Funded by
the European Union

Escaping from Identity Providers: Protecting Privacy with Verifiable Credentials in Community Solid Server

Ben Macdonald, Ross Horne, Biagio Boi

University of Salerno
bboi@unisa.it

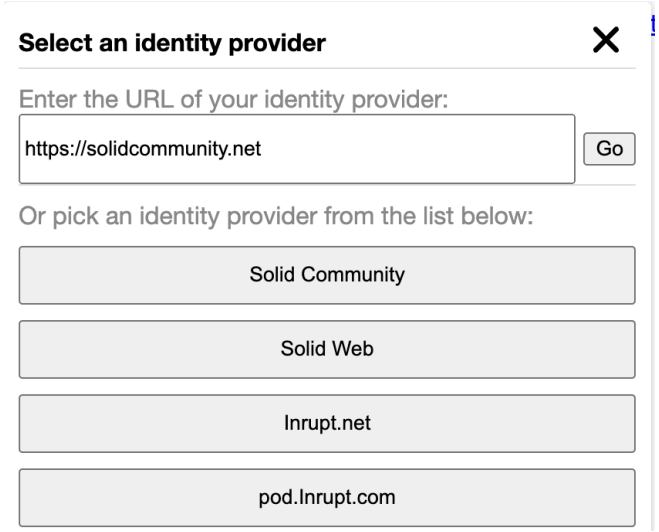
2nd Solid Symposium – SoSy Privacy Session, 2-3 May 2024, Leuven

Introduction

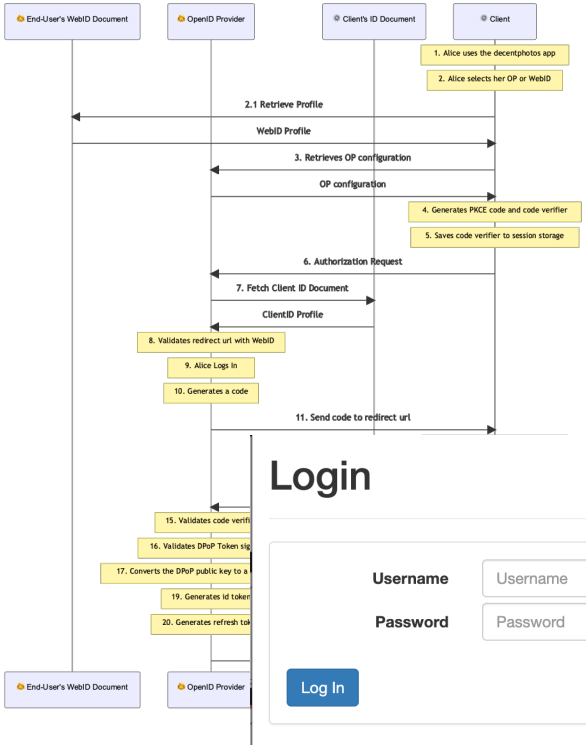
Access to Solid PODs is currently managed using passwords.

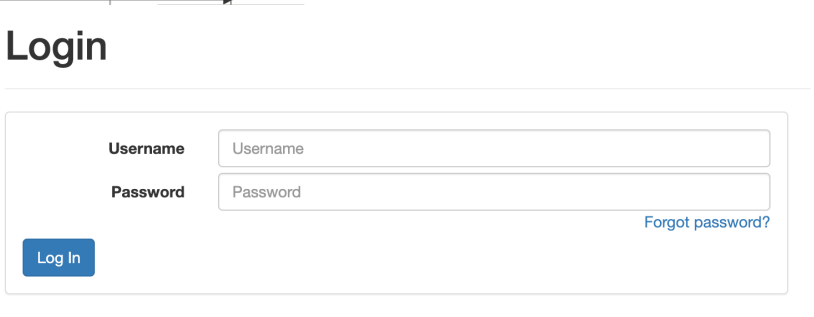
Password-based authentication is the weakest in terms of privacy since credentials are stored on the Service Provider (SP), which is also responsible for service data.

OIDC is a valid alternative, where an Authentication Server (AS) is responsible for managing users identity.



Introduction

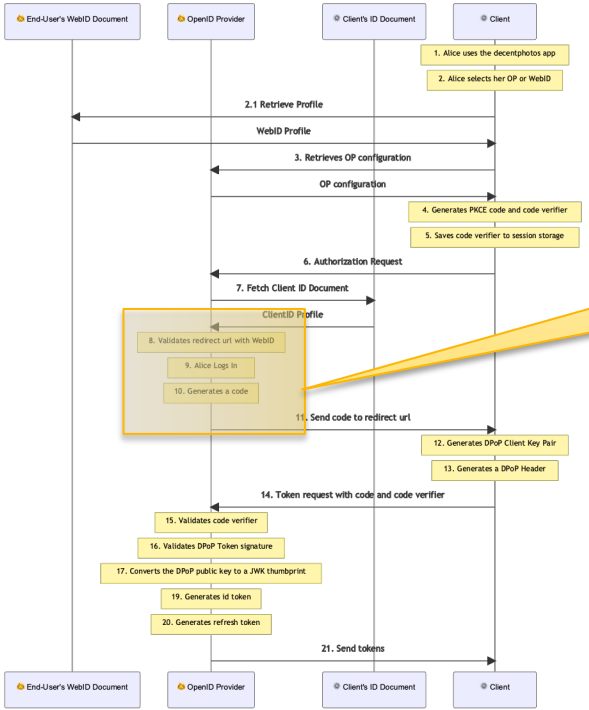




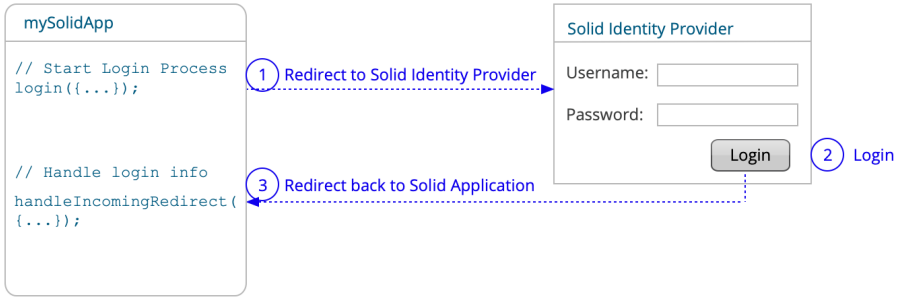
In typical OIDC-based authentication, the AS is contacted by the client to obtain an access token.

The AS checks for the identity through a simple username and password request.

