REMOTE DOCUMENT ENCRYPTION IN FILESENDER

Job Doesburg

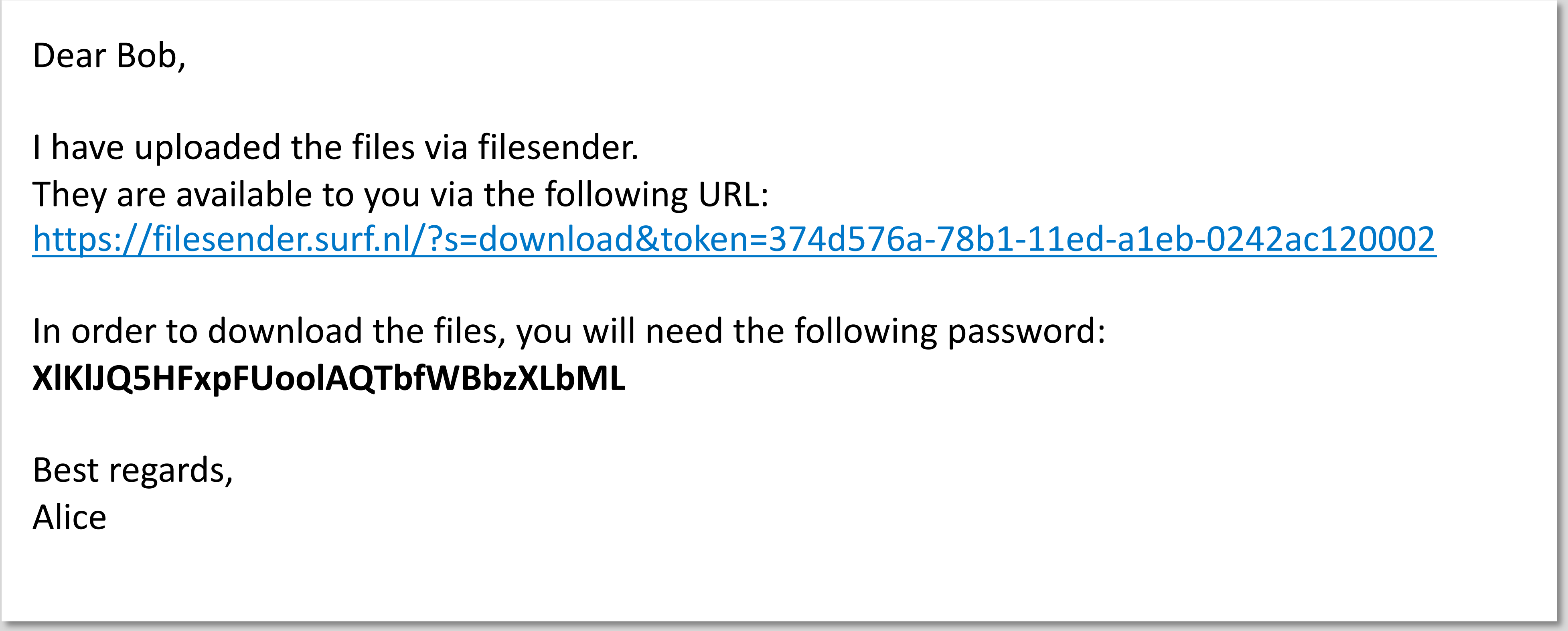
The FileSender project

- https://filesender.org
- Upload large files and make them available for downloading
- Large data sets, or sensitive data
- Anything you don’t want to have in your email attachments
- SURF also doesn’t want to see this data

- End-to-end encryption based on passwords (and PBKDF2)

SURF FILESENDER

Key management with FileSender



Dear Bob,

I have uploaded the files via filesender.
They are available to you via the following URL:
https://filesender.surf.nl/?s=download&token=374d576a-78b1-11ed-a1eb-0242ac120002

In order to download the files, you will need the following password:
XlKlJQ5HFxpFUoolAQTbfWBbzXLbML

