Multifactor Fusion in a Verifiable Credential

Francisco Corella
fcorella@pomcor.com

Revised on April 21, 2024
after presentation at IIW 38 on April 18

pomcor
pomcor.com

Cryptographic authentication

- Definition: authentication by proof of possession of a secret, such as a private key
- Provides protection against man-in-the-middle attacks
  - … If done right
- But it is only one-factor authentication
  - No protection against capture of the secret
- **Strong security requires authentication with 2 or 3 of the following factors: knowledge, possession, inherence**

Three methods of providing multifactor cryptographic authentication

1. Use the factors independently of each other in separate authentication procedures
   - Possession factor provided by presenting a cryptographic credential
   - Knowledge and/or inherence factors provided by authenticating with a password and/or a biometric
2. Use a knowledge factor and/or an inherence factor to unlock the use of a cryptographic credential (as in FIDO)
3. Combine knowledge and/or inherence factors with a possession factor into a fusion credential and use them together in a single authentication procedure with the credential

Security strength of the methods

- Method 1 inherits the drawbacks of the knowledge and/or inherence factors
  - If a password is used, it is vulnerable to phishing, reuse, brute force guessing and dictionary attacks after a backend breach
- Method 2 removes some of these drawbacks by not submitting the knowledge and/or inherence factors to the backend, but some drawbacks remain
  - If a password is used, it is not vulnerable to a backend reach, but it is still vulnerable to ~~phishing~~, reuse and guessing
- A fusion credential provides the strongest security

Examples of fusion credentials

- Camenisch et al., 2013. Fusion of a biometric factor with a zero-knowledge cryptographic factor
- Gunasinghe and Bertino, 2015. Another fusion of biometrics with zero-knowledge technology
- Pomcor, 2016. Fusion of a selective disclosure certificate with a password and/or a biometric
- Pomcor, 2023. Cross-browser authentication with a fusion a selective disclosure certificate with a password and/or a biometric.

Question

- Can a verifiable credential be fused with knowledge and/or inherence factors?

Prerequisite question

- How is a verifiable credential used for authentication?

Verifiable credentials vs. traditional cryptographic credentials

- A traditional credential is a cryptographic construct intended for authentication and presentation of claims over a communication channel, e.g.:
  - A FIDO credential is a key pair that a browser uses for return visit authentication to a relying party over the internet
  - A TLS server certificate (resp., client certificate) is an X.509 public key certificate that a TLS server (resp., TLS client) uses to authenticate over a TCP connection
  - An mDL credential is a selective disclosure public key certificate that a mobile device uses to authenticate and demonstrate driving privileges over a proximity channel such as BLE, NFC or Wi-Fi.

Verifiable credentials vs. traditional cryptographic credentials, continued

- OTOH, a verifiable credential is a signed RDF graph that certifies a state of affairs in the world, e.g.:
  - The kinds of plastics that make up a particular plastic recyclate
  - The supply-chain components in a product shipped across a border
  - The emission reduction methodology, verification status, and ownership of a carbon credit
  - That an adult is a guardian of a child, both being identified by DIDs.
  - That Pat is an alumna of Example University

Verifiable credentials were not originally intended for authentication

Credential presentation

- Authentication with a traditional credential is achieved by means of a **presentation protocol** performed over a **communication channel**, e.g.:
  - In authentication with a FIDO key pair, the presentation protocol consists of the relying party sending a challenge and the browser responding with a signature on the challenge computed with the private key
  - In authentication with a TLS server or client certificate, the presentation protocol is the TLS handshake
- The VC community has morphed the cryptographic concept of a **presentation protocol** into the concept of a **verifiable presentation**, but:
  - Since the VC specification is a data model, a verifiable presentation is data, it's not a protocol
  - The purpose of a verifiable presentation is to establish authorship of data, rather than to authenticate a party at the other end of a communication channel: **"A verifiable presentation is a tamper-evident presentation encoded in such a way that authorship of the data can be trusted after a process of cryptographic verification"**