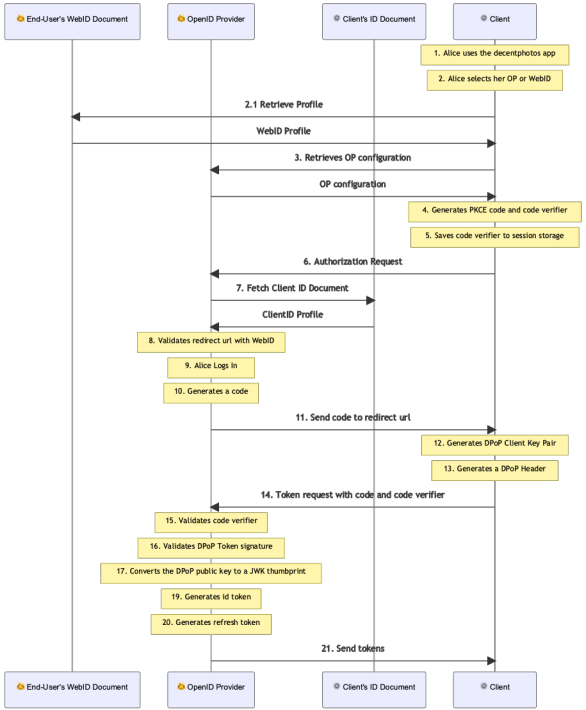
Introduction



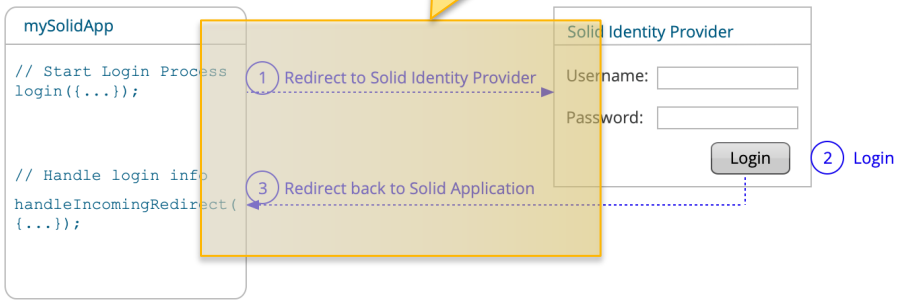
Moreover, the OpenID Provider checks for the client's request URL before releasing the access token.



Introduction



The ID Provider might store information about the Token redirect URL, leading to a violation of the privacy property unlinkability.

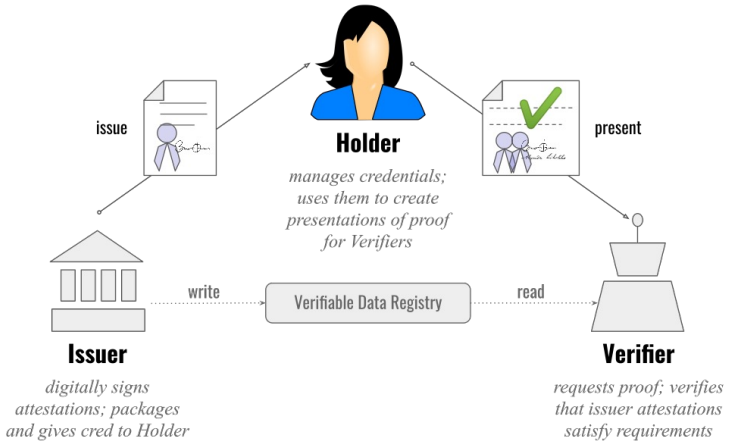


Introduction

To prevent privacy issues, a novel approach based on user-centric authentication has been proposed.

The schema is based on the concept of Verifiable Credentials (VCs). VCs are digitally signed representations of a physical credential.

A trust triangle is defined and a verifiable data registry typically holds the information to check the signature.



Introduction

To prevent privacy issues, a novel approach based on user-centric authentication has been proposed.

The schema is based on the concept of Verifiable Credentials (VCs). VCs are digitally signed representations of a physical credential.

A trust triangle
typically holds

A VC contains one or more attributes, creating the subject-property-value relationship.

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```

Introduction

To prevent privacy issues, a novel approach based on user-centric authentication has been proposed.

The schema is based on the concept of Verifiable Credenti
of a phys

It also defines information on who released the credential, the kind of credential, and the expiration date.

A trust triangle is defined and a verifiable data registry typically holds the information to check the signature.

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```

Introduction

To prevent privacy issues, a novel approach based on user-centric authentication has been proposed.

The schema is based on the concept of Verifiable Credentials (VCs). VCs are digitally signed representations of a physical credential.

A trust triangle is define
typically holds the infor

DIDs are used to identify the subject of a credential.