Best regards,
Alice

SURF

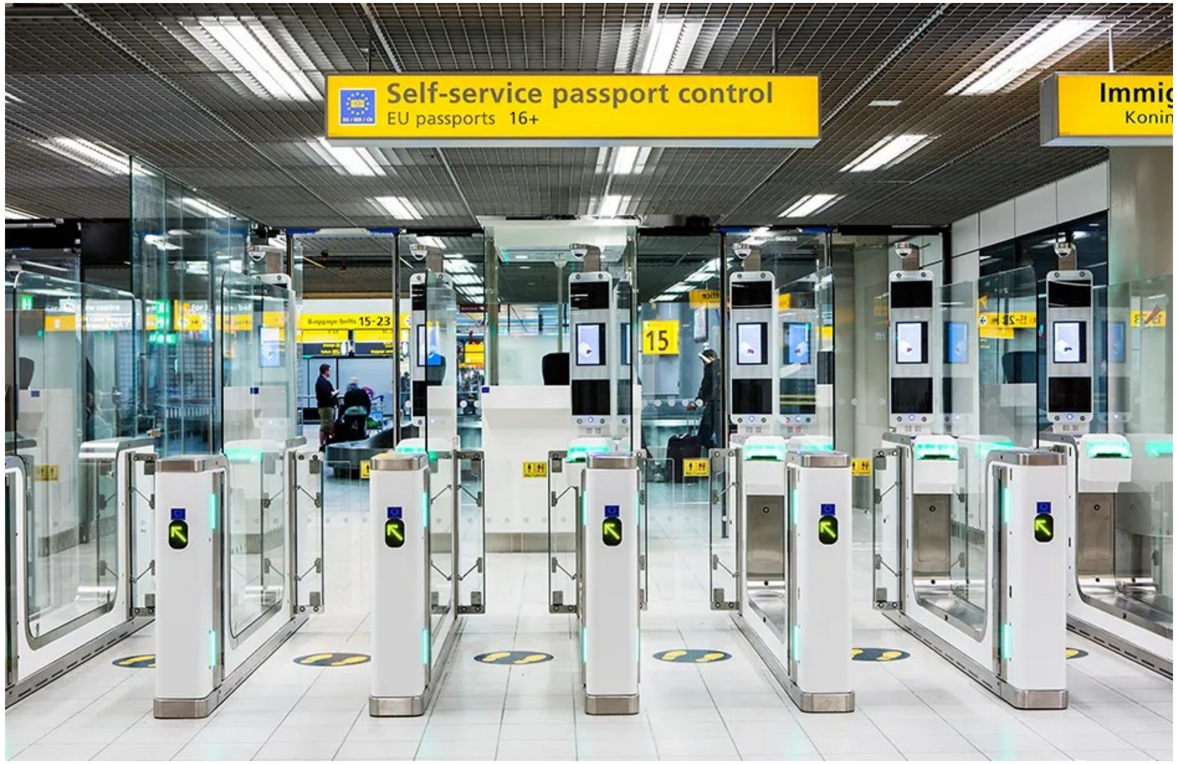
A solution

- Asymmetric keys?
  - Over the whole internet?
    - … PGP?

- If only there existed some PKI for verifying the identity of any person in the world…

An international PKI

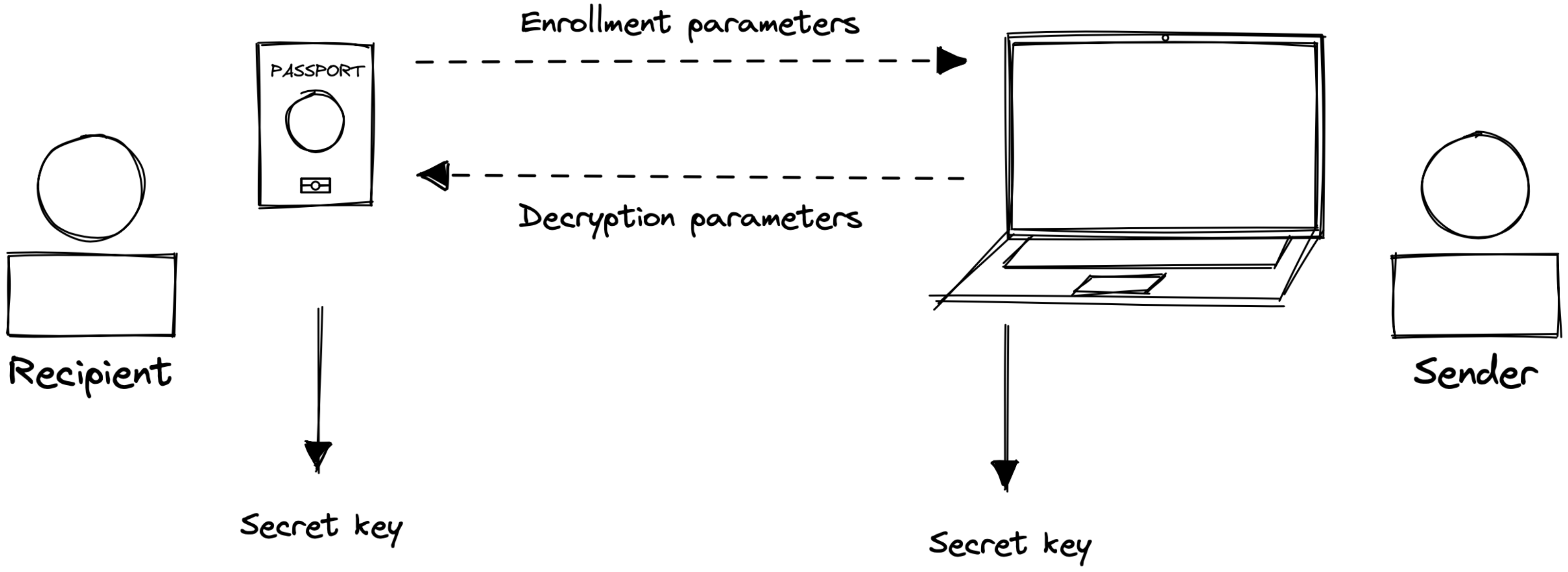
- E-passports (ICAO 9303) with NFC
- Also ID cards, (drivers licenses…)



DigiD

Remote Document Encryption (Verheul, 2017) in a nutshell

- *“Passport as a Yubikey”*



Benefits of RDE for SURFfilesender

- Asymmetric key establishment
  - Download token + key from passport means 2FA-like behaviour for downloading

