ETH zürich

Department of Computer Science

Early Formal Analysis OID4VP over BLE

Felix Linker
PhD Student, ETH Zurich
July 25, 2023

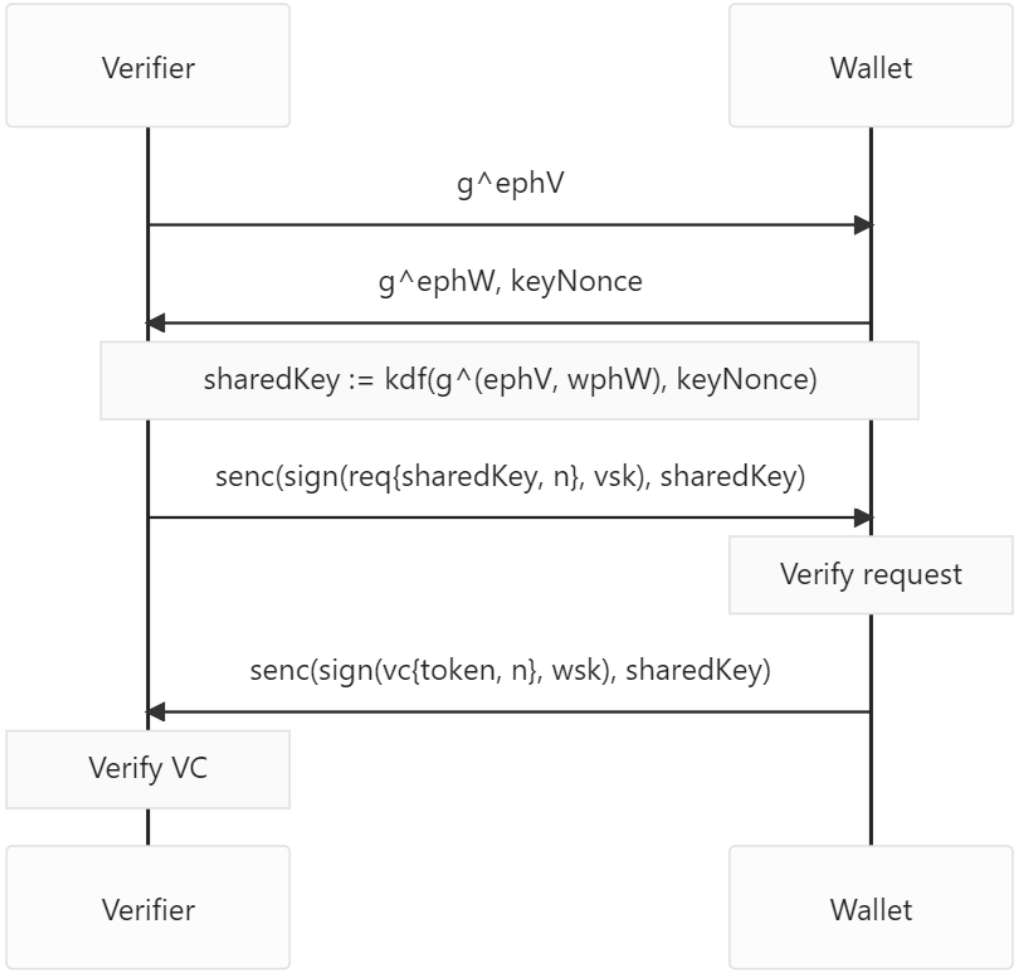
Overview

- How did I model the protocol?
- What did I find?
- What are the conclusions?

Context

- I used the protocol verification tool https://tamarin-prover.github.io/
- Constraint solver in the symbolic model

