1

Remote Document Encryption - encrypting data for passport holders

Eric.Verheul@keycontrols.nl
KeyControls, Radboud Universiteit Nijmegen

NLUUG Najaarsconferentie 2019
21 November 2019

(*) Research done for Dutch Vehicle Authority (RDW) based on a question from Gert Maneschijn.

Paper: https://arxiv.org/abs/1704.05647

1

1

2

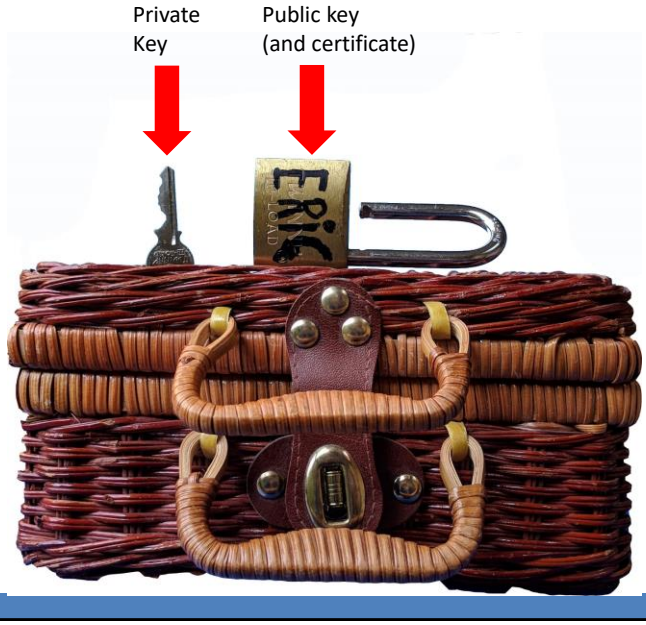
Agenda

- Public key cryptography
- Passport technology introduction
- RDE outline
- Example RDE applications
- RDE application in (SURF) FileSender
- Implementation of RDE in (SURF) FileSender
- Conclusion

2

2

Public key cryptography

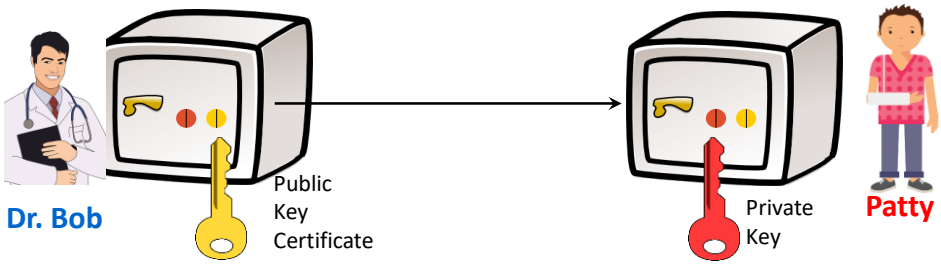


3

3

Public key cryptography

- One key to *close* the safe (*public key*) and another key to open the safe (*private key*).
- Patient **Patty** publishes her public key allowing **dr. Bob** to get hold of it.
- Patient **Patty** keeps her private key secret.
- Dr. **Bob** encrypts data for **Patty** with her public key; only **Patty** can decrypt this with her private key.
- To be sure that the public key really belongs to **Patty**, this key is associated with **Patty**'s identity and signed by a "TTP": a **Public Key certificate**



4

4

Passport technology introduction

5

5

Passport technology introduction

- In essence an identity document (passport) is a contactless USB flash drive with some files on it.
- Several mobile APPs allow you to read the contents of your passport, identity card and driving license
- Since September (IOS13) also on Apple devices!

For all (technical) details see https://www.icao.int/publications/pages/publication.aspx?docnum=9303

6

6

Passport technology introduction

- Data Group 1 (file): Name, date of birth, …
- Data Group 2 (file): Facial image
- Data Group 3 (file): Finger prints
- ….
- Data Group 14 (file): CA (anti-cloning) Public Key — **"RDE public key"**
- Data Group 14 (file): AA (anti-cloning) Public Key
- "EF.SOD" (file) **Public Key Certificates**

Passport technology is based on advanced cryptography

For all (technical) details see https://www.icao.int/publications/pages/publication.aspx?docnum=9303 7

