Nym Private Covid Certificate (PCC) - A Coconut Prototype

The Nym Private Covid Certificate prototype is a privacy wrapper for digital COVID-19 passports, showcasing a very first use-case for Coconut credentials technology. The prototype can work in tandem with existing Covid applications, simply adding a layer of privacy protections by enabling blind issuance of attributes by issuers and selective disclosure of credentials by the users. This prevents verifiers, such as venues or airlines from gathering unnecessary personal data from people. All they need is to be able to verify whether or not the person has the right to access services (i.e. whether they are vaccinated, over a certain age or other such requirements).

'Coconut' is a cryptographic signature scheme that produces privacy-enhanced credentials. It lets application programmers who are concerned with resource access control to think and code in a new way.

Most of the time, when we build system security, we think of *who* questions:

- Has Alice identified herself (authentication)?
- Is Alice allowed to take a specific action (authorisation)?

Coconut fundamentally changes these questions. Rather than asking *who* a user is, it allows application designers to ask different questions, mostly centered around questions of *rights*:

- Does the entity taking this action have a right to do X?

This allows a different kind of security. Many of the computer systems we talk to every day don't need to know *who we are*, they only need to know if we have a *right to use* the system. Coconut allows signing authorities (issuers) and validators (in the case of Covid certificates the Health Authorities) to work together to determine whether a given private key holder has a right to take an action. The credentials are generated cooperatively by decentralised, trustless systems.

Once the credentials are generated, they can be *re-randomized:* creating entirely new credentials, which no one has ever seen before, can be presented to service providers, and magically validated without being linkable back to the credential originally given out by validators.

These properties allow Coconut credentials to act as something like a decentralized and fully private version of OAuth credentials, or like cryptographic bearer tokens generated by decentralised systems. The tokens can be mutated so that they are not traceable, but still verified with the original permissions intact.

As mentioned earlier we have built a prototype application called Private Covid Certificate (PCC), to showcase the capabilities of Coconut credentials. This app is not a production ready replacement of the existing digital covid certificates. But will show you how Coconut could enhance the privacy of such systems. You can check the PCC prototype and play around with it here: https://nym-pcc.vercel.app/

Below are the assumptions and flows of Nym's PCC application:

- We assume (1) that a user has a phone and has downloaded the PCC app. The app has means to authenticate to the health authorities as the owner of the phone. We will simply mock this part, since it is already done by the public health providers, and not the point of the POC.
- We assume (2) that the phone has a secure element (SE), which we will also mock, within which simple computations can be performed. It can contain secrets, which are hard to extract even for the owner of the phone. We will use the SE to bind a show of the credential to a phone -- which we consider to be a proxy for binding it to a physical user.
- In terms of credentials the parties are mapped as follows:
    - Credential Issuer: 3 validators approved by the health authority who can issue the credentials in a decentralized fashion.
    - Credential subject: the user
    - Credential verifier: a party that wants to allow or disallow physical access based on vaccination status.
- The protocol flow proceeds as follows:
    - Step 1. As per assumption (1) the app on the phone authenticates as a health service user with the health service on-line service, and establishes a secure channel (eg. TLS). This authentication can be done or bootstrapped using a password, a key, a number received by post, a fuller registration process (passport, biometrics) or in person. Once the user is authenticated, if a record of vaccination exists for them in the health authorities database, a copy of their Covid certificate is downloaded in clear text in the app over the secure channel. We are simply mocking this bit as it is not the problem PCC is solving.
    - Step 2. The User triggers issuance of the private credential version of their certificate. The secure element of the phone produces a fresh secret *s*, and encrypts *s* and signs it with the *SE* signature. *(eg A = E(s), Sign(A, SE))*. This is part of the user side of the blind issuance protocol of coconut. In this case the blind value is the secret *s*. Only a limited number of certificates are issued per unit time, i.e. a user that loses their phone and re-registers a new one can get a new credential but not a large number of them.
    - Step 3. Validators each issue a part of the credential with attributes (Name, DoB, Vaccine Status, s). The first three are known to the validators, while s is not, the latter uses the blind issuance facilities of coconut. The result is a redandomizable signature on these attributes (the coconut credential), which is given back to the user's app over the secure channel.
    - Step 4. The user app wants to show the vaccine status to a specific verifier or at a specific time (say V and T). The user scans QR code of the verifier (restaurant, bar, etc) rerandomizes the signature and associates with it a proof *NIZK{(name, dob, vaccine, s): Cert(name, dob, vaccine, s) /\ X = H(V,T)^s /\ vaccine = status)*. In the computation of H(V,T)^s, the user can compute instead H(V,T,N)^s, where N is a random nonce received from the verifier just before the user computes the credential

show algorithm. This would be a way to avoid Step 5 described below, if we wish that the verifier always checks that the Phone's SE contains the secret s. The value X represents a one-time tag for this device (s is within the SE) at this time and for this verifier. The rerandomized signature and NIZK is shown to the verifier, e.g. through a QR code or Near Field Communication NFC.
    - Step 5. The verifier receives the signature and NIZK, and checks if they are valid. Then they optionally can interrogate the Phone's SE to ensure it contains the secret s. Specifically using NFC the verifier relays the signature and proof to the SE, along with a challenge C (for freshness) and the values V, T. The SE verifies the signature and NIZK (does the verification) and checks X = H(V,T)^s. If there are no errors it returns a signature on C using a key certified by the SE manufacturer, to ensure the response came from the SE. This flow increases the assurance that the physical person / phone is associated with the certificate.
    - Steps 2-5 are the blind issuance and selective disclosure show procedures of the coconut credential. More attributes can be disclosed by extending the NIZK proof.
- The protocol is of course over complicated still over the benefit it provides: if one trusts an SE to store s, and not clone it, not leak it, and answer challenges as in step 5, then why not also store the vaccine status within the SE, and answer a challenge C with a response *{C, status}_signed_by_SE*.

App flows by actors:

User (Prover)	**<u>Health Authority (Validators or Issuer)</u>**
1. User gets vaccinated **3.** User receives a unique id through post or other methods to sign up to the app **4.** If user id matches Issuer's records → user's vaccination details will be downloaded in the app in clear text (if any of the user's details are wrong the user will have to correct that outside the app by contacting their health service provider) **5.** User initiates issuance of coconut credentials **5.1** Phone sends user's public attributes (name, DoB, vaccination type, …) + private attributes (a fresh secret *s generated by phone's Secure Element) --> this is part of the user side of the blind issuance protocol of coconut. In this case the private value is the secret s.* **5.4** Unblinds credentials and merges them together to have a full credential **7.** Scans QR code of the service provider → agrees to show credentials as per SP's policy	**2.** Users vaccination details goes into a database **5.2** validators validate the public attributes sent by the user against their database **5.3** Validatros each issue part of the blinded coconut credentials and send them back in the clear over TLS. --- **<u>Verifier (Service Provider)</u>** **6.** Has a policy in a QR code format + a random number that changes periodically. **8.** Scans user's generated QR code and verifies it against its own policy and a random nonce **9.** Can verify if the QR code created by the user is **10.** Future work: If the verifier wishes to check that the Phone's SE contains the secret s, this can be done by integrating NFC capability to

→ a rerandomised set of creds are generated in a QR code that includes the random nonce (received from SP's QR code just before the user computes the credential show algorithm)	the Verifier's device (Nym is not building this feature)