- ‘People already have an e-passport’ (and an NFC capable phone)

- Use government PKI to confirm identity of recipient
  (when using RDE with document holder authentication)

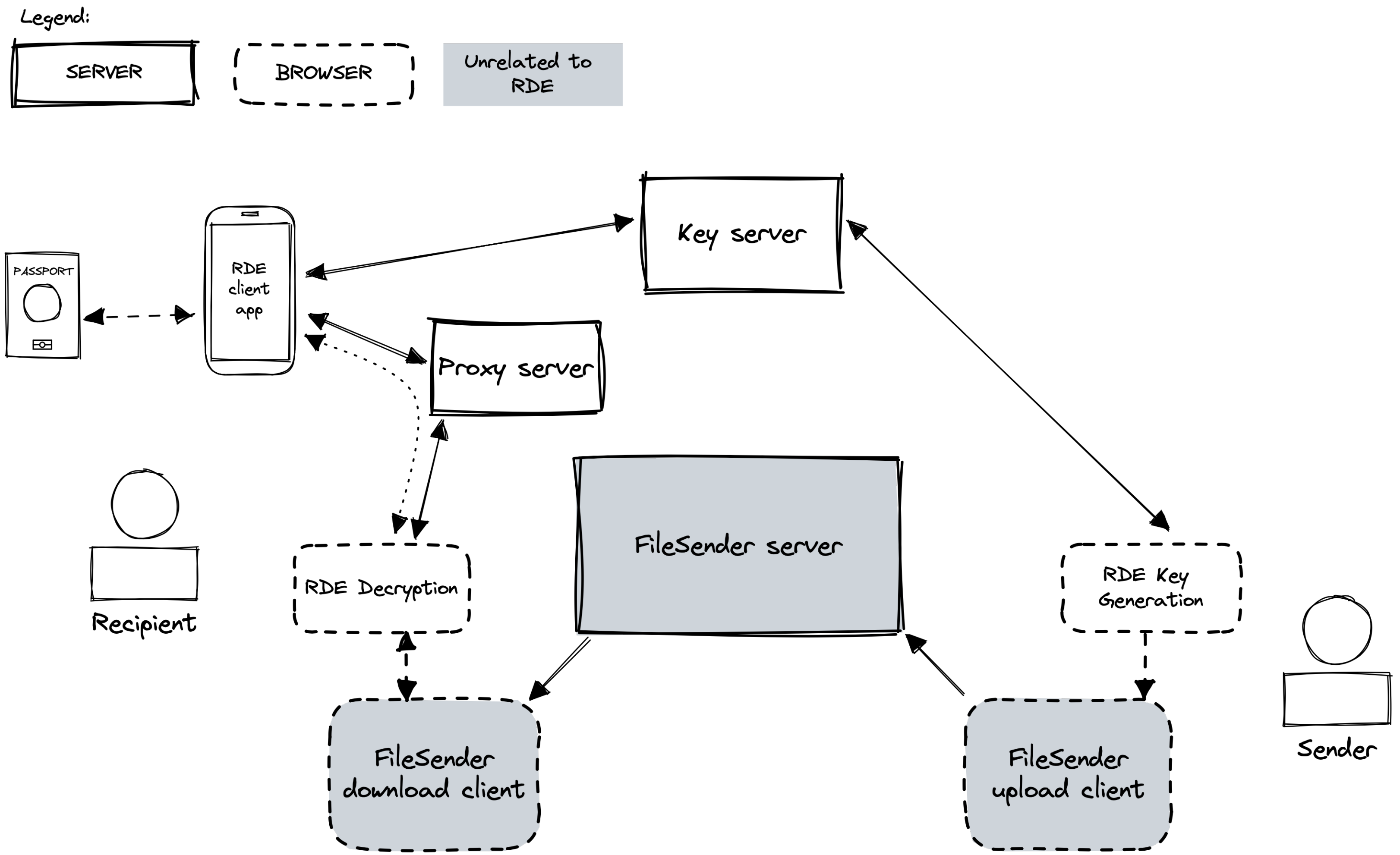
Document holder authentication

- Upon enrolling, not only publish static passport CA public key and contents of one data group (DG14)
- Also include:
    - DG1 (MRZ-data): name, date of birth, nationality, etc
    - DG2: facial image?
    - EFsod: signatures, hashes and certificates to verify everything is legitimate
        - Verify certificate chain against CSCA certificates
            - Dutch National Public Key Directory (https://npkd.nl)
- Sender can verify in-browser, no need to *actually* trust SURF!
- Do note the privacy implications!