7

Passport technology introduction

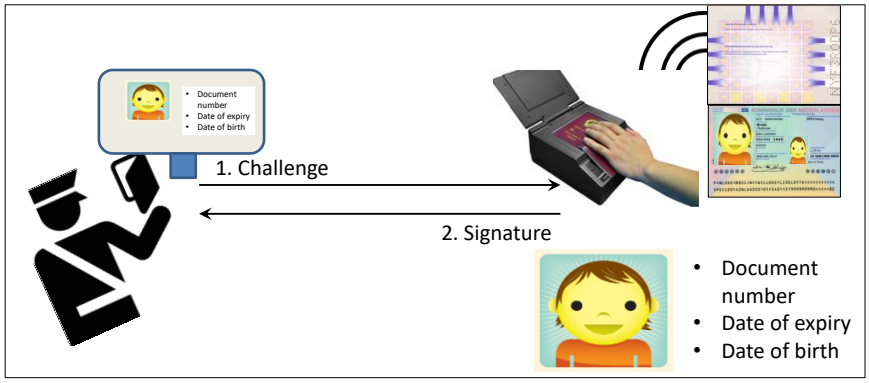
- Access control to the data on an *e-passport* is restricted by Basic Access Control (BAC): only with the Machine Readable Zone (MRZ) one can read information from the passport. The MRZ is used as a shared (3DES) cryptographic key to authenticate the reader and to set up a secure channel.

8

8

Passport technology introduction

- To counter look-alike fraud, *e-passports* can show they are authentic, i.e. not cloned. This is typically done by letting a passport sign random challenges sent by the reader.
- This protocol is called Active Authentication (AA).



9

9

Passport technology introduction

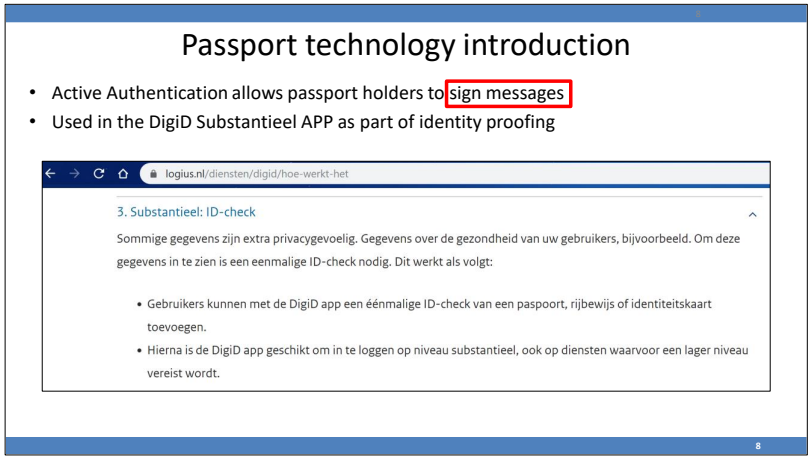
- Active Authentication allows passport holders to sign messages
- Used in the DigiD Substantieel APP as part of identity proofing

10

10

11

RDE introduction



- Question of Gert Maneschijn (RDW): would it be possible to let a passport decrypt data?
- The answer is surprisingly 'yes'.

11

11

12

RDE introduction: *crux*

- Remote Document Encryption (RDE) is a tweak on passport protocols.
- It allows any party to encrypt data for the holder of an electronic passport such that:
- Decryption is only possible with physical possession of the document and takes place *inside* the document, typically by the holder.
- RDE allows for 160 bit security on European identity documents where 128 bit is current good practice, i.e. RDE is $2^{32} \approx 4$ billion times stronger.

Illustrative application

A hospital wants to send its patients (RDE) encrypted e-mails. The hospital develops an RDE mobile APP allowing:

a. the hospital reading a RDE "public key certificate" from the identity document for e-mail encryption,
b. the patient to perform RDE decryption using a PC/mobile device together with her identity document.

12

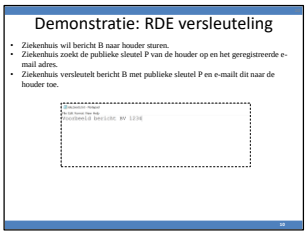
12

13

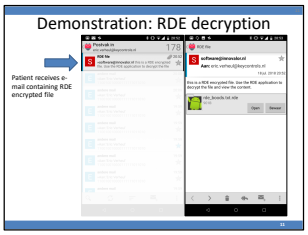
Demonstration ‘RDE secure mail’

The ‘RDE secure email’ use-case consists of three phases.

