NTNU

Kunnskap for en bedre verden

Anonymous Tokens for more Private Contact Tracing

Henrik Walker Moe (Bekk),

Tjerand Silde (NTNU),

and Martin Strand (FFI)

BEKK FFI Forsvarets forskningsinstitutt

Content

Anonymous Communication

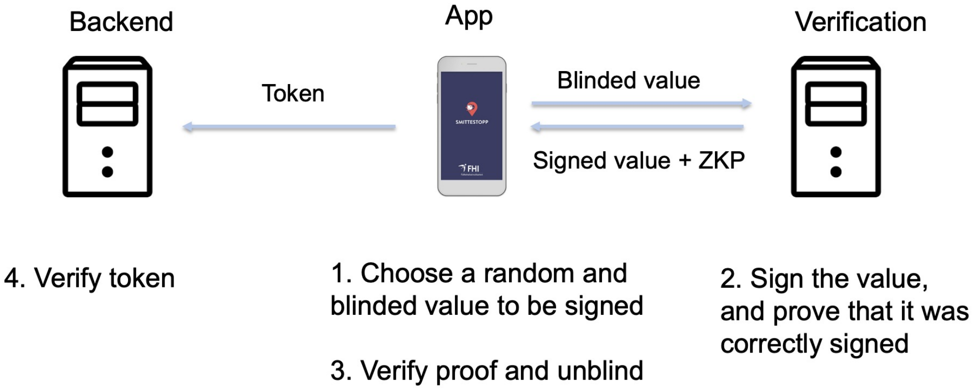
Digital Contact Tracing

The ”Smittestopp 2.0” App

The Cryptographic Protocol

Continued Work

Additional Resources



Backend

App

Verification

Token

Blinded value

Signed value + ZKP

4. Verify token

1. Choose a random and blinded value to be signed

3. Verify proof and unblind

2. Sign the value, and prove that it was correctly signed

Anonymous Communication

Many flavors in literature: Anonymous Tokens, Anonymous Credentials, Blind Signatures, Partially Blind Signatures, …

Properties: unlinkability, unforgeability, public or designated verifiability, revocation, rounds of interaction, efficiency, …

Underlying primitives: factoring, quadratic residues, (elliptic curve) discrete logarithms, bilinear pairings, …

Anonymous Communication

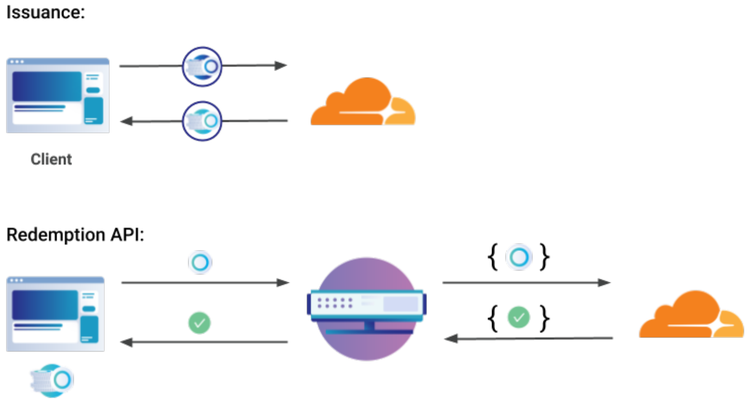
Example: Privacy Pass

Developed by Cloudflare.

Use-case: Users should be able to use Tor without solving CAPTCHAs all the time.

Security: Should not track users, but also prevent DDOS attacks etc.

Problem: Revocation of tokens.



Anonymous Communication

Example: PrivateStats

Developed by Facebook.

Use-case: Used to collect anonymous telemetry data from WhatsApp.

Solve revocation by deterministically updating the public key every day.

Problem: Large overhead for tokens.

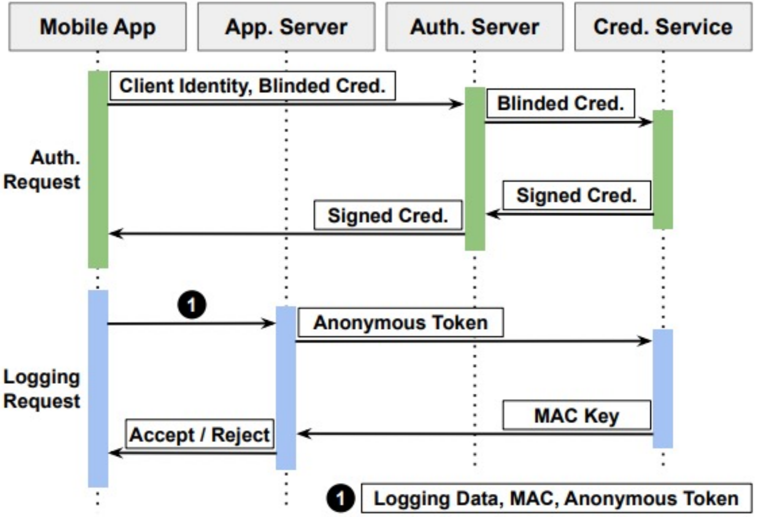


Fig. 1: Protocol flow diagram.

Digital Contact Tracing

The Norwegian Institute of Public Health has developed an app to supplement traditional contact tracing.

The app sends you a notification if you have been close to someone that has tested positive for Covid 19.

The hope is that this may be faster and may notify contacts that you forgot about or didn’t know about.