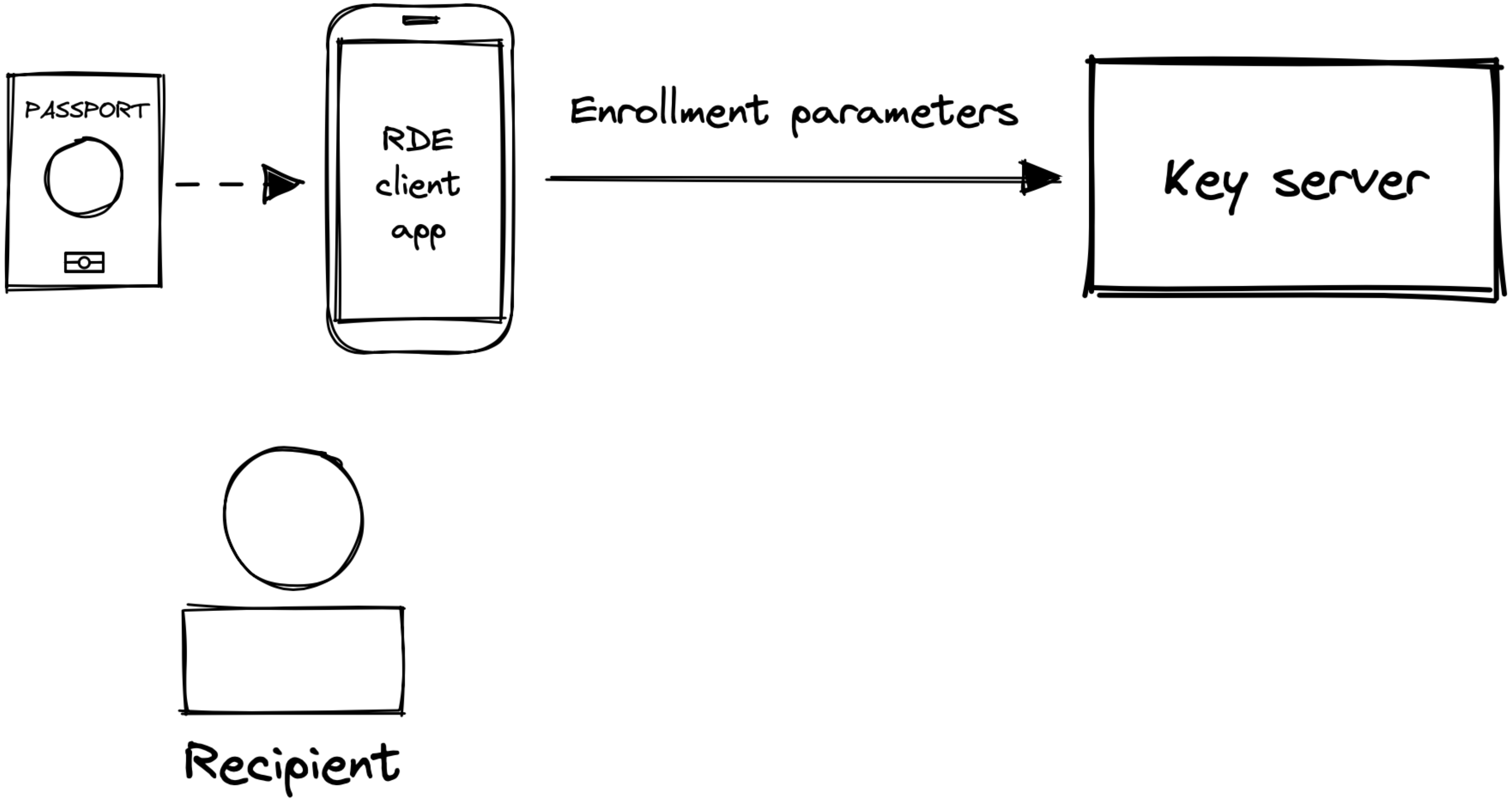
Limitations

- BSN (personal number / social security number) in MRZ-data
    - Processing is restricted in the Netherlands!
    - Deal-breaker for SURF
    - 2021 model of Dutch passports and identity cards don’t include BSN
        - It will take until 2031 for those documents expire…
        - **Until then, no document holder authentication with MRZ data ☹**

