**Businesss logic**
An Identity Verification Provider (IdVP) is an entity whose business model is to ease access and management of Personally Identifiable Information (PII), owned by some Identity Holder (IdH), to their customers which we'll call Identity Verifiers (IdV).
In order to verify the authenticity of these PIIs, the IdVP has to communicate with an Identity Provider (IdP), which is usually a governamental database that has authority over the IdH.
The value of this business model comes from the centralization of many different IdPs under the same API and the possibility to amortize the cost of IdP accesses across customers.
Some examples are the companies Jumio, Onfido and IDnow.

**Problem**
The IdVP should not have access to the data of the IdH since the only concern of the described business transaction is to verify the authenticity of the PIIs through the IdP API.

**Constraints**
- The IdVP is the only one that has access to the IdP API
- The IdP can only be interfaced with a simple HTTPS REST API and is not willing to install or modify any software

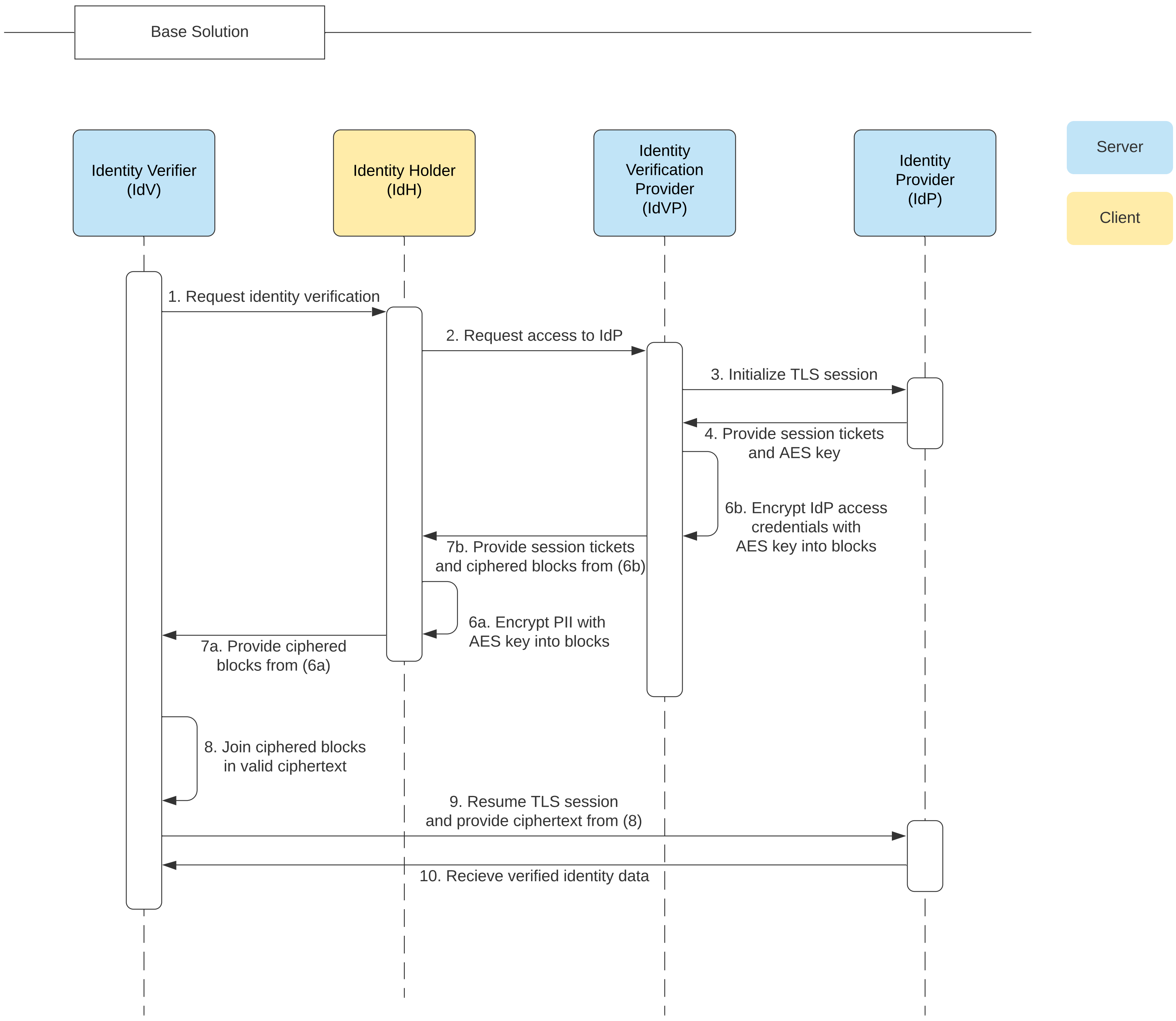
**Proposed solution**
A solution would be for the parties to never completely share all the assets needed for the transaction, and restructure the interactions around the new ownership scheme.

