

NKeys



- [Generating NKeys and Configuring the Server](#)
 - [Client Configuration](#)
-

NKeys are a new, highly secure public-key signature system based on [Ed25519](#).

With NKeys the server can verify identities without ever storing or ever seeing private keys. The authentication system works by requiring a connecting client to provide its public key and digitally sign a challenge with its private key. The server generates a random challenge with every connection request, making it immune to playback attacks. The generated signature is validated against the provided public key, thus proving the identity of the client. If the public key is known to the server, authentication succeeds.

“NKey is an excellent replacement for token authentication because a connecting client will have to prove it controls the private key for the authorized public key.

To generate nkeys, you'll need the [nk tool](#).

Generating NKeys and Configuring the Server

To generate a *User* NKEY:

```
nk -gen user -pubout
```

```
SUACSSL3UAHUDXKF5NVUZRF5UHPMWZ6BFD TJ7M6USDXIEDNPPQYYYYCU3VY  
UDXU4RCSJNZ0IQHZNWXHX0RDPRTGNJAHAFRGZNEEJCPQTT2M7NLCNF4
```

The first output line starts with the letter **S** for *Seed*. The second letter, **U** stands for *User*. Seeds are private keys; you should treat them as secrets and guard them with care.

The second line starts with the letter `U` for *User* and is a public key which can be safely shared.

To use nkey authentication, add a user, and set the `nkey` property to the public key of the user you want to authenticate:

```
authorization: {
  users: [
    { nkey: UDXU4RCSJNZ0IQHZNWXHXORDPRTGNJAHAFRGZNEEJCPQTT2M7NLCNF4 }
  ]
}
```

Note that the user section sets the `nkey` property (user/password/token properties are not needed). Add `permission` sections as required.

Client Configuration

Now that you have a user nkey, let's configure a [client](#) to use it for authentication. As an example, here are the connect options for the node client:

```
const NATS = require('nats');
const nkeys = require('ts-nkeys');

const nkey_seed = 'SUACSSL3UAHUDXKFSNVUZRF5UHPMWZ6BFD TJ7M6USDXIEDNPPQYYYCU3VY';
const nc = NATS.connect({
  port: PORT,
  nkey: 'UDXU4RCSJNZ0IQHZNWXHXORDPRTGNJAHAFRGZNEEJCPQTT2M7NLCNF4',
  sigCB: function (nonce) {
    // client loads seed safely from a file
    // or some constant like `nkey_seed` defined in
    // the program
    const sk = nkeys.fromSeed(Buffer.from(nkey_seed));
    return sk.sign(nonce);
  }
});
...
```

The client provides a function that it uses to parse the seed (the private key) and sign the connection challenge.