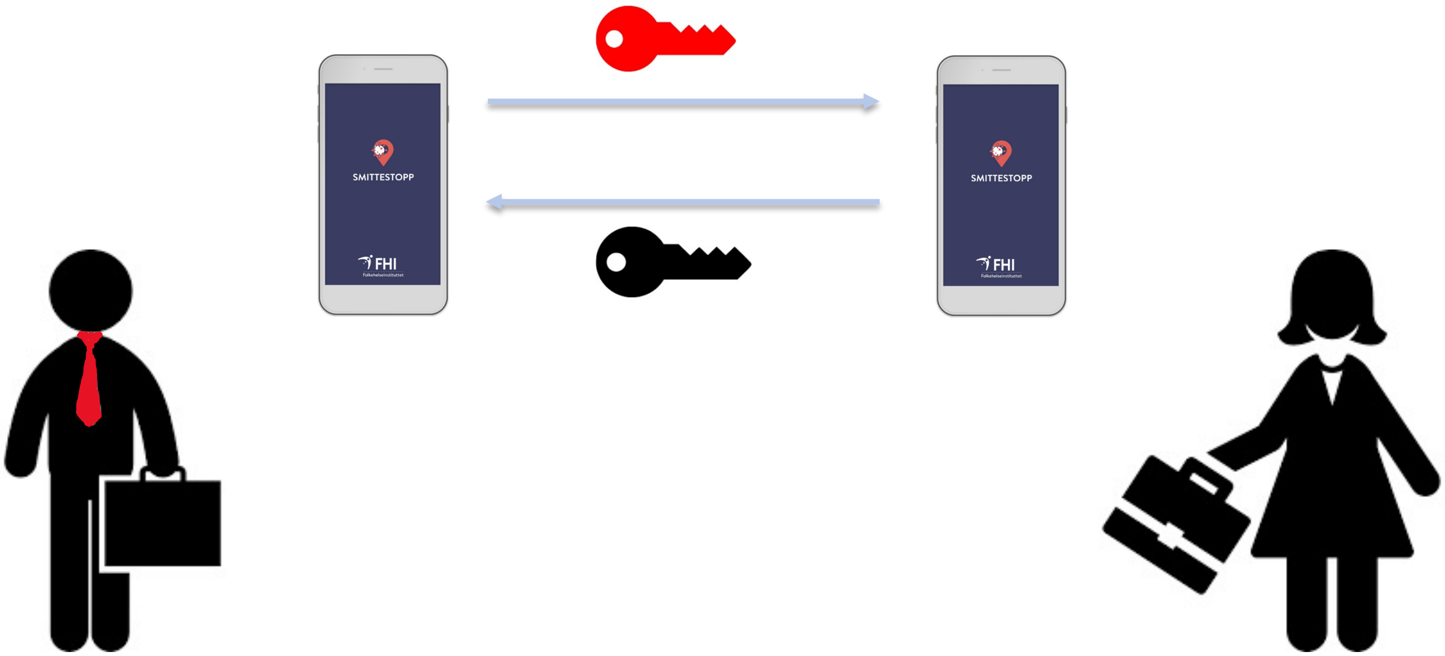
Digital Contact Tracing

All data is stored on the user’s phone. It uses Bluetooth for communication with other phones, but no GPS tracking.

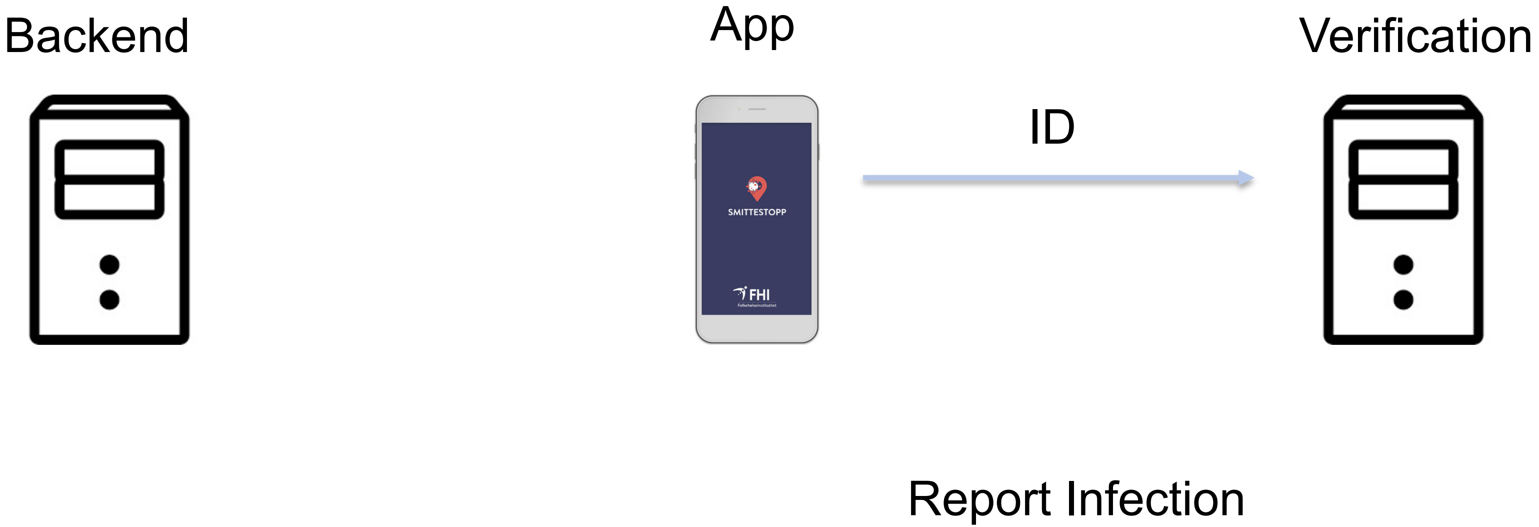
You only identify yourself to report a positive test, and then you upload the “infections keys” anonymously to the server.

The other users check locally if they have been in touch with someone who has uploaded their keys.