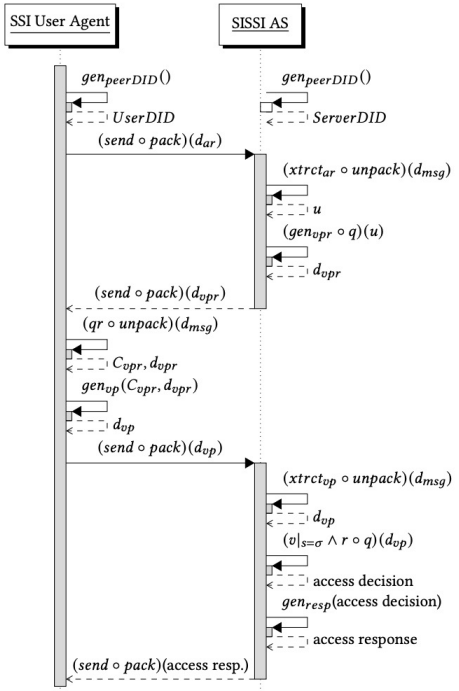
```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```

Introduction

Introduction

An architecture for authentication and authorization approach has been proposed in [1], while a detailed analysis of security properties is described in [2].

Protocol	Property	No.	Relative File Path in Repository	OK	Attack
Plain VCs (PlainVCs/DIDComm/)	Secrecy	1	ssipv.pv#L287	✓	
		2	archive/ssipv_forward_secrecy.pv	✓	
	Agreement	3	ssipv.pv#309	✓	
		4	ssipv_ok_VP_leaked.pv	✓	
		5	ssipv_unforgeable_VC.pv	✓	
		6	ssipv_attack_domain_missing_replay.pv	×	*masquerade as prover*
		7	ssipv_attack_no_nonce_VP_leaked.pv	×	*replay credential*
		8	ssipv_attack_VC_reissued.pv	×	*reissue old credential*
	Unlinkablitiy	9	ssipv_unlinkable.dps	✓	
		10	ssipv_attack_verifier_unlinkablity.dps	×	*verifier tracks prover*



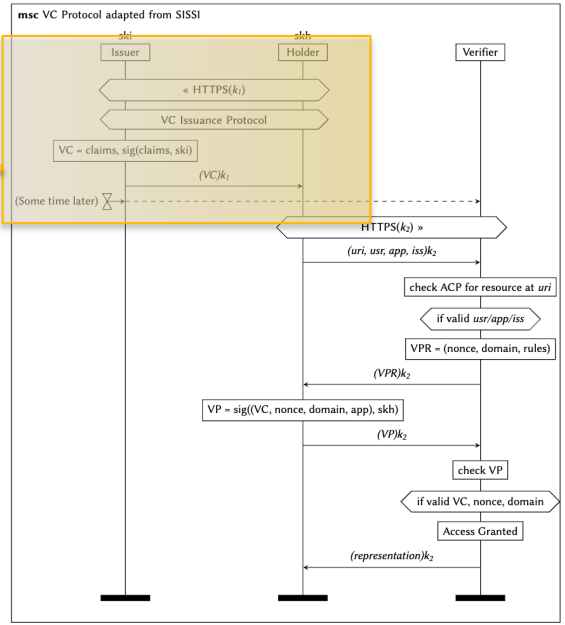
[1] Braun, Christoph H-J., Vasil Papanchev, and Tobias Käfer. "SISSI: an architecture for semantic interoperable self-sovereign identity-based access control on the web." *Proceedings of the ACM Web Conference 2023*. 2023.

