readme.md for @push.rocks/smartcrypto

easy crypto methods

Install

To install `@push.rocks/smartcrypto`, open a terminal and run the following command:

```
npm install @push.rocks/smartcrypto
```

This will add `@push.rocks/smartcrypto` to your project's dependencies.

Usage

This guide demonstrates how to use `@push.rocks/smartcrypto` effectively by covering its main functionalities. The module offers straightforward methods for handling cryptographic operations. We'll go through generating keys, encrypting and decrypting data, and converting keys to various formats.

Setting up your project

Firstly, ensure you have a TypeScript project configured to use ES Modules. Initialize `@push.rocks/smartcrypto` in your project as shown:

```
import { Smartcrypto } from '@push.rocks/smartcrypto';

const smartCrypto = new Smartcrypto();
```

Generating a Key Pair

Generating a new RSA key pair is a common requirement. Let's see how `@push.rocks/smartcrypto` simplifies this process:

```
(async () => {
  const keyPair = await smartCrypto.createKeyPair();
  console.log('Public Key:', keyPair.publicKey.toPemString());
  console.log('Private Key:', keyPair.privateKey.toPemString());
})();
```

This asynchronous function generates a new RSA key pair and logs the PEM-formatted public and private keys. This is particularly useful for creating keys for secure communication.

Working with Public and Private Keys

You can manipulate public and private keys using their respective classes. For instance, converting a private key to a PEM string for storage or transmission is straightforward:

```
import { PrivateKey } from '@push.rocks/smartcrypto';

// Assuming you already have a privateKey object:
const privateKeyPemString = privateKey.toPemString();
console.log(privateKeyPemString);
```

Similarly, you can create a `PrivateKey` object from a PEM string. This is useful when you retrieve a stored key and need to use it in your application:

```
const privateKeyFromPem = PrivateKey.fromPemString(privateKeyPemString);
```

The same methods apply to public keys with the `PublicKey` class:

```
import { PublicKey } from '@push.rocks/smartcrypto';

const publicKeyPemString = publicKey.toPemString();
console.log(publicKeyPemString);

const publicKeyFromPem = PublicKey.fromPemString(publicKeyPemString);
```

Advanced Usage

Beyond basic key generation and format conversion, `@push.rocks/smartcrypto` leverages `node-forge` for advanced cryptographic operations. You can engage in more complex scenarios such as signing data, verifying signatures, and even using the library for SSH key generation.

```
// Example: Signing a message with RSA private key
const message = 'Hello, SmartCrypto!';
const signedMessage = privateKey.signMessage(message);
console.log('Signed Message:', signedMessage);

// Example: Verifying a signature with the corresponding public key
const isVerified = publicKey.verifyMessage(message, signedMessage);
console.log('Is the message verified?', isVerified);
```

Note: The `.signMessage` and `.verifyMessage` methods are conceptual and may require specific implementation or extension of the base classes provided by `@push.rocks/smartcrypto`.

Conclusion

`@push.rocks/smartcrypto` provides an easy-to-use interface for cryptographic operations, significantly reducing the complexity of managing keys and performing secure data transmission. The examples above showcase a subset of what's possible, encouraging you to explore further and integrate cryptography seamlessly into your Node.js applications.

For more advanced use cases and customization, refer to the node-forge documentation. `@push.rocks/smartcrypto` is designed to work smoothly with `node-forge`, enabling you to extend its functionalities according to your project's needs.

License and Legal Information

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:10:30 UTC by foss.global Team
Updated 2026-03-28 12:17:14 UTC by foss.global Team