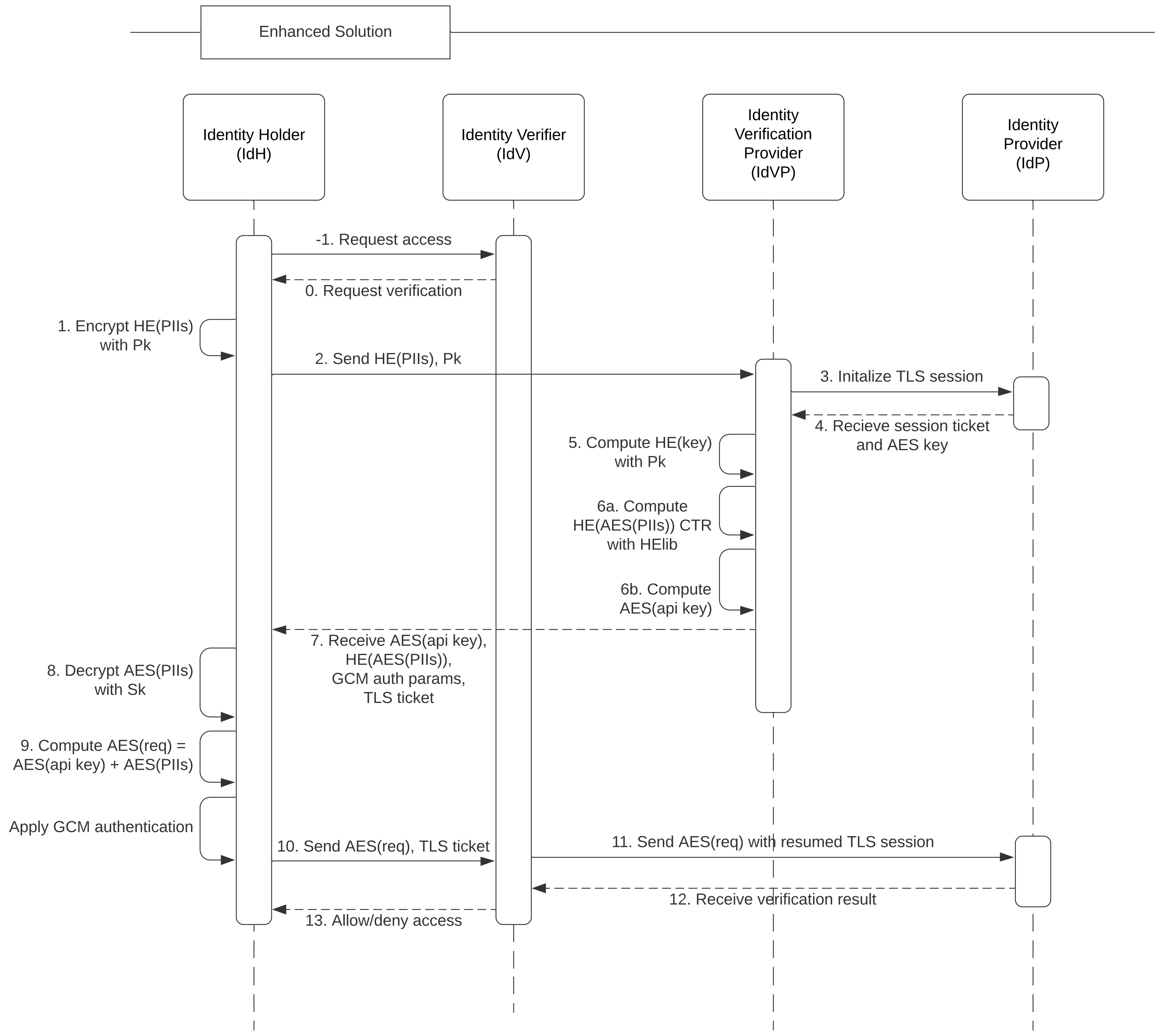
**Weaknesses**
- IdH and IdVP can each try to MITM the connection at step (9) and decrypt each other's information
- The IdH could collaborate with IdV and provide him the AES key to decrypt the IdP credentials