Verifiable presentations are not intended for authentication either

Rephrasing the prerequisite question

- **Can a verifiable credential actually be used for authentication?**

Answer

- **Yes, by imitating the way in which a traditional X.509 public key certificate is used for authentication**

- An X.509 certificate is **functionally similar** to a VC

  CA signature's <---> Issuer's signature

  Attributes <---> Claims

  Public key <---> DID of the subject

- An X.509 certificate is **functionally almost identical** to a VC if the DID of the subject is a **did:key** decentralized identifier

  CA signature's <---> Issuer's signature

  Attributes <---> Claims

  Public key <---> Public key encoded into method-specific identifier of the DID of the subject

- The only difference is that a did:key identifier is used in multiple credentials, but that does not make a difference for authentication.
  - (The public key in an X.509 certificate is actually a **decentralized identifier** that is usually not persisted but could be.)

Authentication with an X.509 certificate

- Wallet has:
  - An X.509 certificate
  - The private key associated with the public key in the certificate
- Verifier sends:
  - A random challenge
- Wallet sends:
  - The certificate
  - A signature on the challenge computed with the private key
- Verifier verifies:
  - The signature of the issuer in the certificate
  - The identity and trustworthiness of the issuer (e.g. as evidenced by a certificate chain)
  - The signature on the challenge

Authentication with a verifiable credential

- Wallet has:
  - A verifiable credential that has a DID as subject identifier, shared with other credentials, of the form:

    did:key:<encoding of a public key>
  - The private key associated with the public key encoded into the shared DID, stored in the secure enclave of the wallet
- Verifier sends a random challenge
- Wallet sends:
  - The verifiable credential
  - A signature on the challenge computed with the private key
- Verifier verifies:
  - The signature of the issuer in the verifiable credential
  - The identity and trustworthiness of the issuer
  - The signature on the challenge

Now that we have answered the prerequisite question, back to the original question

- **Can a verifiable credential be fused with knowledge and/or inherence factors?**

Answer

- **Yes, by fusing the factors into a DID used as subject identifier**

Fusion with knowledge factor: (1) DID creation

- Wallet generates a key pair and a secret salt, and stores them in the secure enclave of the wallet
- User chooses a password
- Wallet computes:
  - A hash of the password and the secret salt (the "salted password")
  - A hash of the public key and the salted password (the "joint hash")
- Wallet constructs the DID identifier as the concatenation:

  did:fusion:1<encoding of the joint hash>

  where "1" indicates the kind of fusion (like "numalgo" in did:peer)
- Wallet deletes the password and the salted password
- Wallet will use the DID as subject identifier in multiple VCs

Remark

- The joint hash provides some amount of protection against post-quantum attacks or weakness of the cryptosystem of the DID key pair by only exposing the public key to parties to which the verifiable credential is submitted for authentication

Fusion with knowledge factor: (2) Authentication

- Wallet has:
  - A verifiable credential with a shared DID of type did:fusion:1 as subject identifier
  - The private key and secret salt associated with the shared DID
- User submits a password
- Wallet computes the salted password
- Verifier sends a random challenge
- Wallet sends:
  - The verifiable credential
  - The public key associated with the shared DID
  - A signature on the challenge computed with the associated private key
  - The salted password
- Verifier:
  - Verifies the credential and the signature on the challenge
  - Computes the joint hash of the public key and the salted password
  - Verifies that the joint hash is encoded into the DID

Fusion with inherence factor

- Problem:
  - We want to use a biometric sample instead of a password
  - But genuine biometric samples have small variations, and a cryptographic hash of a slightly different sample will be completely different
- Solution:
  - A technology known as **revocable biometrics** can derive an immutable **biometric key** from genuine but varying biometric samples, **using error correction coding**