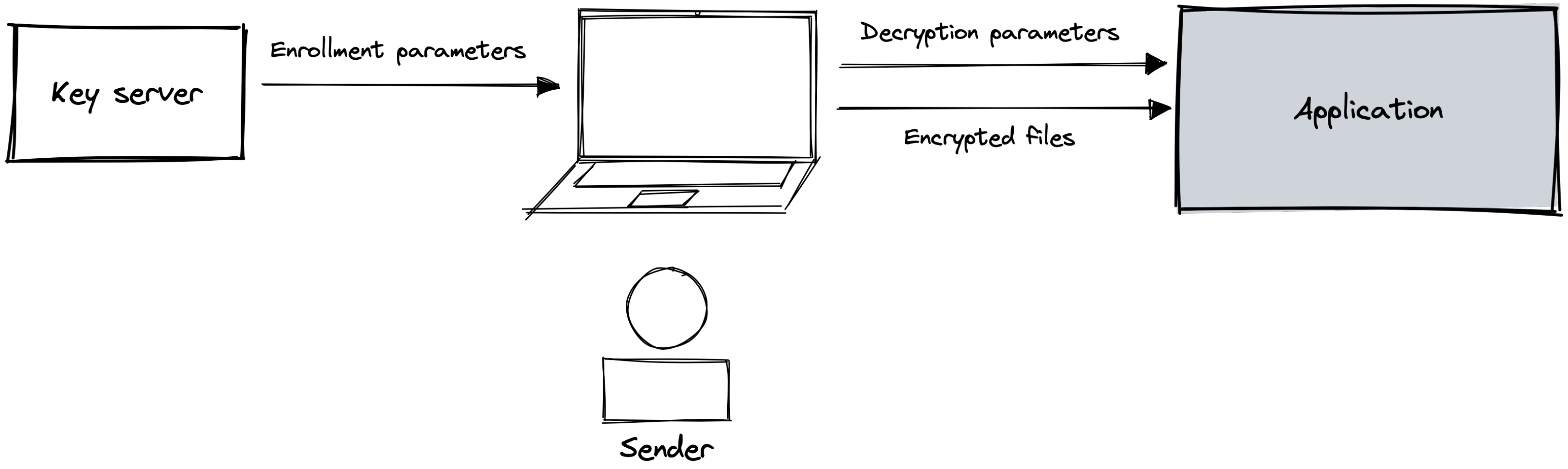
Infrastructure RDE for FileSender



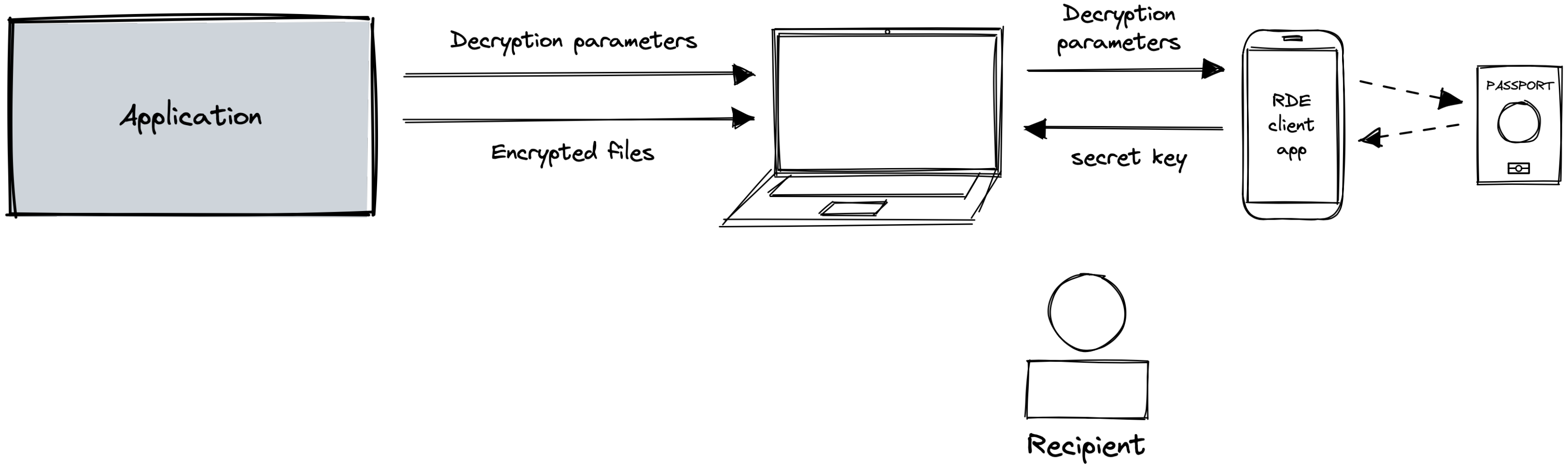
RDE enrollment



RDE key generation



RDE decryption



DEMO

Going forward

- iOS app
- Drivers license support requires small tweaks (and e-residence permits?)
- Usability!
    - User-friendly terminology, explain what’s happening
    - OCR in the app for BAC / PACE
- Key server implementation for production (many decisions to make)
- Encrypting for multiple e-passports

- Prototype → production is a big step to take
- A lot of considerations and configuration options (privacy, key rollover)

Further research

- Split-key infrastructure
  - Remote blocking of a lost/stolen document
  - Possibly even: face scan / liveness check
- Implementing a PIN for unlocking
- USB NFC readers

QUESTIONS

👤 Job Doesburg

✉ job.doesburg@{ru,surf}.nl

💻 demo.rde.filesenderbeta.surf.nl

SURF

ADDITIONAL SLIDES

The trick behind RDE

- Passport can perform Chip Authentication (CA): ECDH key establishment
  - Passport key is **fixed** (signed by country)
  - Only reader key is ephemeral
- After CA, passport communicates with keys, deterministically derived from ECDH