Smittestopp



Smittestopp



Backend

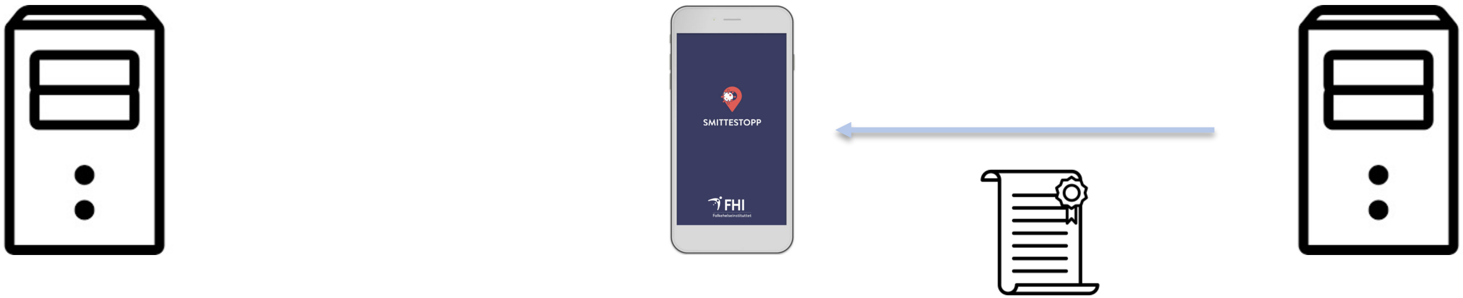
App

ID

Verification

Report Infection

Smittestopp



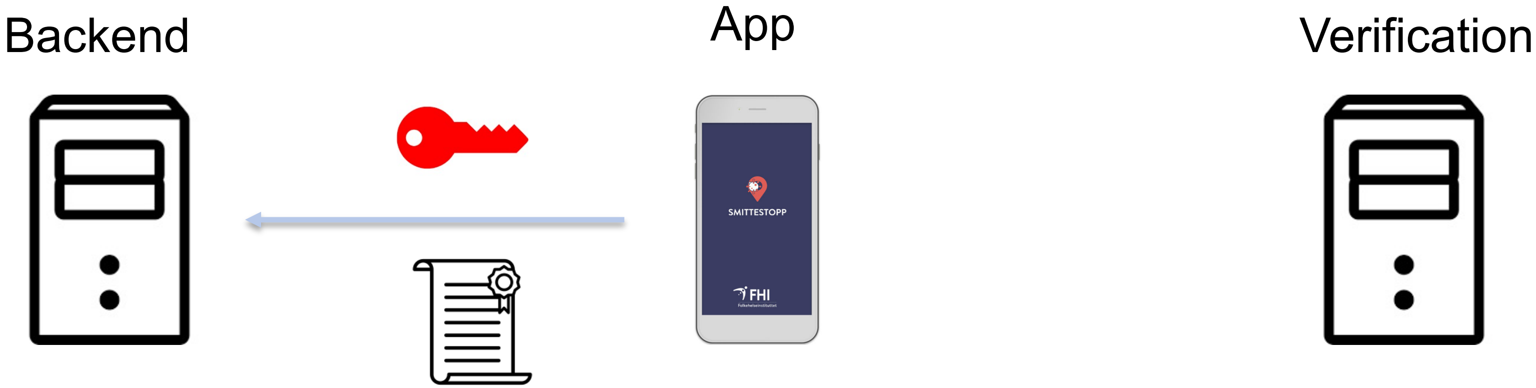
Backend

App

Verification

