Audit Cardano-JS-SDK

MetaData Audit

Audit Author: Vincent Hanquez
When: 21 November 2023
Commit: `e7e2e6f11bc1b815a8e09f9755ffd1e35ac99be2`
Reviewed: `cardano-js-sdk / packages / crypto`

Disclaimer: The author is not familiar with javascript/typescript, and specially in a cryptography context. While due diligence has been applied to grasp the logic and potential issues, there's a chance subtleties in the language's syntax or conventions have been missed.

Possible issues found

1. Invalid base for all number conversions
2. Bug in padding
3. Timing attack doing arithmetic operations

Issue 1

Location: `packages/crypto/src/Bip32/Bip32KeyDerivation.ts` **line** 74, 82, 93
Severity: Surprising but not problematic
Description:

BN initial function is defined as `function init (number, base, endian)`
But all current usage in the crypto code set base=16, this is surprising to the reader as parsing from bytes imply base=256.
“Thankfully” the base parameter seems completely ignored when reading an ‘Object’ according to the code, and tested through empirical use.

Suggestion:

1. Set `base=256` which look correct despite being ignored
2. remove `base=16` and use `base='le'` which gets reassigned by the code to mean base=10 (which is also incorrect, btw) + endian=‘le’.
3. If there’s a way to use directly the initArray constructor which take a base value also (but never actually use it)

Issue 2

Location: packages/crypto/src/Bip32/Bip32KeyDerivation.ts **line** 97
Severity: minor, but look like a potential timing attack

Description:

The padding used is invalid, resulting in padding never added (unless the number is less than half the size, which should happen only extremely rarely)
This code snippet transforms a “32 bytes number” into the hexadecimal string, which should be 64 characters long, In a case of a short number, the padEnd is passed the value 32 characters to pad to, when it should be 64 characters .

```
if (r.length !== 32) {
    r = Buffer.from(r.toString('hex').padEnd(32, '0'), 'hex');
}
```

Suggestion:
Use `padEnd(64, ‘0’)`, or solve issue 3

Issue 3

Location: packages/crypto/src/Bip32/Bip32KeyDerivation.ts **line** 74, 82, 93
Severity: minor, but previous code was safer
Description:

BN usage to do addition and multiplication is not subject to constant time, unless due care has been applied. So for certain classes of number, this creates a potential timing attack oracle that could result in leaking cryptographic key material.

Suggestion:
Do not use BN and instead use operation using the uint8array operation directly, which can be copied from the rust based code at
https://github.com/typed-io/rust-ed25519-bip32/blob/master/src/derivation/v2.rs

Note on use of BN

The library uses the Bn library to do big number addition and multiplications required for derivation. This has not been audited, but as cryptography goes, using generic big numbers library in the context of cryptography, result in most cases in potential timing attack, as the library are usually optimised for speed and conduct optimisation on their limbs especially during multiplications.