**Mitigations**
- It's not in the interest of any party to degrade the service for their respective customers
- The connection at step (9) is hard to MITM as the IdV infrastructure is hardly ever hosted on machines that communicate via wireless, and the attack would require physically tampering with the wired links.

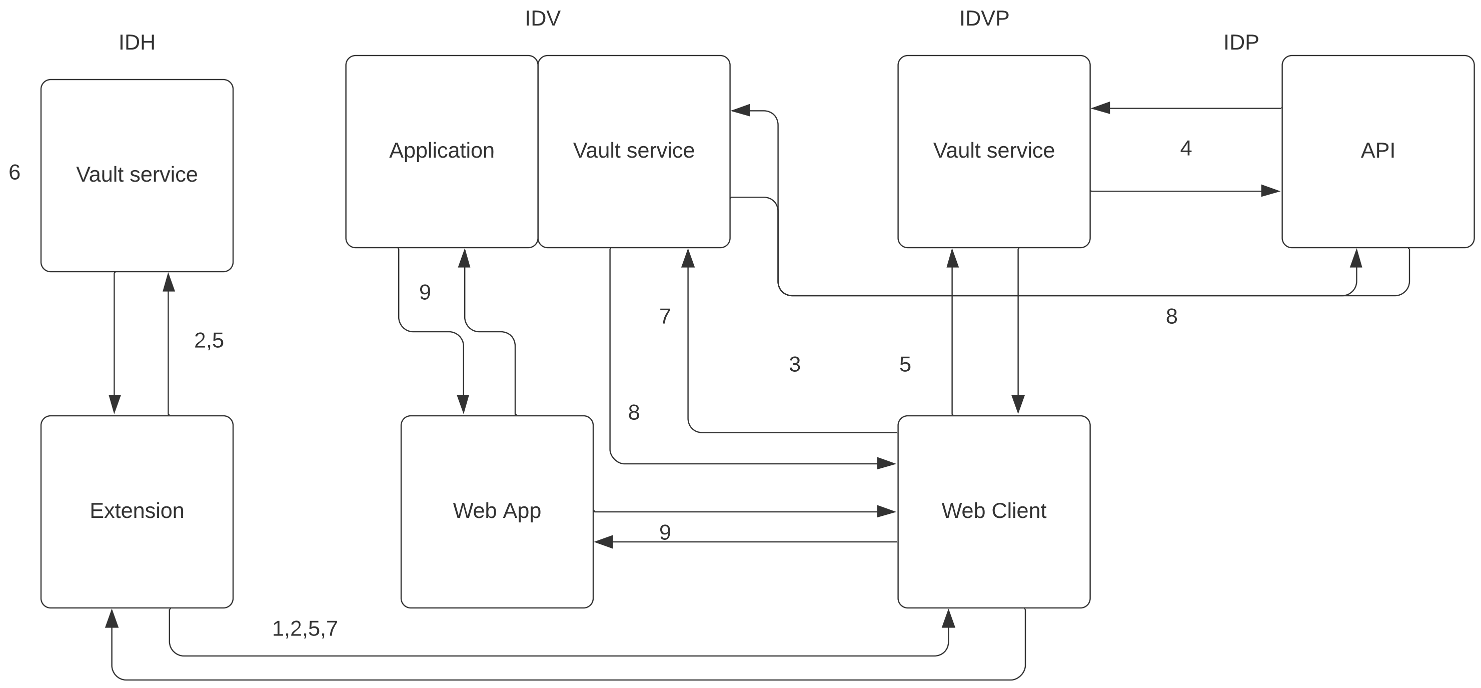
**Assets**

| IdV | IdH | IdVP |
|---|---|---|
| Encrypted message body | PIIs | IdP API credentials |
| TLS session ticket | AES key | AES key |


Base Solution
Identity Verifier (IdV)
Identity Holder (IdH)
Identity Verification Provider (IdVP)
Identity Provider (IdP)
Server
Client
1. Request identity verification
2. Request access to IdP
3. Initialize TLS session
4. Provide session tickets and AES key
6b. Encrypt IdP access credentials with AES key into blocks
7b. Provide session tickets and ciphered blocks from (6b)
6a. Encrypt PII with AES key into blocks
7a. Provide ciphered blocks from (6a)
8. Join ciphered blocks in valid ciphertext
9. Resume TLS session and provide ciphertext from (8)
10. Recieve verified identity data


Enhanced Solution
Identity Holder (IdH)
Identity Verifier (IdV)
Identity Verification Provider (IdVP)
Identity Provider (IdP)
-1. Request access
0. Request verification
1. Encrypt HE(PIIs) with Pk
2. Send HE(PIIs), Pk
3. Initalize TLS session
4. Recieve session ticket and AES key
5. Compute HE(key) with Pk
6a. Compute HE(AES(PIIs)) CTR with HElib
6b. Compute AES(api key)
7. Receive AES(api key), HE(AES(PIIs)), GCM auth params, TLS ticket