[2] Christoph H.-J. Braun, Ross Horne, Tobias Käfer, and Sjouke Mauw. 2024. SSI, from Specifications to Protocol? Formally Verify Security! . In Pro- ceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore. ACM, New York, NY, USA, 12 pages. https://doi.org/ 10.1145/3589334.3645426

Introduction

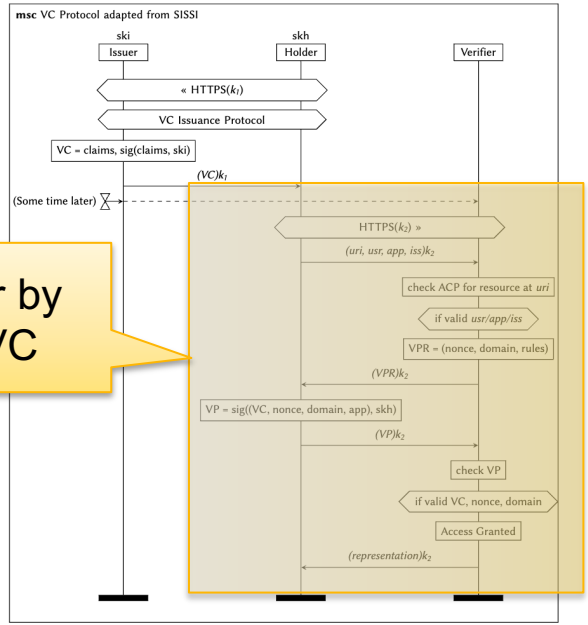
A simple adaption of the protocol defined in [1] might be possible in Community Solid Server by using the server as both Issuer and Verifier.

The Issuer releases the VC containing the WebID, which is stored by the Holder.



Introduction

A simple adaption of the protocol defined in [1] might be possible in Community Solid Server by using the server as both Issuer and Verifier.

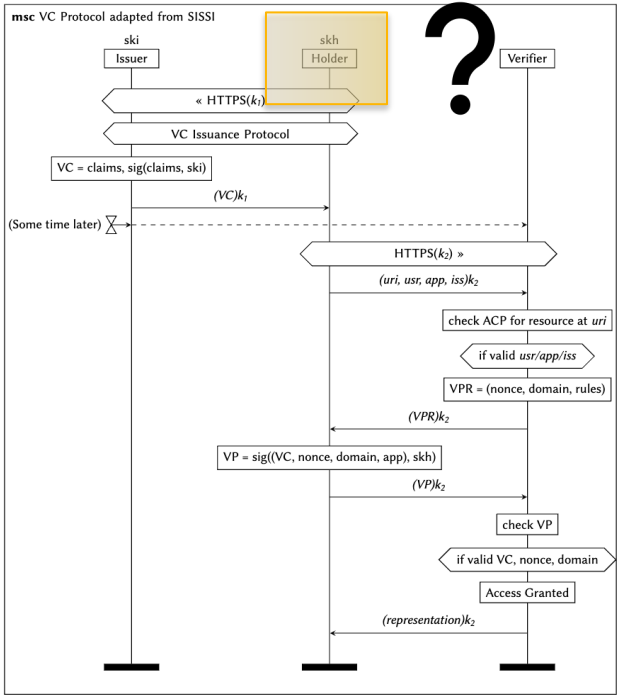


The Verifier authenticates the Holder by checking the signatures applied on VC

Motivation

Anyway, in such a protocol, the Holder represents both the user and the application that interacts with the AS.