Confirm Infection

Smittestopp



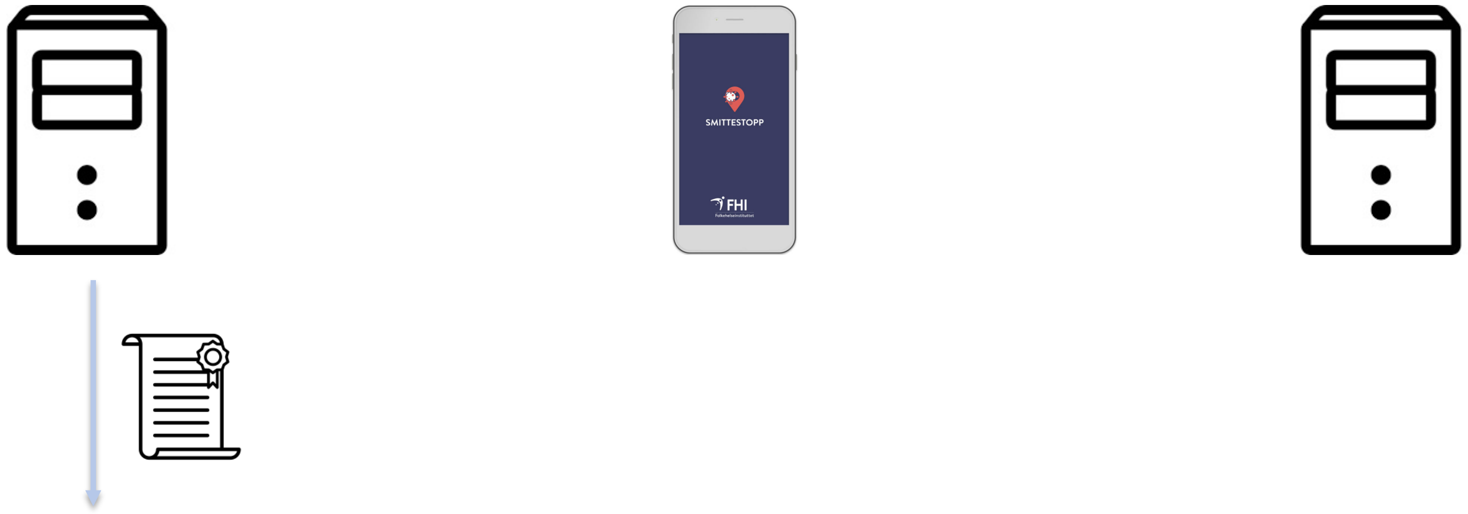
Backend

App

Verification

Send Infection Keys

Smittestopp



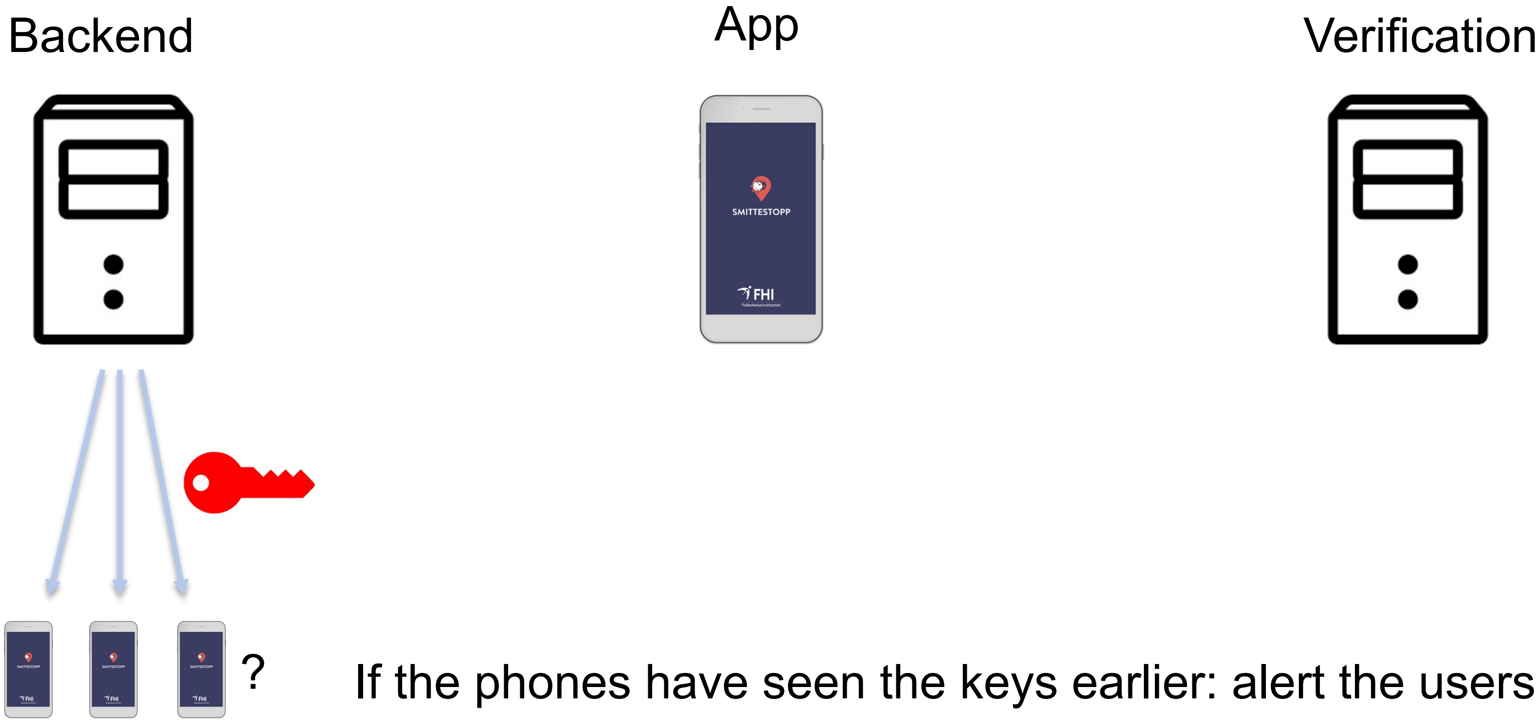
Backend

App

Verification

Valid?

