

Smart contract libraries

Proof of Machinehood (PoM) libraries enable developers to implement on-chain machine attestation validations for integration with smart contracts. Open-source smart contract libraries as such provide reusable code components that simplify the development process.

Integrate Proof of Machinehood

PoM integration is as straightforward as importing `AttestationVerificationBase.sol` to the smart contract. This is independent of the machine (or device type) that the project supports.

Start by adding this line to the smart contract:

...

```
Copy import{AttestationVerificationBase}from"@automata-network/proof-of-machinehood-
contracts/AttestationVerificationBase.sol";
```

...

Upon importing the Proof of Machinehood libraries, the smart contract can invoke the `verifyAttStmt()` method. In doing so, the expected challenge and attestation data generated by the user's device via the [Web Authentication API](#) is passed on to the function. This returns a boolean value that indicates the validity of the provided attestation.

The attestation format contains two parameters:

- Attestation Object - Includes authenticator data and Attestation Statement. Different devices have pre-defined attestation statements, which is covered in the next section.
- Client data - Stored as JSON
- string in an `ArrayBuffer`
-

The open-source code for Proof of Machinehood smart contract libraries can be found [here](#).

?

Attestation Statements

Verification of Android device attestation

Below is the Attestation Statement from an Android device, verified by this library, along with a detailed explanation of each field.

...

```
Copy structAttStmt{ ISigVerifyLib.Algorithm alg; stringjwtHeader; stringjwtPayload; stringjwtSignature;
ISigVerifyLib.Certificate[] x5c; }
```

...

- `alg`
- : The algorithm used to generate the signature(`jwtSignature`) for the JWT (JSON Web Token).
- `jwtHeader`
- : The header of the JWT obtained from Google's SafetyNet Service. This field contains a certificate chain that can be used to verify the identity of the device.
- `jwtPayload`
- : The payload of the JWT from Google's SafetyNet Service. It includes fields such as `asctsProfileMatch` and `basicIntegrity`, which help in checking the device's integrity.
- `jwtSignature`
- : The signature part of the JWT from Google's SafetyNet Service, which is signed using the first certificate in the `x5c` array.
- `x5c`
- : The certificate chain included in the `jwtHeader`.
- . This field is added to simplify the on-chain implementation process. Technically, it's possible to extract the certificate chain directly from the `jwtHeader`.
- .
-

Refer to the[complete verification procedure](#) for exact details.

Verification of Windows device attestation

Below is the Attestation statement from a Windows device, verified by this library, along with a detailed explanation of each field.

...

```
Copy structAttStmt{ ISigVerifyLib.Algorithm alg; bytessig; ISigVerifyLib.Certificate[] x5c; bytescertInfo; }
```

...

- alg
- : The algorithm used to generate the signaturesig
- .
- sig
- : The signature created using the first certificate inx5c
- . It provides cryptographic proof of various properties of the device and the credential.
- x5c
- : The certificate chain that verifies the identity of the device.
- certInfo
- : This is the data that is signed and represents a[complex structure](#)
- defined by Microsoft.
-

Refer to the[complete verification procedure](#) for exact details.

Verification of YubiKey attestation

Below is the attestation statement from a YubiKey, verified by this library, along with a detailed explanation of each field.

...

```
Copy structAttStmt{ ISigVerifyLib.Algorithm alg; bytessignature; ISigVerifyLib.Certificate[] x5c; }
```

...

- alg
- : The algorithm used to generate the signaturesig
- .
- sig
- : The signature created using the first certificate inx5c
- . It provides cryptographic proof of specific properties of the device and the credential.
- x5c
- : The certificate chain that verifies the identity of the device.
-

Refer to the[complete verification procedure](#) for exact details.

[Previous Proof of Machinehood](#)[Next Attestations on Verax](#) Last updated26 days ago On this page * [Integrate Proof of Machinehood](#) * [Attestation Statements](#) * [Verification of Android device attestation](#) * [Verification of Windows device attestation](#) * [Verification of YubiKey attestation](#)

Was this helpful?