A secure flow requires that the application must be fully trusted by the user to create and use VPs without his explicit approval each time.



Objective

A simple adaption of the protocol defined in [1] might be possible in Community Solid Server by using the server as both an Issuer and Verifier.

Challenges

Anyway, the role of the Holder must be adequately designed, in order to separate the real User from the Application.

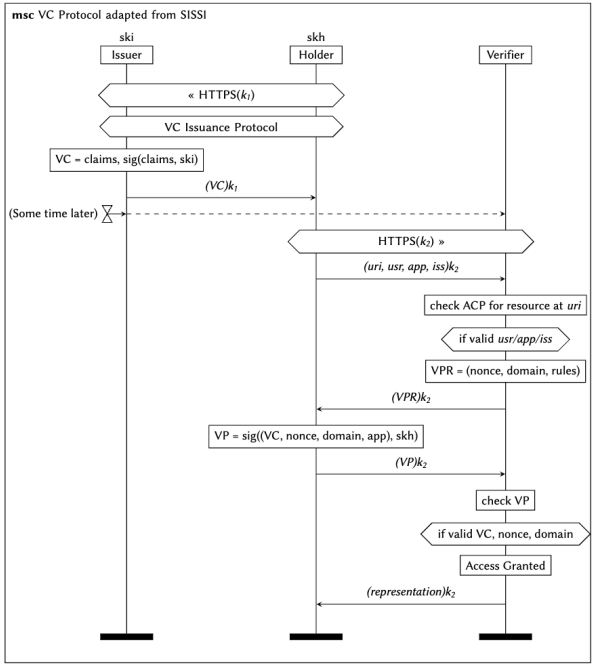
Key Contribution

The key contribution of our research is to create a protocol able to:

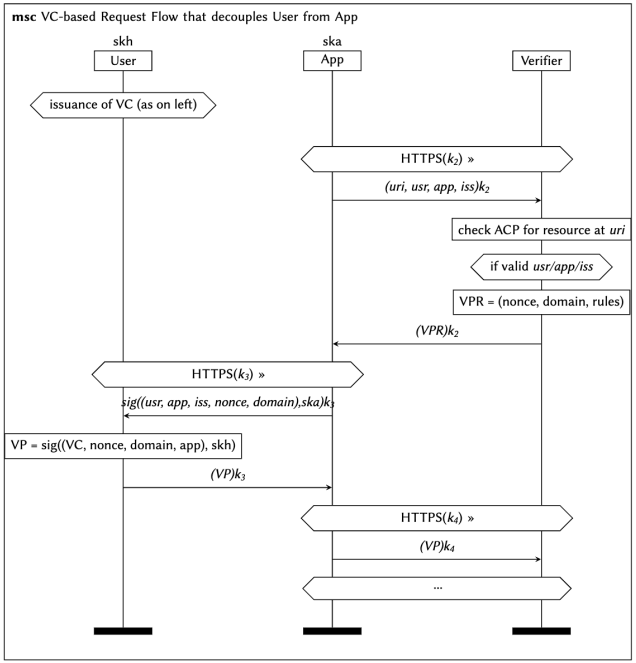
1. Guarantee users access to resources using VCs
2. Respect the separation between applications and users

Proposed Protocol

Previous (no decoupling)



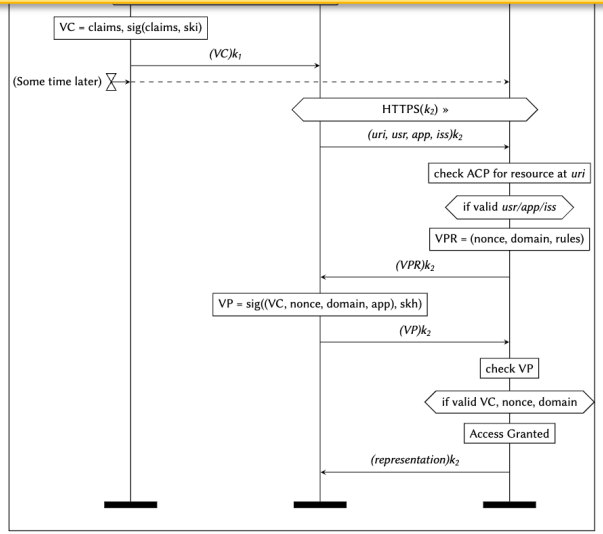
Proposed (decoupled)



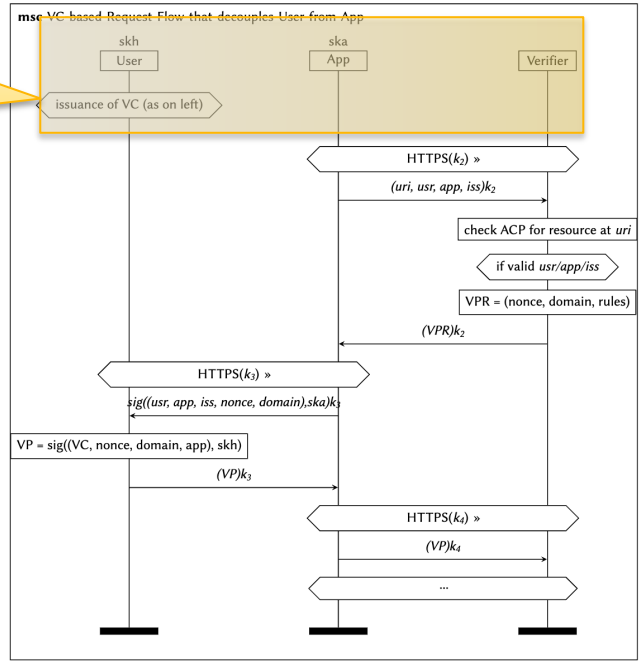
Proposed Protocol

Previous (no decoupling)

We have a complete separation between the user and the application

