 13  
 signing\_pubkey (*django\_e2ee.models.MasterKey* attribute), 13  
 signing\_pubkey\_loaded (*django\_e2ee.models.MasterKey* property), 13  
 signing\_secret (*django\_e2ee.models.MasterKeySecret* attribute), 15  
 signing\_secret (*django\_e2ee.models.SessionKey* attribute), 17  
 suffix (*django\_e2ee.views.EncryptionKeyViewSet* attribute), 20  
 suffix (*django\_e2ee.views.MasterKeySecretViewSet* attribute), 22  
 suffix (*django\_e2ee.views.MasterKeyViewSet* attribute), 23  
 suffix (*django\_e2ee.views.SessionKeyViewSet* attribute), 25

## T

test\_decryption() (*django\_e2ee.tests.test\_views.TestEncryptionKeyStore* method), 28  
 test\_encrypt\_decrypt() (*django\_e2ee.tests.test\_views.TestEncryptionKeyStore* method), 27  
 test\_encryption\_decryption\_post() (in module *django\_e2ee.tests.test\_views*), 28  
 test\_key\_creation() (*django\_e2ee.tests.test\_views.TestEncryptionKeyStore* method), 28  
 test\_key\_generation() (*django\_e2ee.tests.test\_views.TestEncryptionKeyStore* method), 27  
 test\_key\_retrieval() (*django\_e2ee.tests.test\_views.TestEncryptionKeyStore* method), 28



[test\\_key\\_upload\(\)](#) (*django\_e2ee.tests.test\_views.TestEncryptionKey*  
*method*), [27](#)  
[test\\_key\\_upload\\_for\\_multiple\\_users\(\)](#)  
*(django\_e2ee.tests.test\_views.TestEncryptionKey*  
*method)*, [27](#)  
[test\\_key\\_upload\\_for\\_other\\_user\(\)](#)  
*(django\_e2ee.tests.test\_views.TestEncryptionKey*  
*method)*, [28](#)  
[test\\_master\\_key\\_creation\(\)](#) (*in module*  
*django\_e2ee.tests.test\_views*), [28](#)  
[test\\_master\\_key\\_secret\\_creation\(\)](#) (*in module*  
*django\_e2ee.tests.test\_views*), [28](#)  
[test\\_model\(\)](#) (*in module*  
*django\_e2ee.tests.test\_models*), [26](#)  
[test\\_remote\\_password\\_authentication\(\)](#) (*in mod-*  
*ule django\_e2ee.tests.test\_views*), [28](#)  
[test\\_remote\\_session\\_authentication\(\)](#) (*in mod-*  
*ule django\_e2ee.tests.test\_views*), [28](#)  
[test\\_session\\_authentication\(\)](#) (*in module*  
*django\_e2ee.tests.test\_views*), [29](#)  
[TestEncryptionKey](#) (*class in*  
*django\_e2ee.tests.test\_views*), [27](#)  
[TestEncryptionKeyStore](#) (*class in*  
*django\_e2ee.tests.test\_views*), [28](#)

## U

[user](#) (*django\_e2ee.models.MasterKey attribute*), [13](#)  
[user\\_id](#) (*django\_e2ee.models.MasterKey attribute*), [14](#)  
[uuid](#) (*django\_e2ee.models.EncryptionKey attribute*), [9](#)  
[uuid](#) (*django\_e2ee.models.MasterKeySecret attribute*),  
[15](#)

## V

[validate\\_public\\_key\(\)](#) (*in module*  
*django\_e2ee.models*), [17](#)