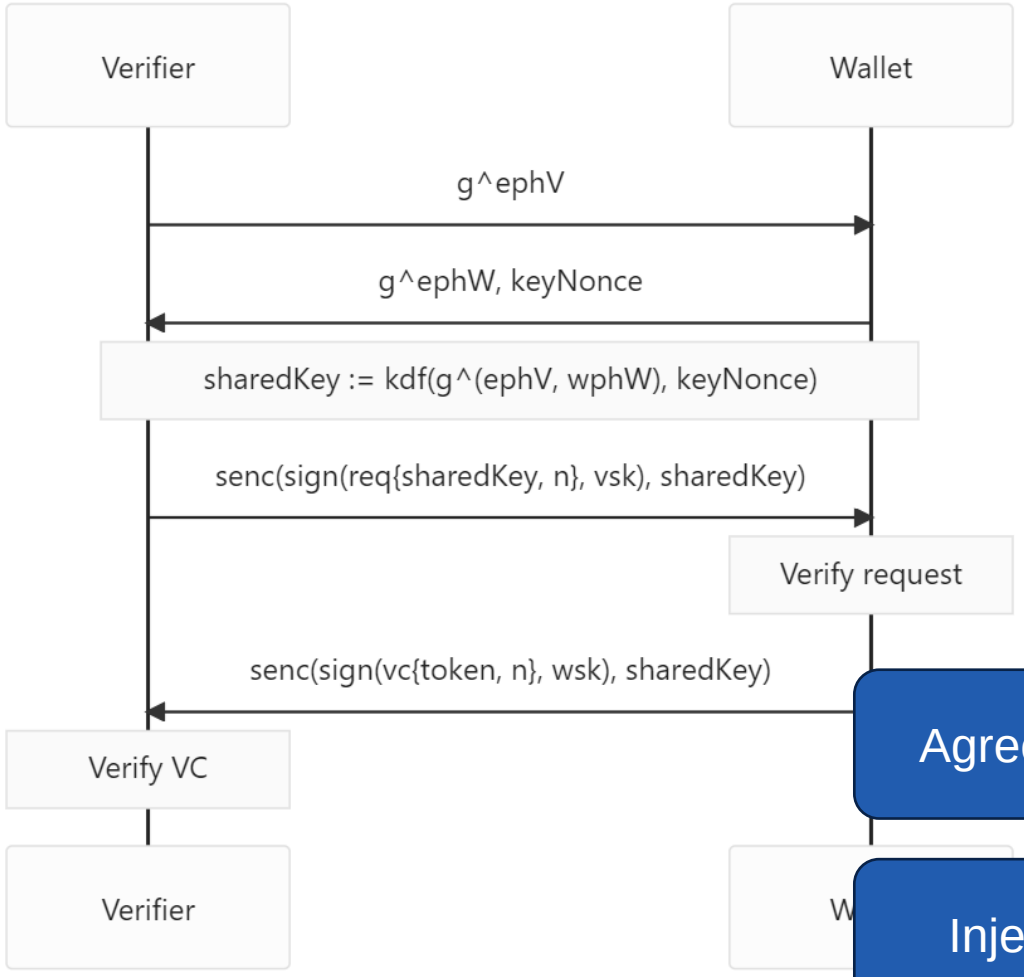
Formal Model



Assumptions

- Long-term wallet/verifier keys do not leak
- Verifier/wallet generate new ephemeral keys with every session (how?)
- Token within VC is an opaque value (modeled like a secret key)
- Modeled BLE channel implicitly using AEAD
- Didn't model EC Diffie-Helman explicitly

Properties



Secrecy

- The adversary cannot learn “token”

Injective agreement

- Whenever…

  …verifier v_1 accepts a VC with token t (using shared key k_1 and nonce n_1)

  …wallet sends VC with token t to verifier v_2 (using shared key k_2 and nonce n_2)

  en…

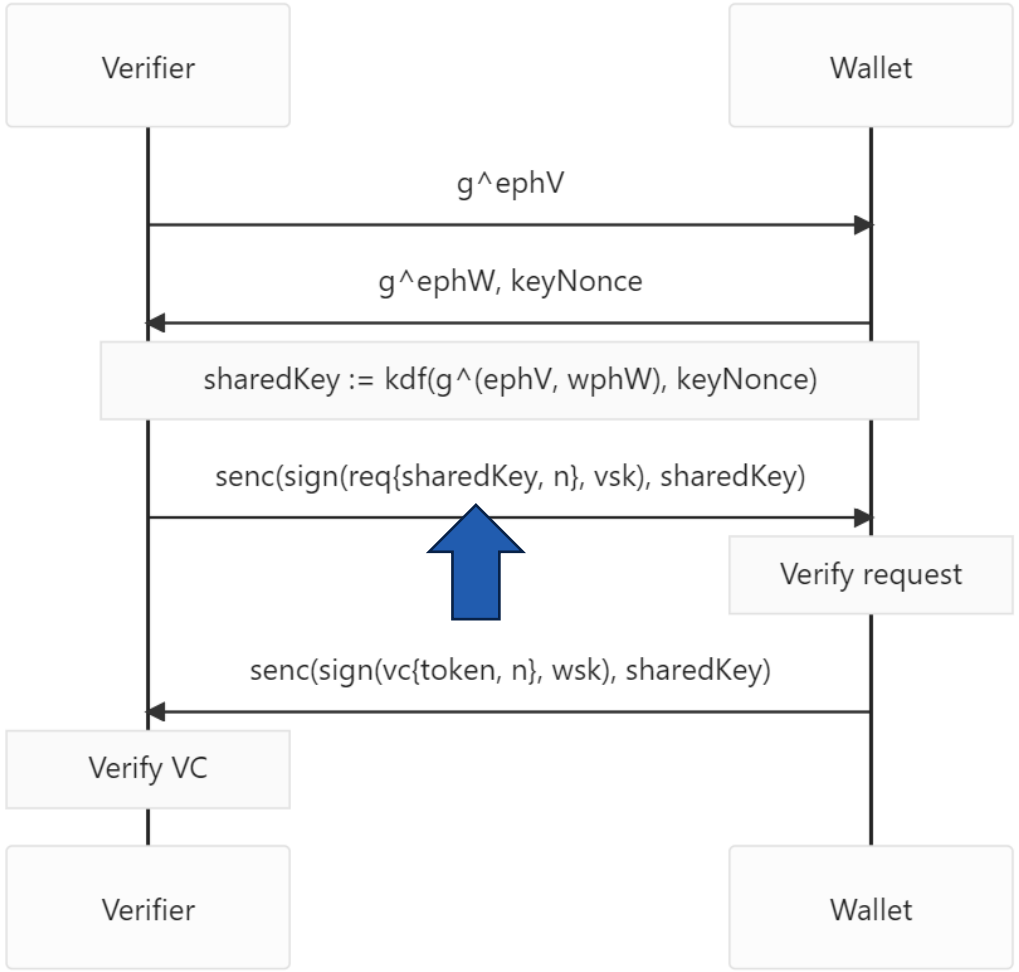
  …$v_1 = v_2$ and $k_1 = k_2$ and $n_1 = n_2$