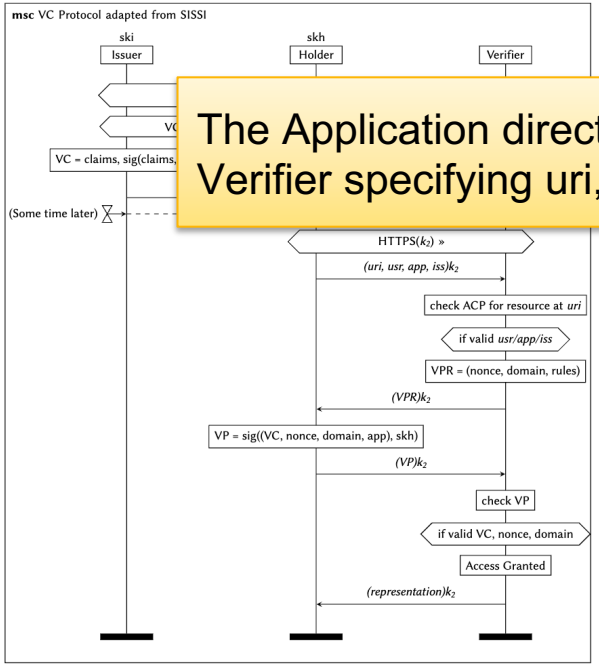


Proposed (decoupled)

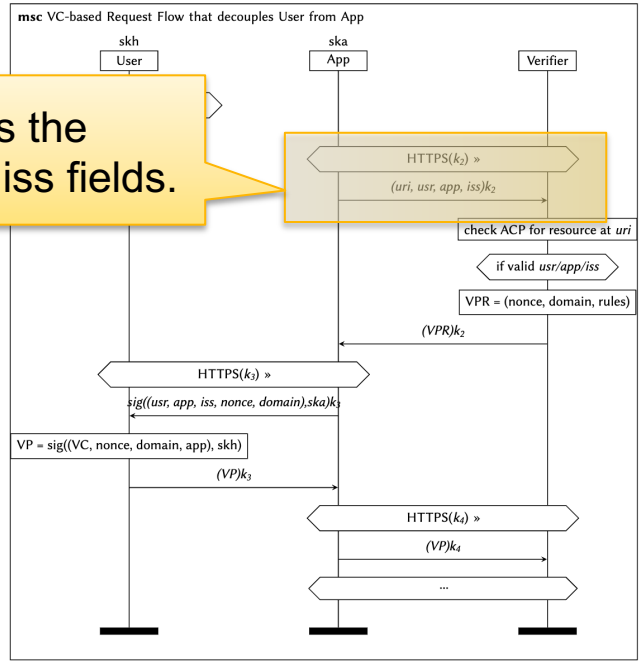


Proposed Protocol

Previous (no decoupling)



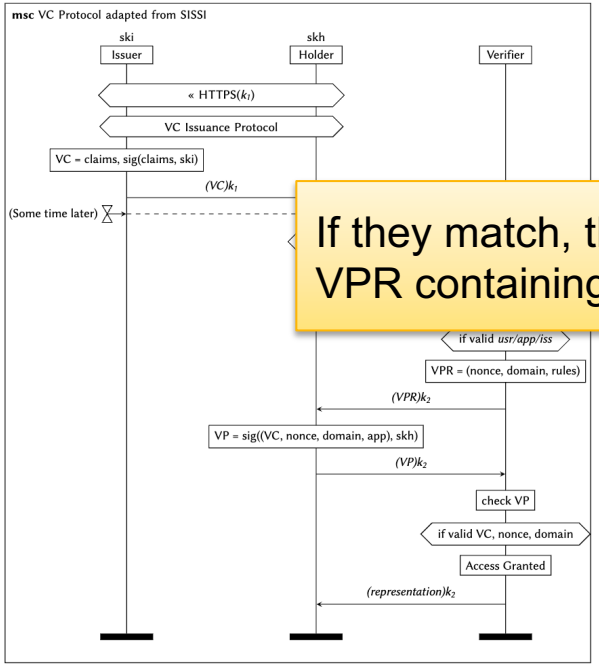
Proposed (decoupled)



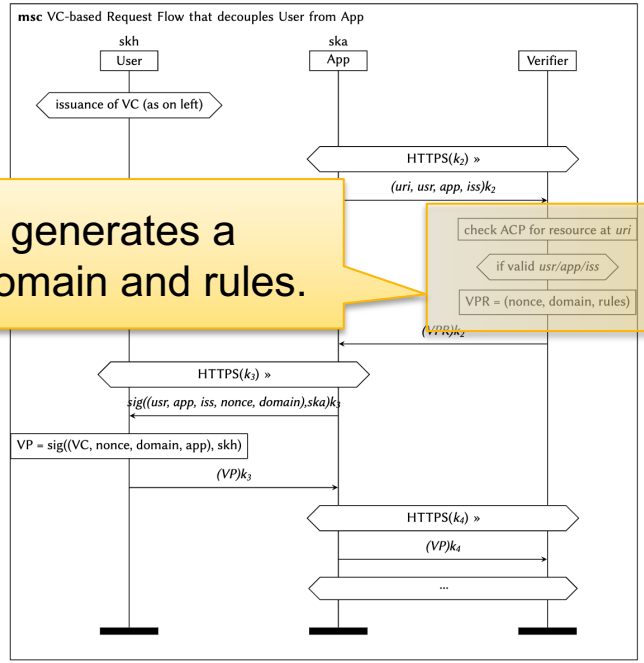
The Application directly contacts the Verifier specifying uri, usr, app, iss fields.

Proposed Protocol

Previous (no decoupling)



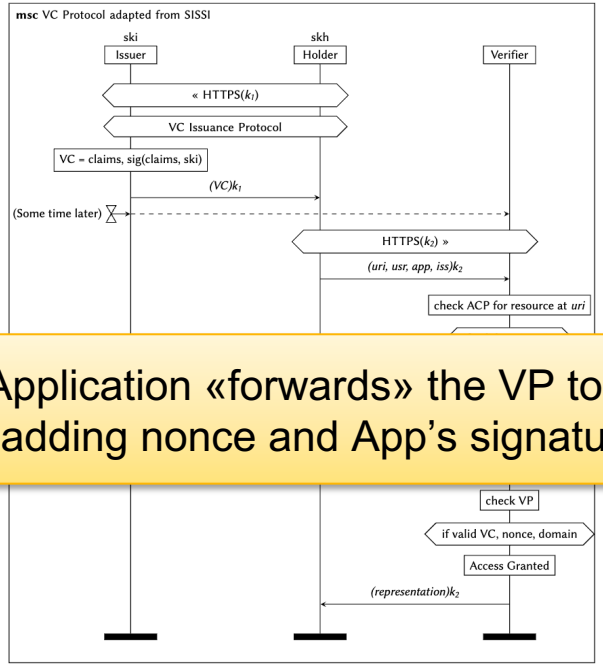
Proposed (decoupled)



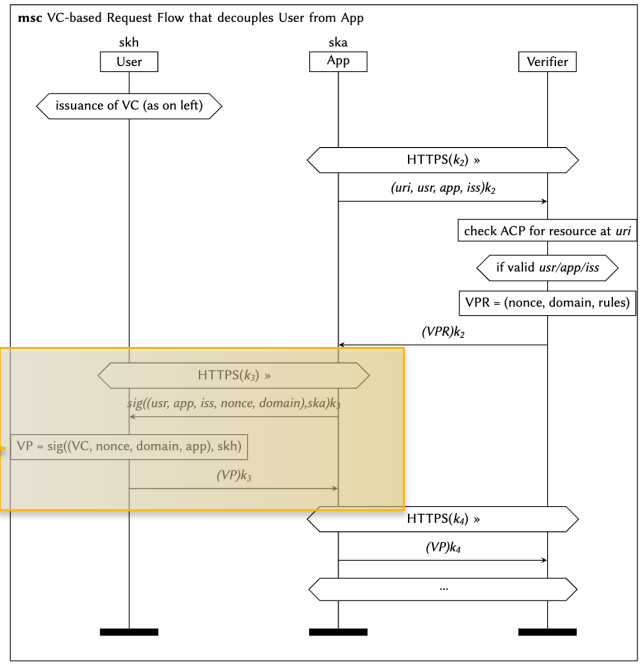
If they match, the Verifier generates a VPR containing nonce, domain and rules.

Proposed Protocol

Previous (no decoupling)



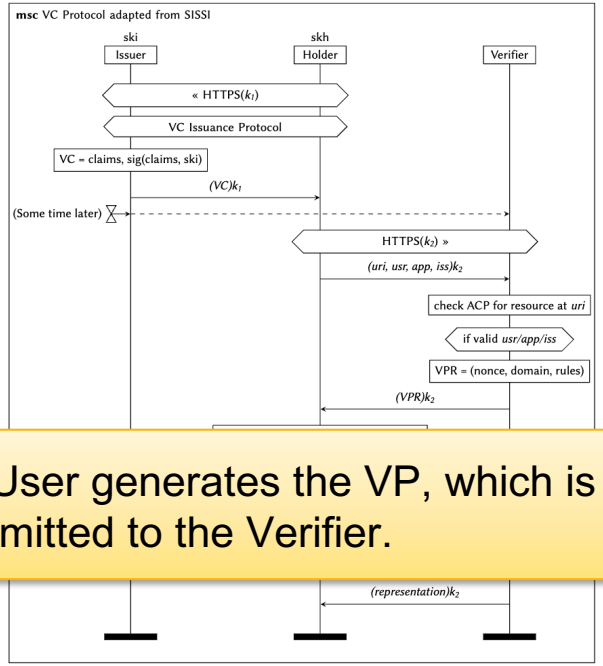
Proposed (decoupled)



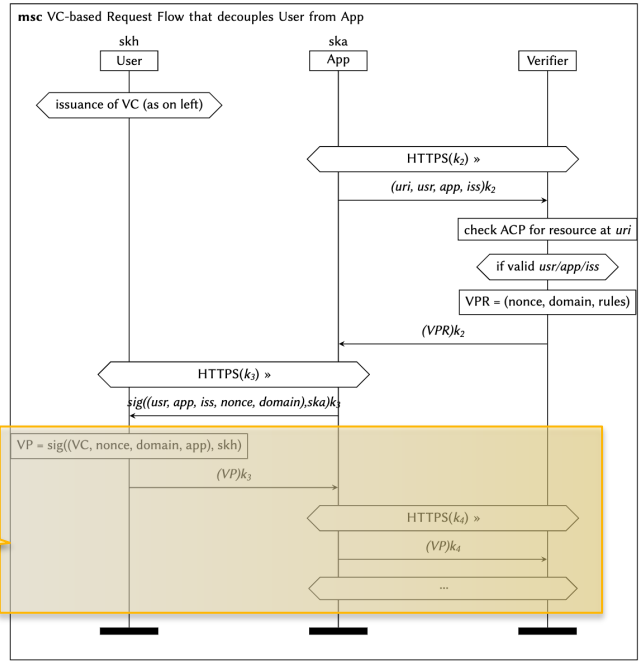
The Application «forwards» the VP to the User adding nonce and App's signature.

Proposed Protocol

Previous (no decoupling)



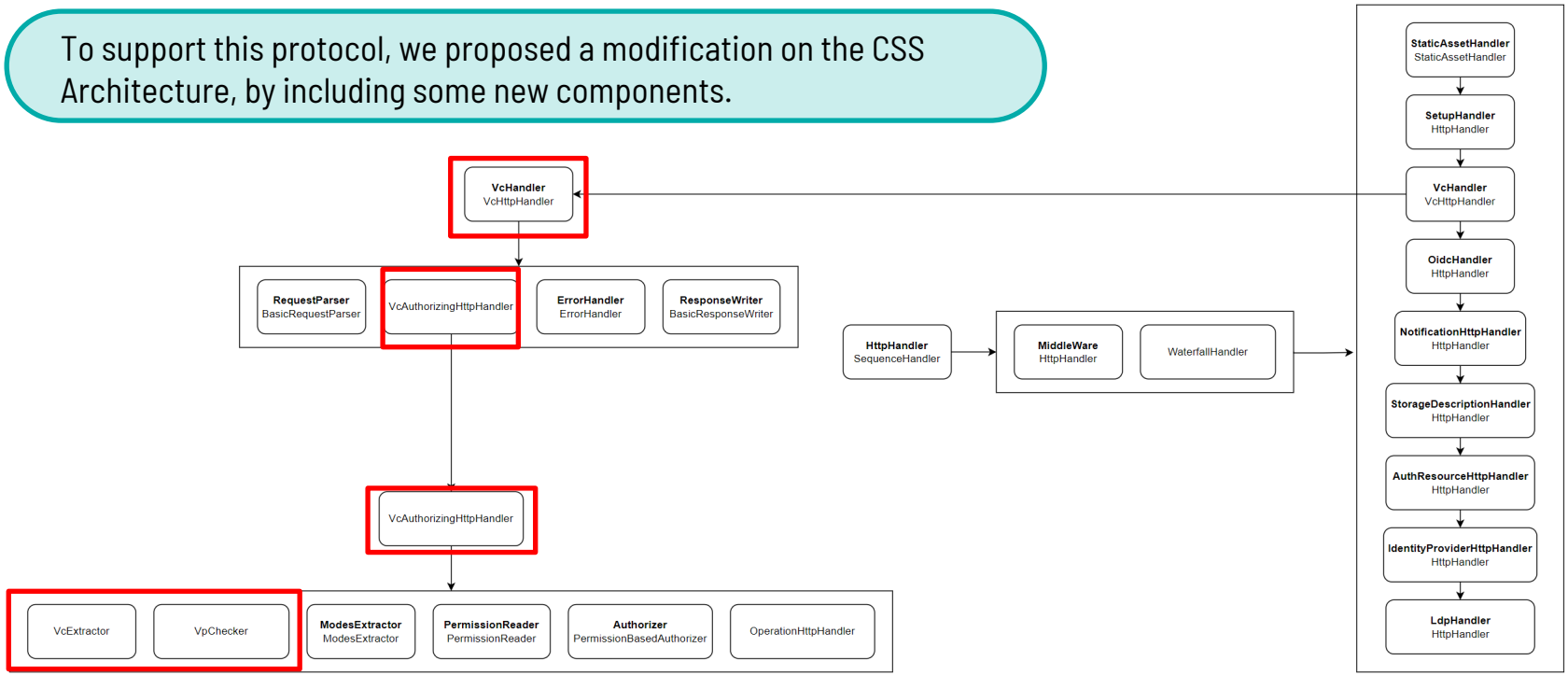
Proposed (decoupled)



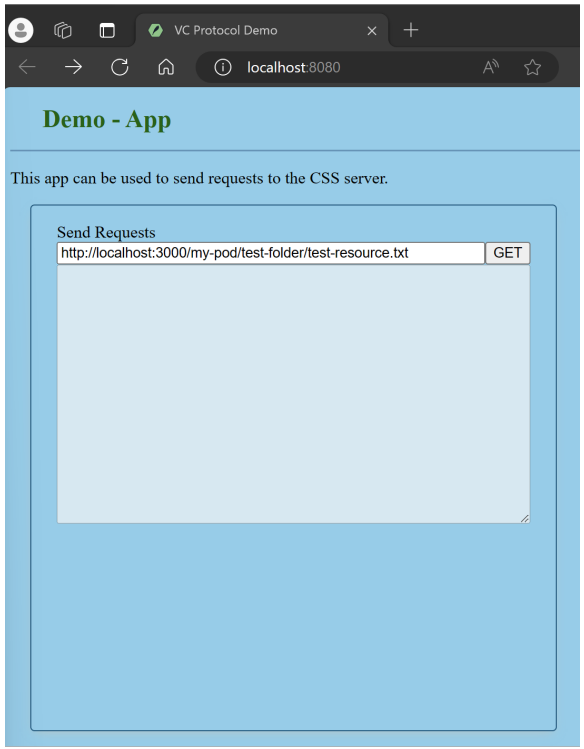
The User generates the VP, which is then transmitted to the Verifier.