8. Decrypt AES(PIIs) with Sk
9. Compute AES(req) = AES(api key) + AES(PIIs)
Apply GCM authentication
10. Send AES(req), TLS ticket
11. Send AES(req) with resumed TLS session
12. Receive verification result
13. Allow/deny access

IdH wants to access a part of the product of IdV that requires authentication via identity verification. IdV has outsourced verification to IdVP.
1. IdH opens the IdV application and is presented with an integration of the IDvP client that requires him to authenticate.
2. IdH has the HE vault service + browser extension running on his machine. The IDvP client calls the extensions which calls the vault service to generate a Pk, Sk pair and encrypt the PIIs with the Pk.
3. HE(PIIs), Pk are sent to IdVP with some identifier generated by the IDvP client
4. IDvP makes a dummy HTTPS request to IdP to obtain TLS ticket and AES key
5. IDvP computes HE(AES(PIIs)) and sends back everything to IdH
6. IdH vault service can now decrypt HE(AES(PIIs)) and build a correct AES(body)
7. IdH vault service talks back to the IDvP client with the AES(body) and TLS ticket, which sends everything to the IdV hosted vaulted service
8. Now IdV vault service can send everything to IdP and get a result, and communicate it back to the IdVP client
9. The web clinet can now communicate to the IdV app if the user is authenticated or not


IDH
IDV
IDVP
IDP
Vault service
Application
Vault service
Vault service
API
Extension
Web App
Web Client
6
2,5
9
7
8
3
5
4
8
9
1,2,5,7