Revocable biometrics: (1) Biometric key and helper data generation

- User supplies an enrollment biometric sample
- Wallet generates a random **biometric key**
- Wallet adds redundancy to the biometric key to produce an error-correction **codeword**
- Wallet encodes the enrollment sample into a biometric **enrollment code**
- Wallet uses **bitwise x-or**, written ⊕ below, to combine the codeword and the enrollment code into **helper data**:

  helper_data = enrollment_code ⊕ codeword

Revocable biometrics: (2) Biometric key recovery for authentication

- User supplies an authentication biometric sample, which wallet encodes into an **authentication code**
- Wallet uses combines the authentication code and the helper data:

  authentication_code ⊕ helper_data =
  authentication_code ⊕ (enrollment_code ⊕ codeword) =
  (authentication_code ⊕ enrollment_code) ⊕ codeword =
  bits_that_differ ⊕ codeword =
  codeword_with_bit_differences
- If the authentication sample is genuine, the error correction algorithm may be able to revert the codeword with differences to the original codeword:

  codeword = error_correction_function(codeword_with_bit_differences)
- The biometric key can then be recovered from the original codeword:

  biometric_key = redundancy_removal_function(codeword)

Fusion with inherence factor: (1) DID creation

- Wallet generates a key pair and a secret salt, and stores them in its secure enclave
- User supplies an enrollment biometric sample
- Wallet derives a biometric code from the sample, generates a random biometric key, computes helper data from the biometric code and the biometric key, and stores the helper data in the secure enclave
- Wallet computes:
  - A hash of the biometric key and the secret salt (the "salted biometric key")
  - A hash of the public key and the salted biometric key (the "joint hash")
- Wallet constructs the DID identifier as the concatenation:

  did:fusion:2<encoding of the joint hash>

  where "2" indicates the kind of fusion (like "numalgo" in did:peer)
- Wallet deletes the biometric sample, the biometric code, the biometric key, and the salted biometric key
- Wallet will use the DID as subject identifier in multiple VCs

Fusion with inherence factor: (2) Authentication

- Wallet has:
  - A verifiable credential with a shared DID of type did:fusion:2 as subject identifier
  - The private key, secret salt, and helper data associated with the shared DID
- User submits an authentication biometric sample
- Wallet derives the biometric code from the biometric sample and computes the biometric key from the biometric code and the helper data
- Wallet computes the salted biometric key
- Verifier sends a random challenge
- Wallet sends:
  - The verifiable credential
  - The public key associated with the shared DID
  - A signature on the challenge computed with the associated private key
  - The salted biometric key
- Verifier:
  - Verifies the credential and the signature on the challenge
  - Computes the joint hash of the public key and the salted biometric key
  - Verifies that the joint hash is encoded into the DID

Fusion with knowledge and inherence factors: (1) DID creation

- Wallet generates a key pair and a secret salt, and stores them in its secure enclave
- User supplies an enrollment biometric sample
- Wallet derives a biometric code from the sample, generates a random biometric key, computes helper data from the biometric code and the biometric key, and stores the helper data in the secure enclave
- Wallet computes:
  - A hash of the password, the biometric key, and the secret salt (the "salted inputs")
  - A hash of the public key and the salted inputs (the "joint hash")
- Wallet constructs the DID identifier as the concatenation:

  did:fusion:3<encoding of the joint hash>

  where "3" indicates the kind of fusion (like "numalgo" in did:peer)
- Wallet deletes the password, the biometric sample, the biometric code, the biometric key, and the salted inputs
- Wallet will use the DID as subject identifier in multiple VCs

Fusion with knowledge and inherence factors: (2) Authentication

- Wallet has:
  - A verifiable credential with a shared did:fusion:3 DID as subject identifier
  - The private key, secret salt, and helper data associated with the shared DID
- User submits an authentication biometric sample
- Wallet derives the biometric code from the biometric sample and computes the biometric key from the biometric code and the helper data
- Wallet computes the salted inputs
- Verifier sends a random challenge
- Wallet sends:
  - The verifiable credential
  - The public key associated with the shared DID
  - A signature on the challenge computed with the associated private key
  - The salted inputs
- Verifier:
  - Verifies the credential and the signature on the challenge
  - Computes the joint hash of the public key and the salted inputs
  - Verifies that the joint hash is encoded into the DID