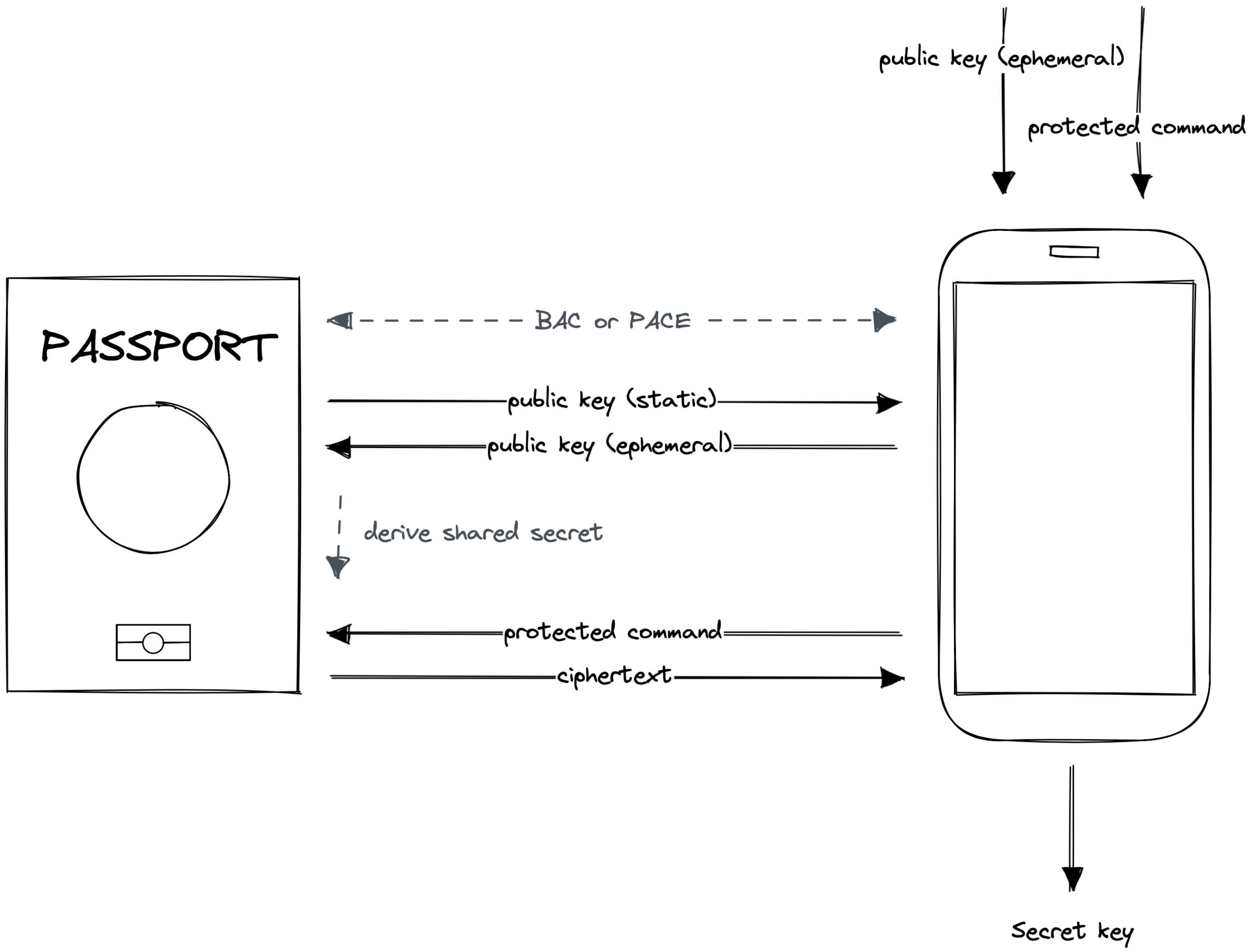
- If a reader selects the same ephemeral key twice, and reads the same data group twice, it **results in the same ciphertext**!
- Use ciphertext as secret key

The trick behind RDE

- **Known from enrollment: passport public key + plaintext DG14**
- Senders choose ephemeral key pair
- Generate shared secret (*passport public key* × *sender private key*)
- Emulate passport ciphertext response to a READ command (known plaintext)
- Forms decryption parameters:
    - Ephemeral sender public key
    - Emulated ciphertext READ command

- Upon decryption, reader sends sender public key
- Passport generates shared secret (*sender public key* × *passport private key*)
- Responds to READ command with same ciphertext

The trick behind RDE



Difference with DigiD passport check

- DigiD = authentication
  - Passport signs DigiD app key
  - Signature is intended to be published
- RDE = encryption
  - Passport generates encryption key
  - Encryption key should not be published
    - Note that at key retrieval, ciphertext is **sent in the clear from passport to reader over the air**, so reader (and its environment!) is trusted

Crypto

- Most passports use
  - ECDH with a variety of curves (brainpool320r1 in NL)
  - AES-256-CBC (or AES-128)
- Some passports still use RSA based DH and 3DES
  - We did not implement support for those documents, but RDE does work

- Brainpool320r1 with AES-256-CBC results in 160 bit security for our final secret key (Verheul, 2017)
  - Note that ciphertexts are at most 255 bytes long, with 223 bytes for data