System Architecture

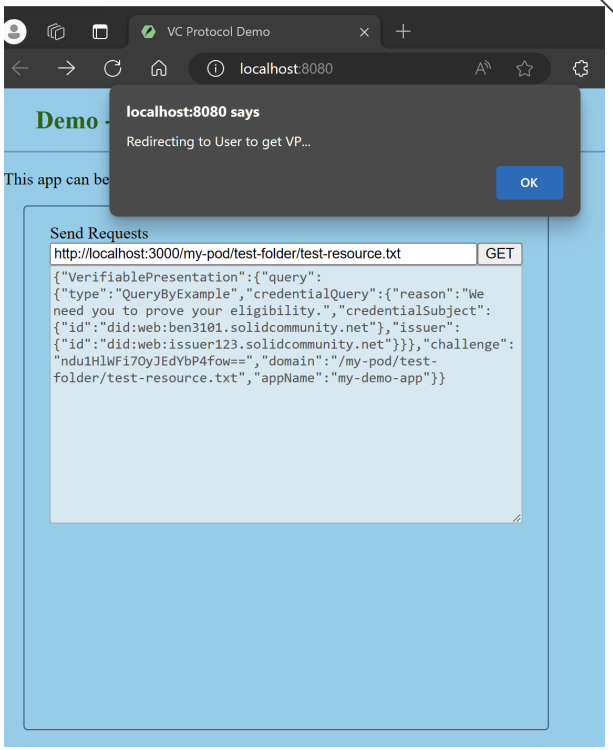
To support this protocol, we proposed a modification on the CSS Architecture, by including some new components.



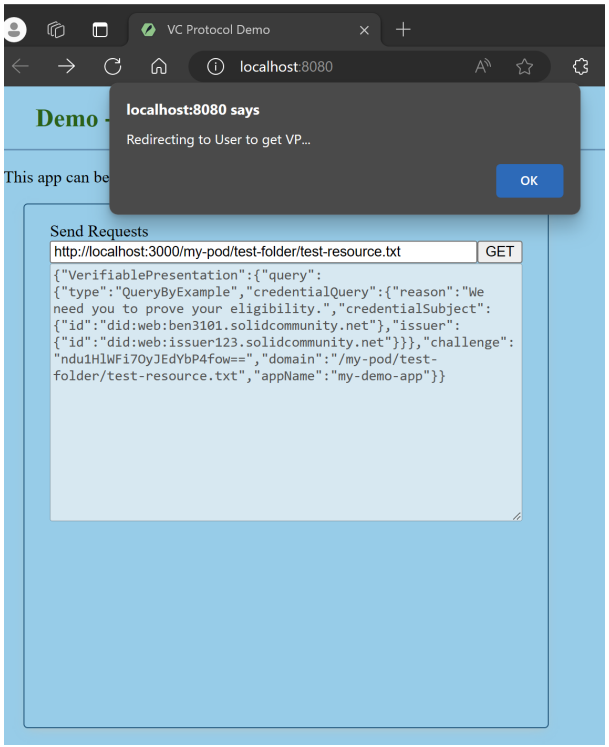
Demo



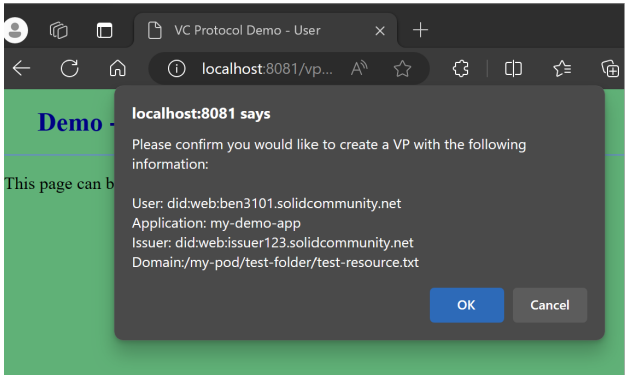
The Application sends a request to the CSS.



Demo

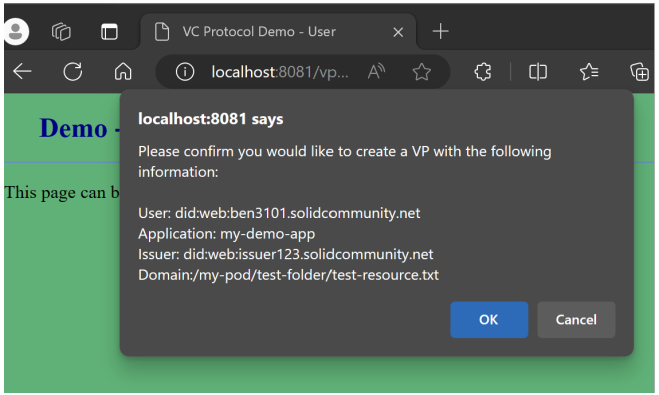


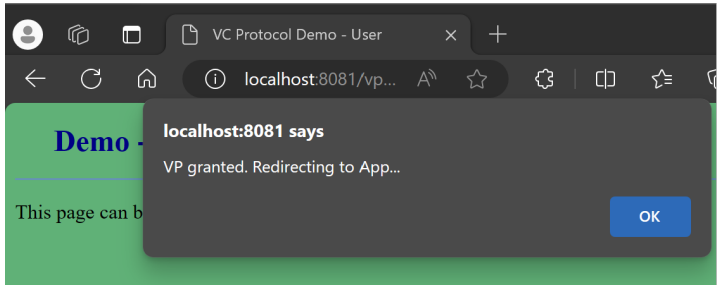
A VP request is sent to the User



Demo

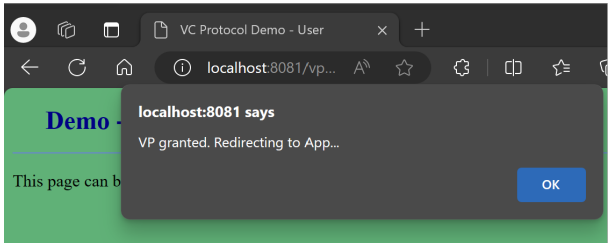
Once the User read the field of the request, then he can proceed with VP generation.

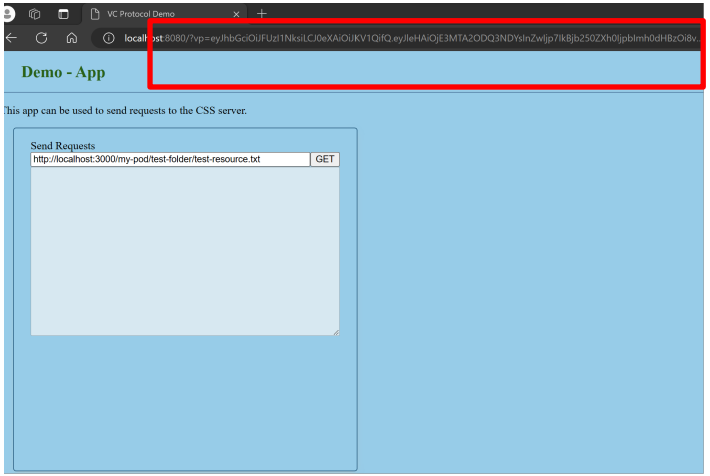




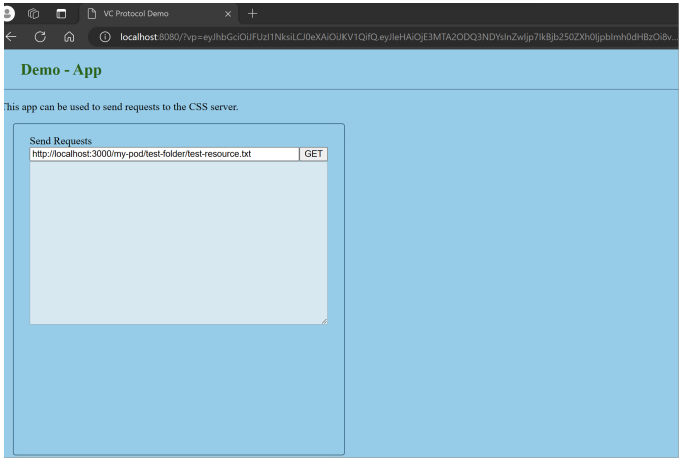
Demo

The VP is included in the request to the App.

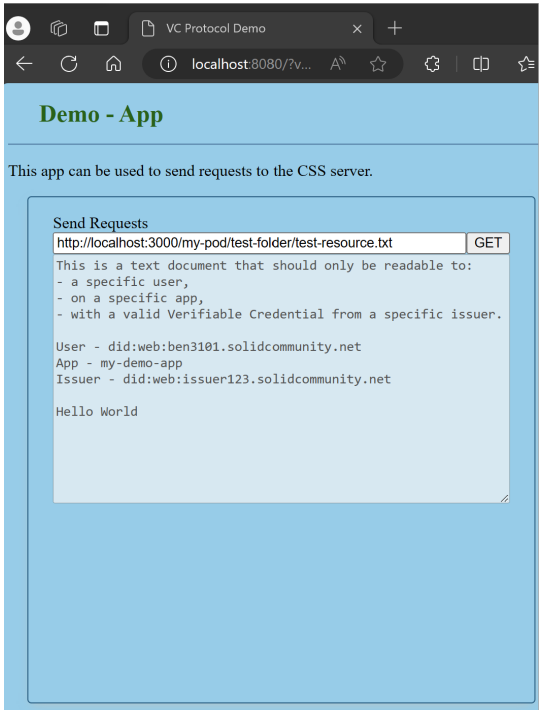




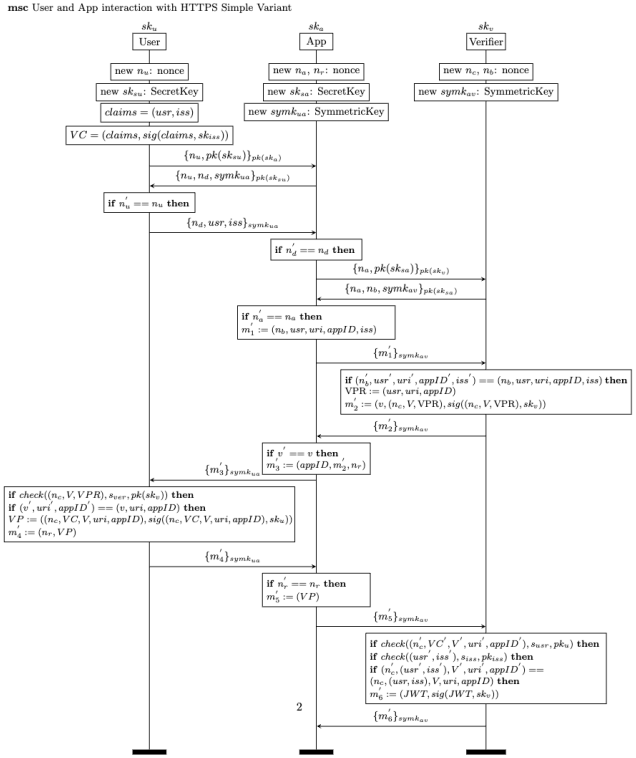
Demo



The app can now get the content from the CSS.



Formal Verification



To evaluate the security of the proposed protocol, we also formally verified the entire protocol using ProVerif, ensuring security properties.