  …and there is no other event where the same token is accepted

Agreement

Injective

Findings



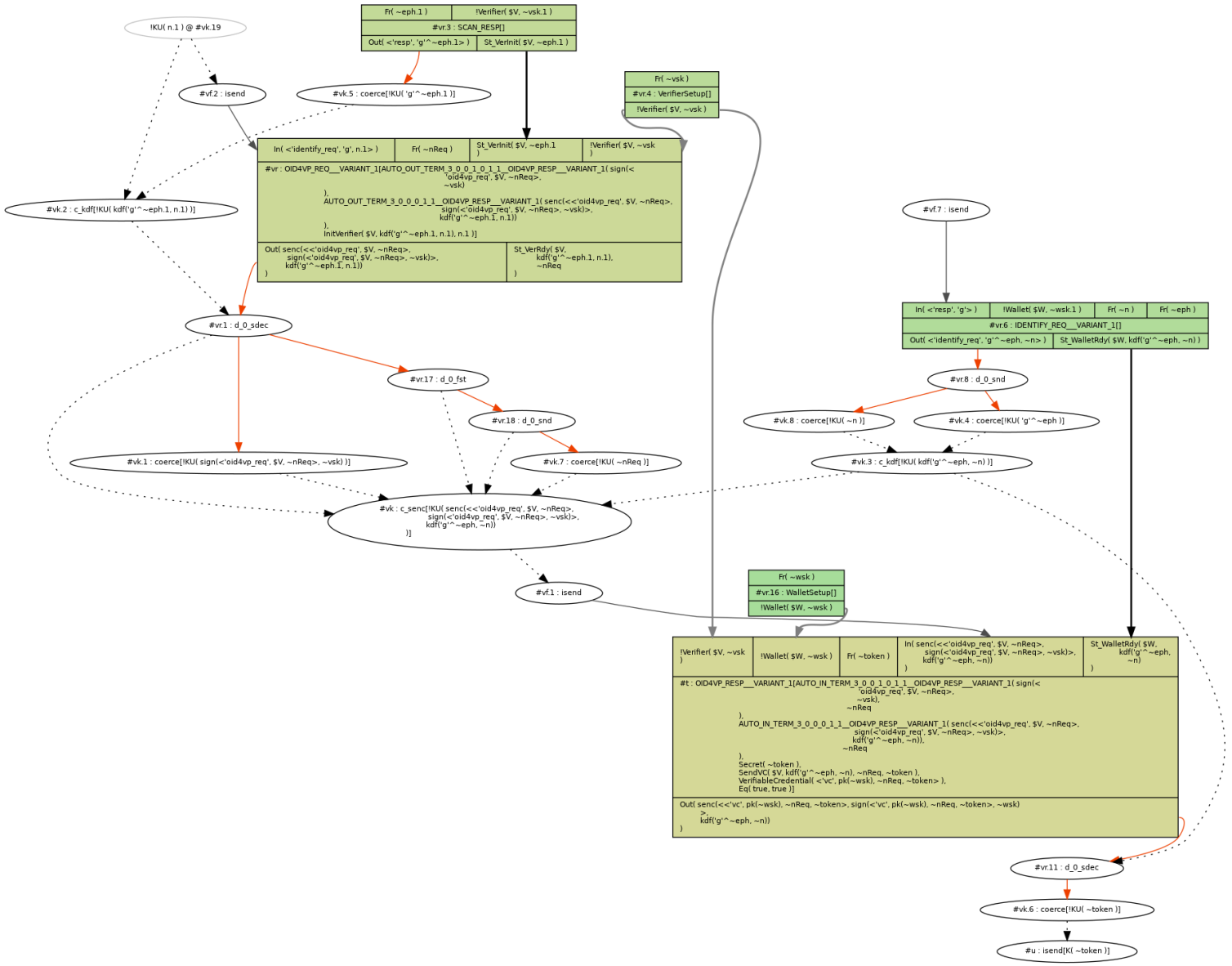
- Without sharedKey in request, neither Secrecy nor Injective Agreement hold
  - Easy mitigation against active MITM
- Probably one wants to link received and verified credential to BLE session
  → To device
  → To holder
- How would this work in practice, though? I cannot “see” the session, I can only see device
  - Scan another QR code?
  - Show a number?
  - Don’t specify at all?

Suggestions

- A design should meet certain **security requirements** against a certain **adversary**
  - Specify both!
- Add hash of shared key into signed request
- I have more nitpicks

- I volunteer to put my thoughts and suggestions in writing
- Where to put formal models? Should this be reproducible?

Attack on Secrecy



Attack on Injective Agreement