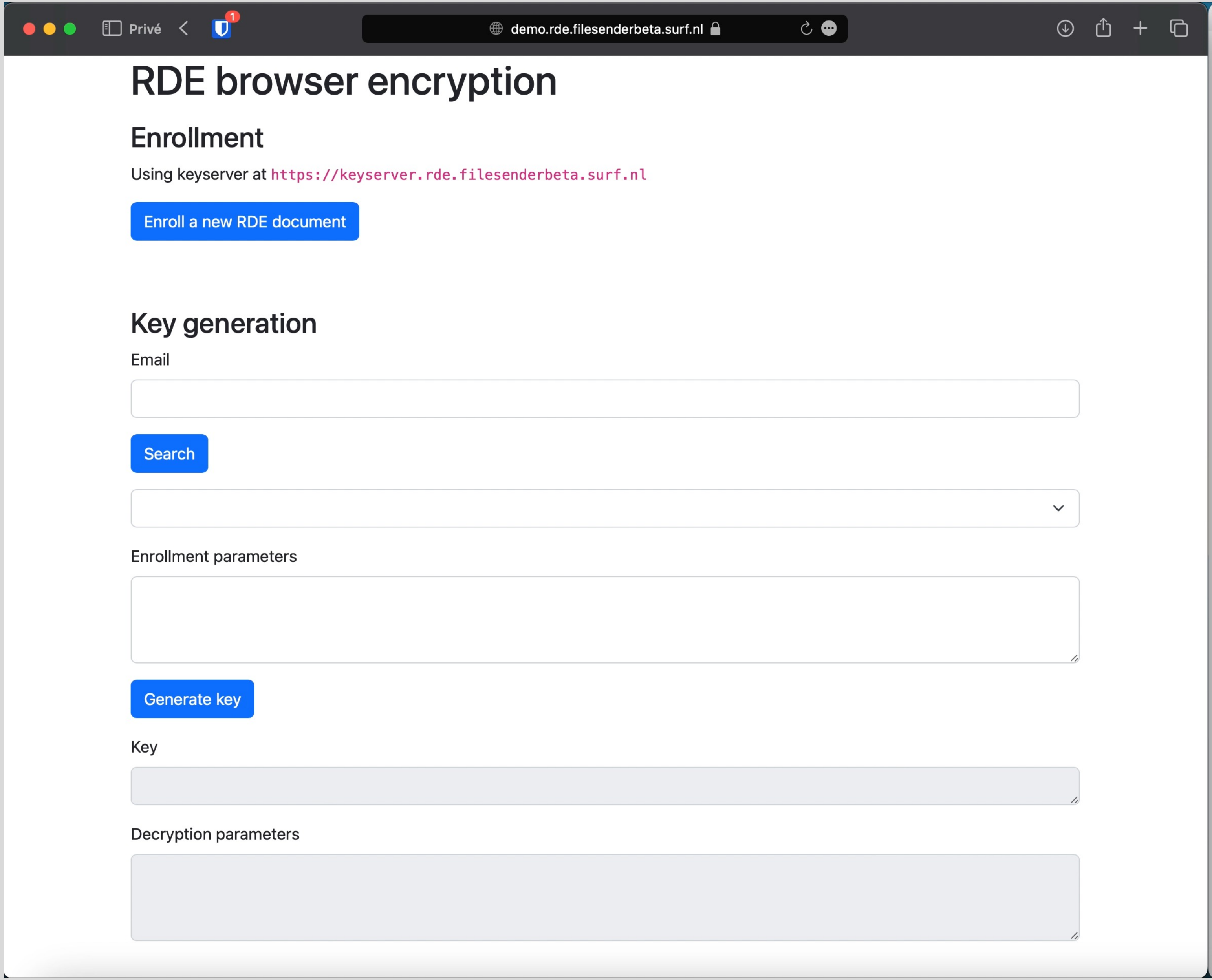
Crypto dependencies

- TypeScript (JavaScript) library
  - @peculiar/x509
  - indutny/elliptic (for ECC on arbitrary curves)
  - indutny/hash.js
  - rosek86/aes-cmac (for AES-CMAC)
  - leonardodino/aes-ts (for AES-CBC and AES-ECB with no padding)
- Note that WebCrypto API cannot be used, because it has limited support for curves and no AES modes

- Kotlin (Java) library
  - JMRTD
  - BouncyCastle

Links

- Demo, source code and report: https://demo.rde.filesenderbeta.surf.nl
- Paper E. Verheul (2017): https://arxiv.org/abs/1704.05647