```
Verification summary:

Query not attacker(rule_fromVerifier[]) is true.

Query not attacker(vp_fromVerifier[]) is true.

Query not attacker(access_token_fromVerifier[]) is true.

Query inj-event(auth_UserCompletesProtocol(m_handshakeReq,m_handshakeResp,m_complete,m_vpr,m_vp)) ==> inj-ev
ent(auth_AppSendsLastMessageToUser(m_handshakeReq,m_handshakeResp,m_complete,m_80,m_81,m_82,m_83,m_vpr)) &&
inj-event(verifierInConeOfUser(m_80,m_81,m_82,m_83)) is true.

Query inj-event(auth_VerifierCompletesProtocol(m_handshake_ver,m_handshake_resp_ver,m_uri,m_vpr,m_vp,m_acces
stoken)) ==> inj-event(auth_AppSendsLastMessageToVerifier(m_80,m_81,m_2_bis_8,m_handshake_ver,m_handshake_re
sp_ver,m_uri,m_vpr,m_82,m_83,m_vp)) && inj-event(auth_UserSendsLastMessageToApp(m_80,m_81,m_2_bis_8,m_82,m_8
3)) is true.

Query inj-event(auth_AppCompletesProtocol(m_80,m_81,m_2_bis_8,m_handshakeReq,m_handshakeResp,m_uri,m_rule,m_
82,m_83,m_vp,m_accesstoken)) ==> inj-event(auth_VerifierSendsLastMessageToApp(m_handshakeReq,m_handshakeResp
,m_uri,m_rule,m_vp,m_accesstoken)) && inj-event(auth_UserSendsLastMessageToApp(m_80,m_81,m_2_bis_8,m_82,m_83
)) is true.
```

Conclusion

We proposed a novel authentication schema for CSS, able to decouple user and application.

We also proposed a modification to the CSS architecture to support our protocol.

Formally Security of the protocol has been proven.

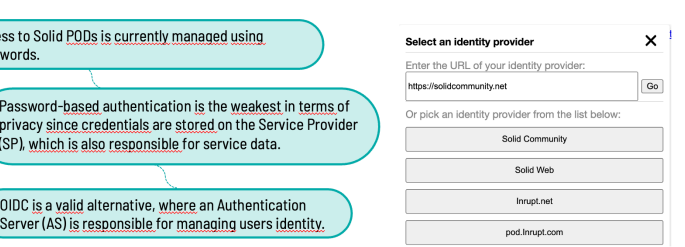
In future work, we aim to enhance the entire protocol by including non-repudiation as an additional property.

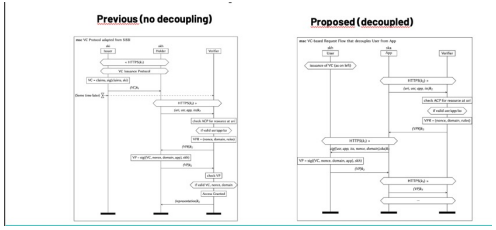
Thank you!

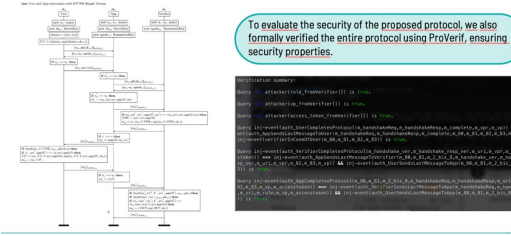
Questions?

Ben Macdonald, Ross Horne, **Biagio Boi**

Escaping from Identity Providers: Protecting Privacy with Verifiable Credentials in Community Solid Server







cost
EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Funded by the European Union