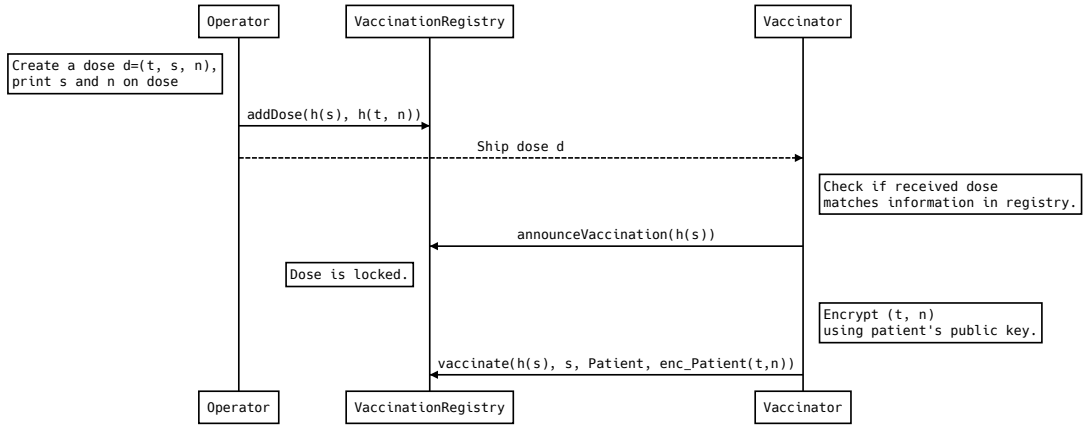
Implementation

Registering

In the present implementation the register operation is done by generating a new keypair and publishing the public key using the *register* function. This way, however, if the private key is lost all the information in the registry concerning your vaccinations is lost aswell.

If the system is applied in the real world, a system such as the "Handy-Signatur", can be used to sign a predefined value and use this signature to derive the keypair. This way if the private key is lost it can be recovered only by the person the keypair belongs to.

Vaccination process



In the registry a dose d is identified by the hash of its secret value s. As long as the dose is not used, this value is not shared in the registry. When creating a dose the operator adds $h(s)$ and $h(t, n)$ to the registry. t is the type of the vaccine and n is a random nonce. s and n is then printed on the dose and can be read by the vaccinator. This can be done on the web front-end on the Operator and Vaccinator tabs, and the information is printed as a QR-Code.

Before applying the dose the vaccinator checks if the received dose matches the one in the registry (by comparing $h(t, n)$) and locks the dose using *announceVaccination*. Then the dose information is encrypted using the patients public key and this encrypted information is made public along with the secret value s. By doing so everyone can verify that the vaccinator actually had the dose, because they knew s.

The commit reveal scheme involving *announceVaccination* and *vaccinate* is required to ensure the value s can not be "stolen" before the vaccination is processed in the registry. After calling *announceVaccination* only the vaccinator who called this function is also able to call *vaccinate* for a specific $h(s)$. This lock is released after a fixed number of blocks.

The nonce n is used to ensure the type of the applied vaccine (which can be read from the registry if $h(s)$ is known) can not be inferred by others by brute-force guessing $h(t)$.

Verifying vaccinations

A patient can query all received vaccinations using *getNumberOfVaccinations* and *getVaccination*. The latter returns $h(s)$ and $(t, n)_{PK(Patient)}$ which can be decrypted using the private key. The validity of the vaccination can then be verified by calculating $h(t, n)$ and comparing it with the dose stored in the registry.

Disclosing vaccination information

If a person P_0 wants to disclose their vaccination against a specific disease to another person P_1, they can again store (t, n) in the registry, but this time encrypted with the public key of P_1: $discloseVaccination(h(s), (t, n)_{PK(P_1)}, index, P_1)$. Here, $index$ is used by the registry to check if P_0 actually received the dose identified by $h(s)$.

P_1 can then query and verifiy the disclosed vaccinations using *getNumberOfDisclosedVaccinations* and *getDisclosedVaccination*.