RDE browser encryption

Enrollment

Using keyserver at `https://keyserver.rde.filesenderbeta.surf.nl`

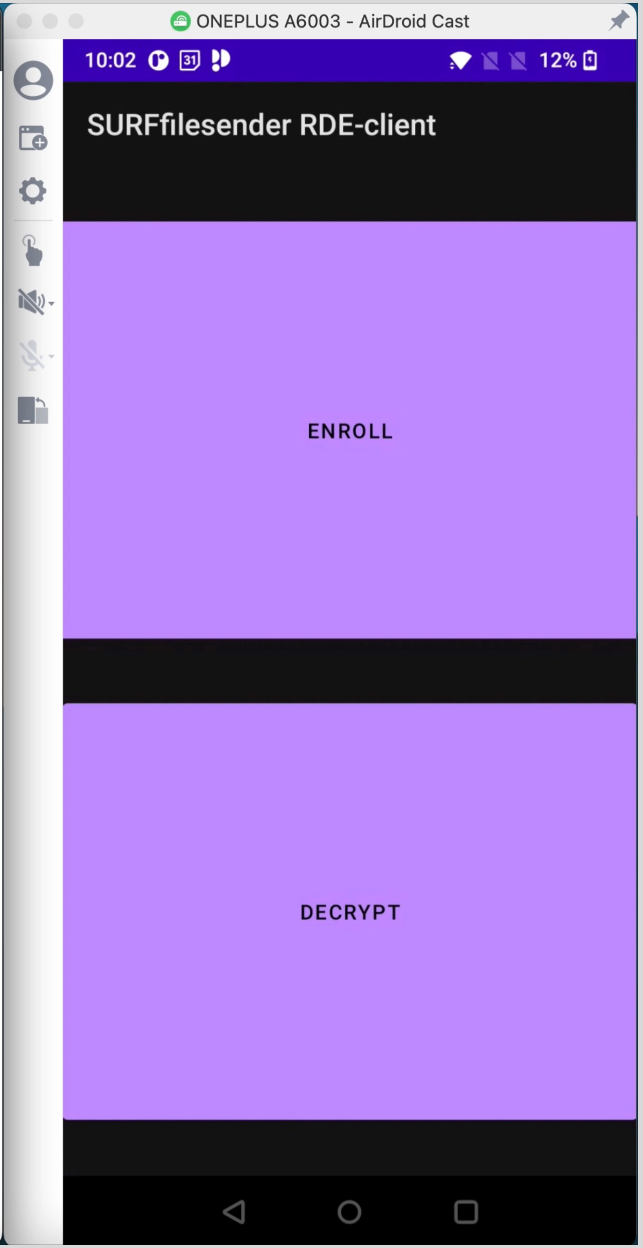
Key generation

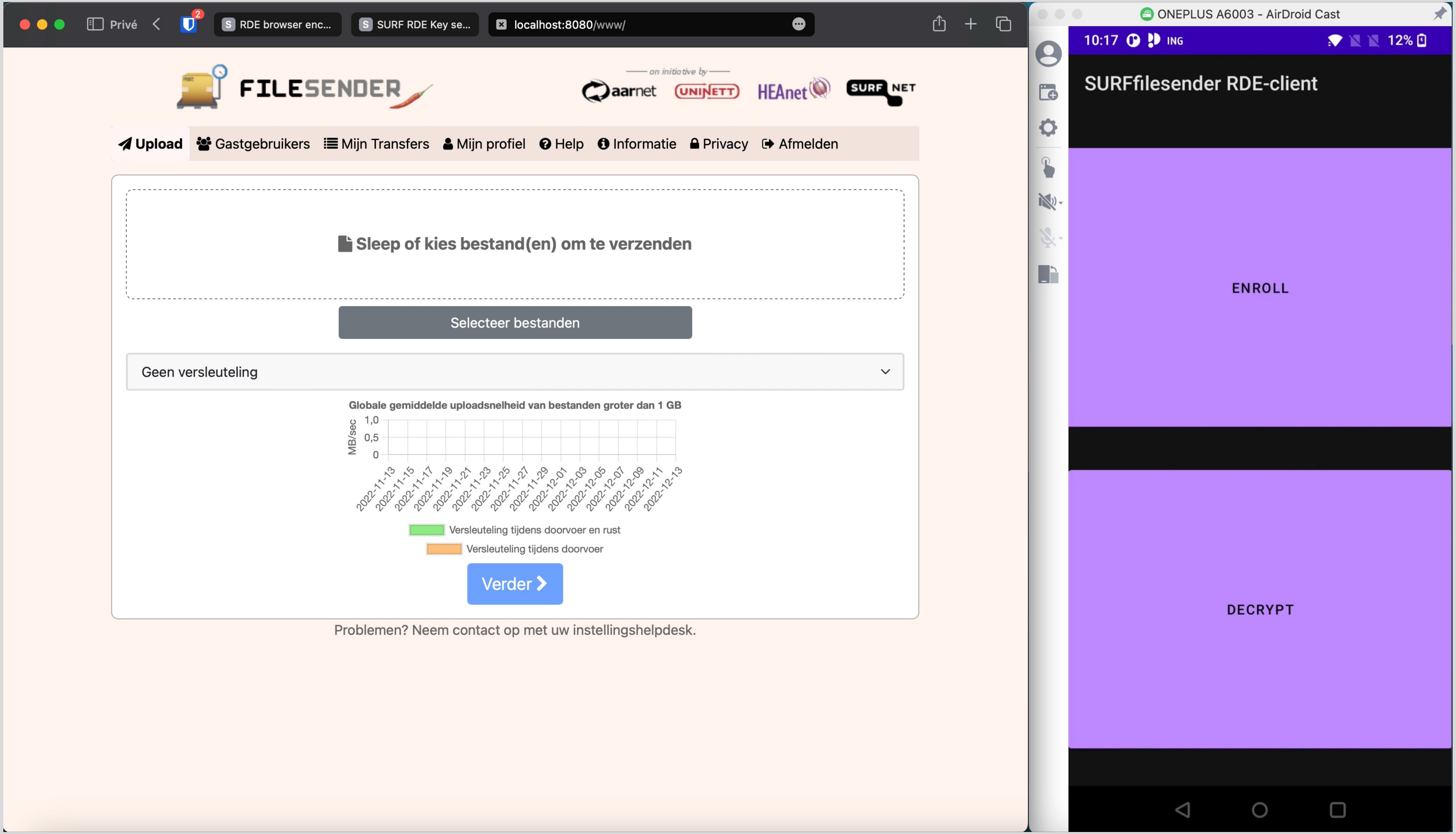
Email

Enrollment parameters

Key

Decryption parameters





Sleep of kies bestand(en) om te verzenden

Geen versleuteling

Problemen? Neem contact op met uw instellingshelpdesk.