Smittestopp



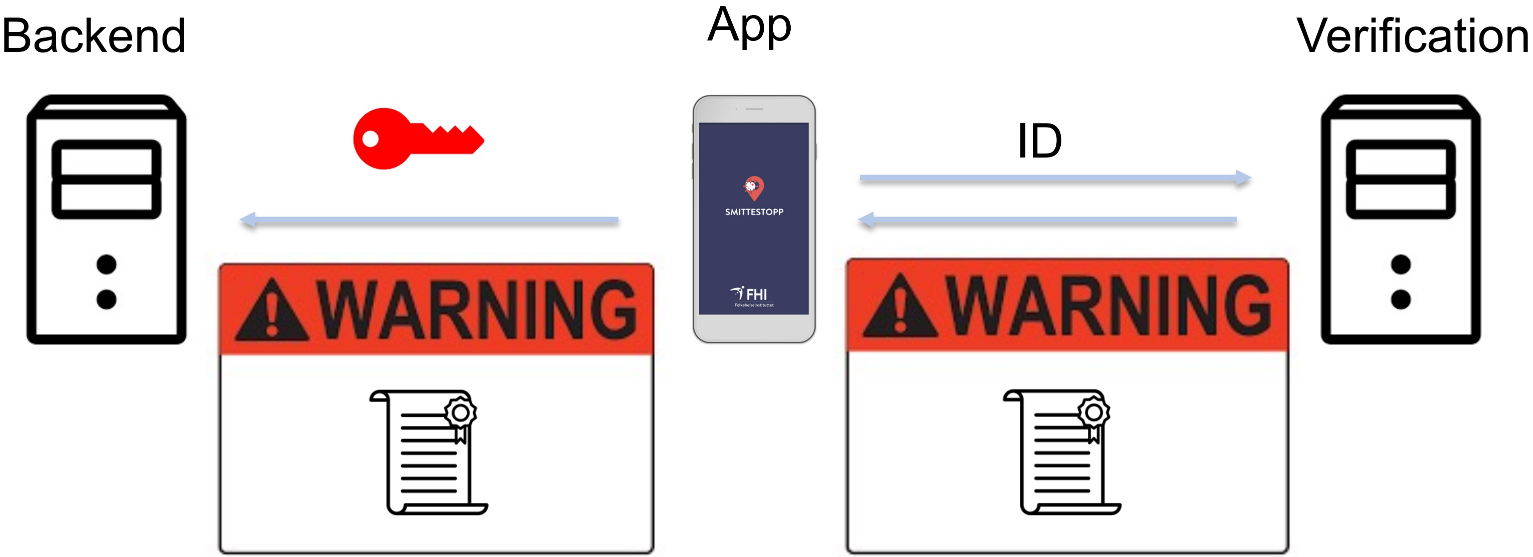
Backend

App

Verification

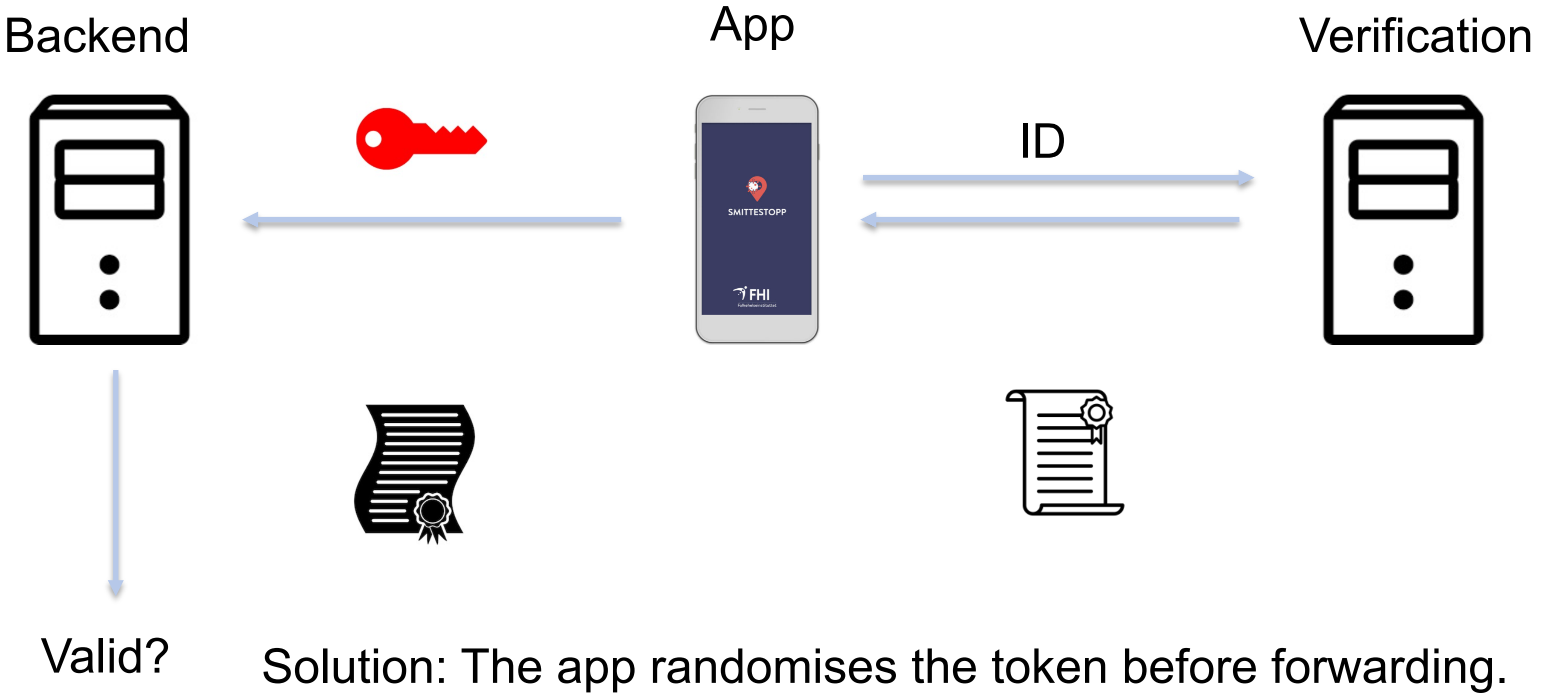
If the phones have seen the keys earlier: alert the users.

Smittestopp



ID can be tied to infection keys when uploading!

Smittestopp



Solution: The app randomises the token before forwarding.

Smittestopp

We require certain security properties:

- Correctness: Backend will accept an honest randomised token.
- Unlinkability: Verification and Backend cannot link tokens.
- Unforgeability: Malicious users cannot create new tokens.

We also require that:

- Communication between app and servers are private.
- Communication between app and servers are authenticated.

Smittestopp

Problem: Users should not be able to hold onto a token and upload later. We revoke all unspent tokens older than 3 days.

Solution: The client needs to download new public keys from a public API every time it wants to talk to the server. Impractical.

Note: Still possible to correlate identities with “infection keys” if the servers are logging IP-addresses and timestamps.

Protocol

Hash function SHA-256

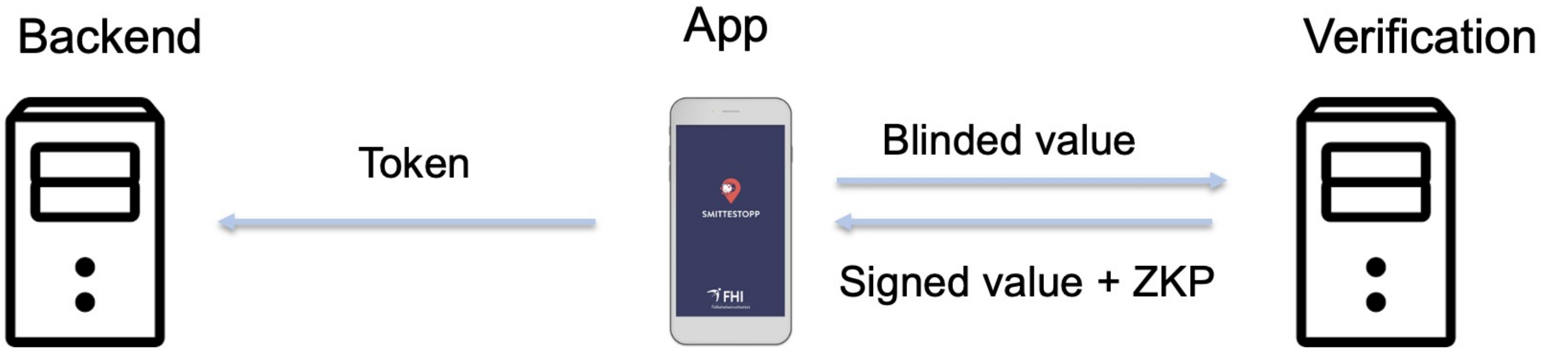
Hash function H such that:

- Output $y = H(x)$ is random
- It is hard to find x and y such that $H(x) = H(y)$
- Can transform string t to elliptic curve point $T = H(t)$

Elliptic curve P-256

- Elliptic curves give high security and efficiency
- Hard to find secret a given G and $A = a \cdot G$
- Randomised points hide all information

Protocol



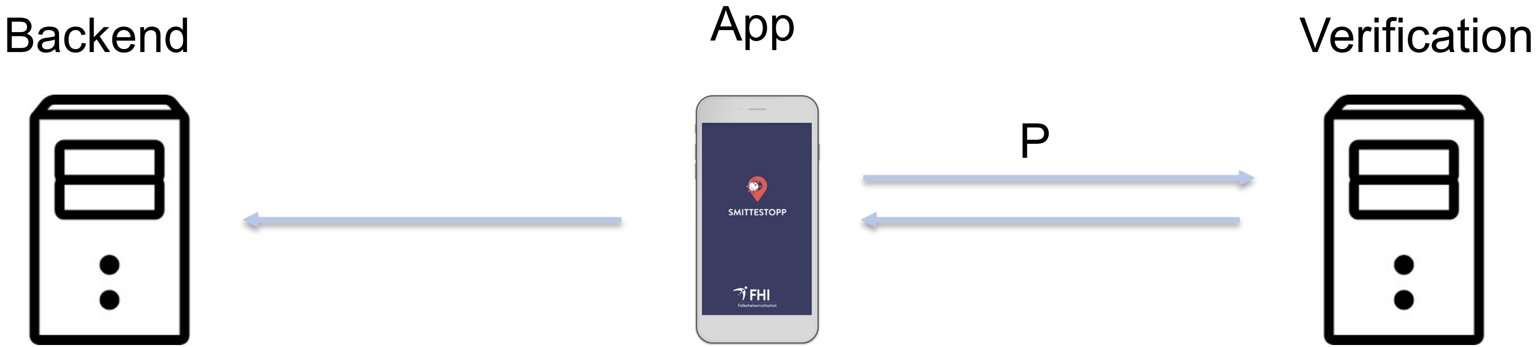
4. Verify token

1. Choose a random and blinded value to be signed

3. Verify proof and unblind

2. Sign the value, and prove that it was correctly signed

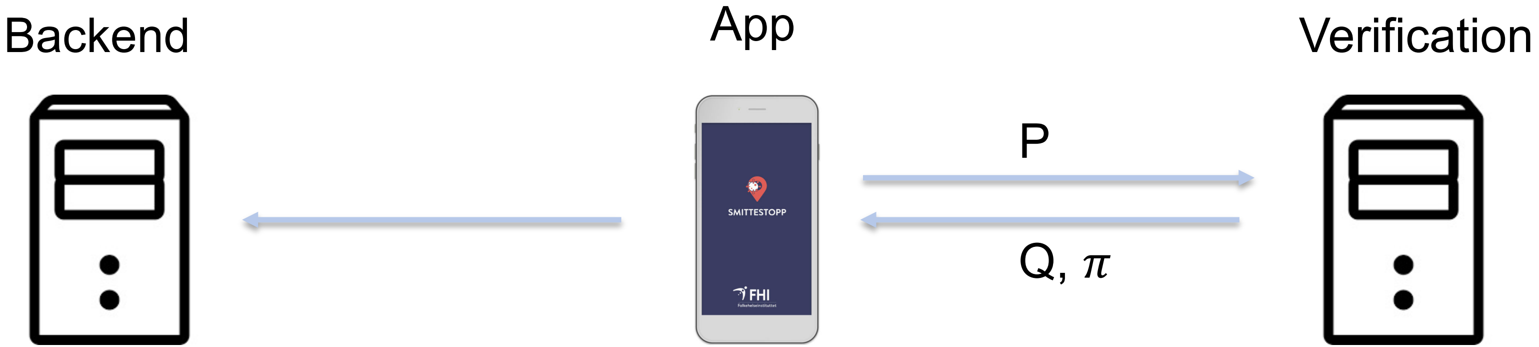
Protocol



$t \leftarrow$ random bits
$T = \mathrm{Hash}(t)$

$r \leftarrow$ random integer
$P = r \cdot T$

Protocol



Backend

App

Verification

P

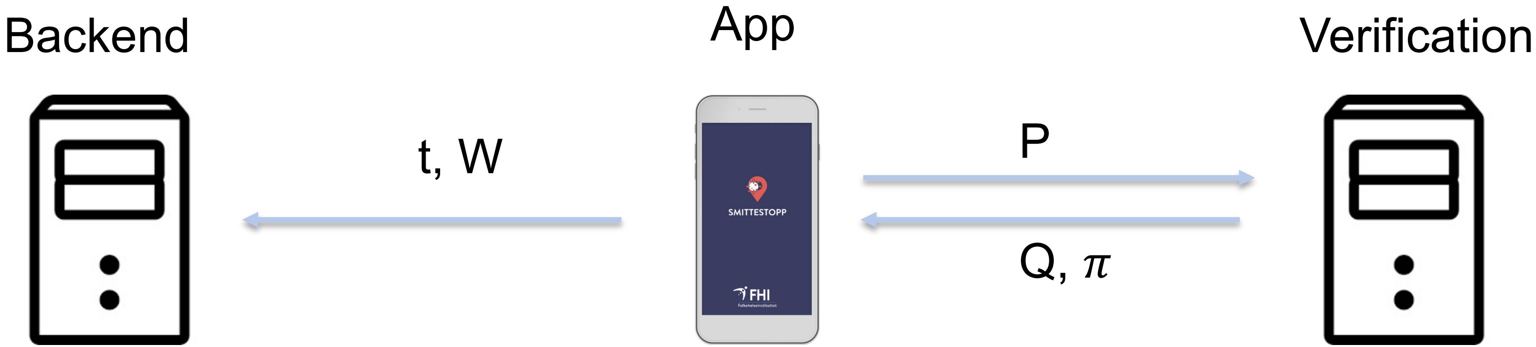
Q, π

$k \leftarrow$ signing-key

$Q = k \cdot P$

$\pi = \mathrm{ZKP}(P, Q, K, k)$

Protocol



Backend

App

Verification

t, W

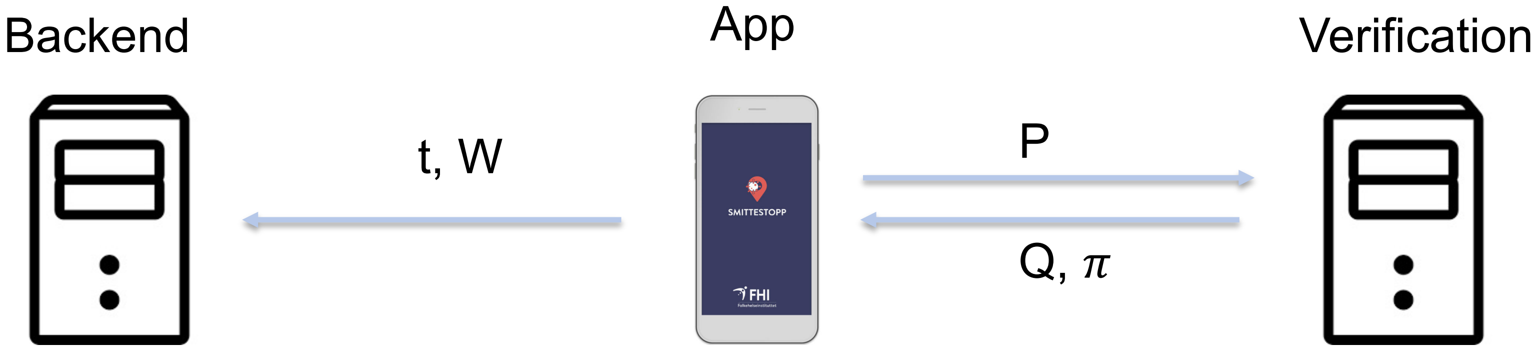
P

Q, π

W = (1/r) · Q = k · T

Verify that π is valid

Protocol



Backend

App

Verification

t, W

P

Q, π

k ← signing-key

T = Hash(t)

