```
"use strict";

let blindSignatures = require("blind-signatures");

let SpyAgency = require("./spyAgency.js").SpyAgency;

function makeDocument(coverName) {
  return The bearer of this signed document, ${coverName}, has full diplomatic immunity.;
}

function blind(msg, n, e) {
  return blindSignatures.blind({
    message: msg,
    N: n,
    E: e,
  });
}

function unblind(blindingFactor, sig, n) {
  return blindSignatures.unblind({
    signed: sig,
    N: n,
    r: blindingFactor,
  });
}
```

```
let agency = new SpyAgency();

// Prepare 10 documents with 10 different cover identities.
let documents = [];
let blindedDocs = [];
let blindingFactors = [];

for (let i = 0; i < 10; i++) {
  let coverName = CoverIdentity${i + 1};
  let doc = makeDocument(coverName);
  documents.push(doc);

  let { blinded, r } = blind(doc, agency.n, agency.e);
  blindedDocs.push(blinded);
  blindingFactors.push(r);
}

agency.signDocument(blindedDocs, (selected, verifyAndSign) => {
  let blindingFactorsForVerification = [];
  let originalDocsForVerification = [];

  // Populate arrays for verification, skipping the selected document
  for (let i = 0; i < 10; i++) {
    if (i === selected) {
```

```
    blindingFactorsForVerification.push(undefined);
    originalDocsForVerification.push(undefined);
  } else {
    blindingFactorsForVerification.push(blindingFactors[i]);
    originalDocsForVerification.push(documents[i]);
  }
}

// Call verifyAndSign function
let blindedSignature = verifyAndSign(
  blindingFactorsForVerification,
  originalDocsForVerification
);

// Unblind the signature for the selected document
let unblindedSignature = unblind(
  blindingFactors[selected],
  blindedSignature,
  agency.n
);

// Validate the signature
let isValid = blindSignatures.verify({
  unblinded: unblindedSignature,
  message: documents[selected],
```

```
    N: agency.n,
    E: agency.e,
  });

  console.log(Document ${selected} signature is valid: ${isValid});
  console.log(Signature: ${unblindedSignature});
});
```