W’ = k · T

Are W’ and W the same?

Continued Work

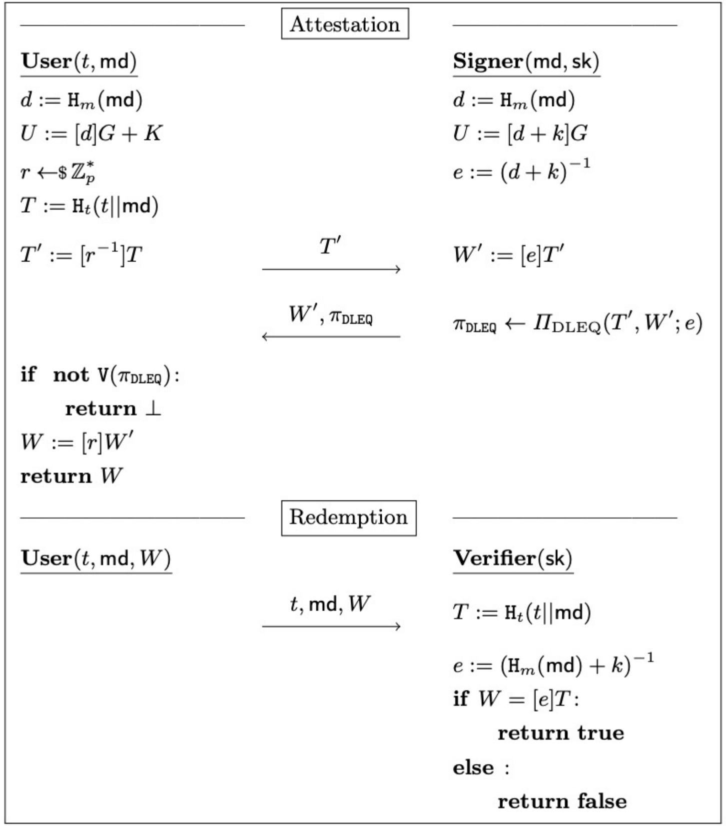
New anonymous token protocol with public metadata and public verifiability.

Based on ECC, avoids pairings.
Public verification with pairings.

Revocation based on metadata.

Currently being implemented by summer students working at FFI.

Paper is available at: ia.cr/2021/203



Attestation

```
User$(t, \mathsf{md})$                                      Signer$(\mathsf{md}, \mathsf{sk})$
$d := \mathtt{H}_m(\mathsf{md})$                            $d := \mathtt{H}_m(\mathsf{md})$
$U := [d]G + K$                                             $U := [d + k]G$
$r \leftarrow\$ \mathbb{Z}_p^*$                              $e := (d + k)^{-1}$
$T := \mathtt{H}_t(t||\mathsf{md})$
$T' := [r^{-1}]T$            $\xrightarrow{T'}$              $W' := [e]T'$
                    $\xleftarrow{W', \pi_{\mathrm{DLEQ}}}$    $\pi_{\mathrm{DLEQ}} \leftarrow \mathit{\Pi}_{\mathrm{DLEQ}}(T', W'; e)$
if not $\mathtt{V}(\pi_{\mathrm{DLEQ}})$:
    return $\perp$
$W := [r]W'$
return $W$
```

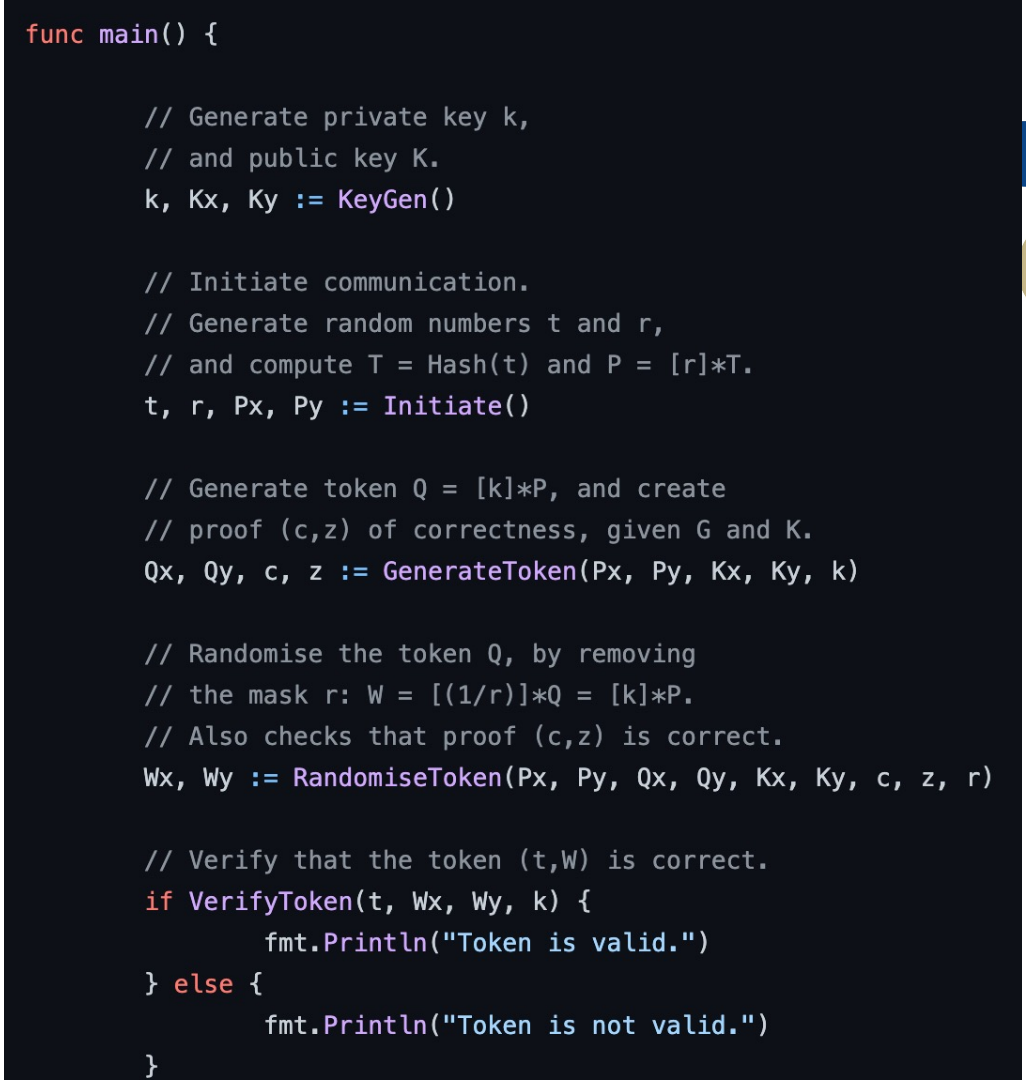
Redemption

```
User$(t, \mathsf{md}, W)$                                   Verifier$(\mathsf{sk})$
                    $\xrightarrow{t, \mathsf{md}, W}$        $T := \mathtt{H}_t(t||\mathsf{md})$
                                                            $e := (\mathtt{H}_m(\mathsf{md}) + k)^{-1}$
                                                            if $W = [e]T$:
                                                                return true
                                                            else :
                                                                return false
```

Resources

Proof of Concept implementation of Anonymous Tokens in Go:

github.com/tjesi/anonymous-tokens



```go
func main() {

        // Generate private key k,
        // and public key K.
        k, Kx, Ky := KeyGen()

        // Initiate communication.
        // Generate random numbers t and r,
        // and compute T = Hash(t) and P = [r]*T.
        t, r, Px, Py := Initiate()

        // Generate token Q = [k]*P, and create
        // proof (c,z) of correctness, given G and K.
        Qx, Qy, c, z := GenerateToken(Px, Py, Kx, Ky, k)

        // Randomise the token Q, by removing
        // the mask r: W = [(1/r)]*Q = [k]*P.
        // Also checks that proof (c,z) is correct.
        Wx, Wy := RandomiseToken(Px, Py, Qx, Qy, Kx, Ky, c, z, r)

        // Verify that the token (t,W) is correct.
        if VerifyToken(t, Wx, Wy, k) {
                fmt.Println("Token is valid.")
        } else {
                fmt.Println("Token is not valid.")
        }
```

Resources

- Alex Davidson - Privacy Pass: Bypassing Internet Challenges Anonymously (https://youtu.be/9DsUi-UF2pM)
- Nick Sullivan - Privacy Pass: A Lightweight Zero Knowledge Protocol Designed for the Web (https://youtu.be/HIqBJKnnHVk)
- Privacy Pass Paper: https://www.petsymposium.org/2018/files/papers/issue3/popets-2018-0026.pdf
- Documentation for our anonymous-tokens library: https://github.com/HenrikWM/anonymous-tokens/wiki
- Notes about anonymous contact tracing: https://github.com/HenrikWM/anonymous-tokens/tree/main/docs
- Blog-post about tokens with public metadata: https://world.hey.com/tjerand/anonymous-tokens-with-public-metadata-1253024d

Thank you! Over to Henrik...

Slides: tjerandsilde.no/talks

Twitter: TjerandSilde