1. User registration (one-time only)



2. Encryption of message by hospital



3. Decryption by patient

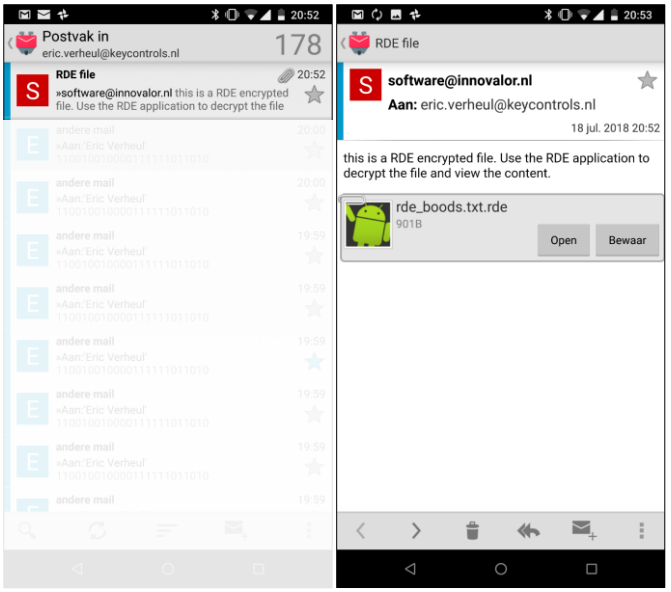
13

13

14

Demonstration: RDE decryption

Patient receives e-mail containing RDE encrypted file



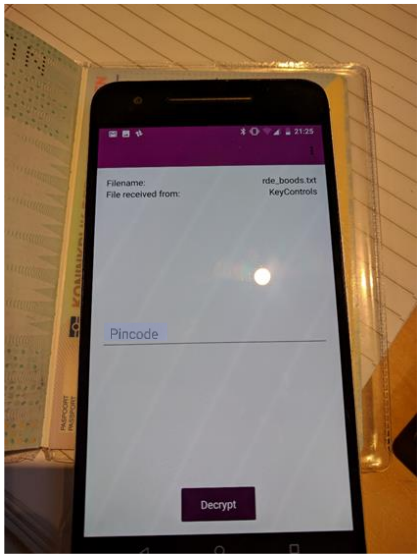
14

14

15

Demonstration: RDE decryption

Automatic Start-up of RDE APP + PIN entry



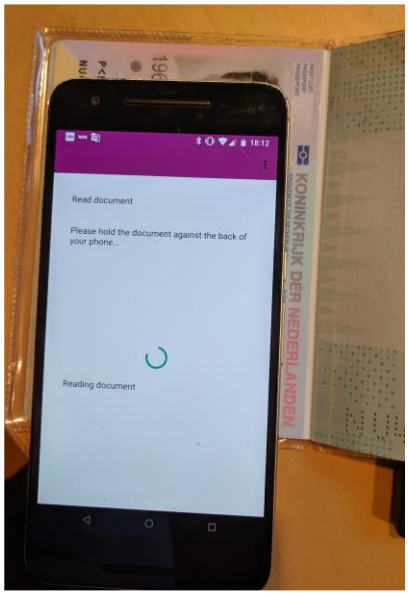
15

15

16

Demonstration: RDE decryption

RDE decryption

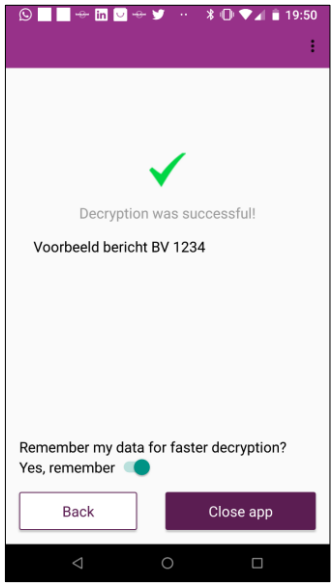


16

16

Demonstration: RDE decryption

RDE plaintext



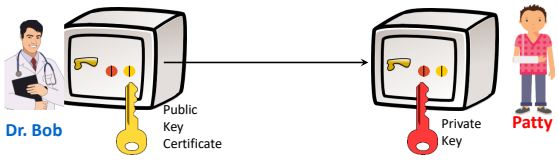
17

17

Outline RDE (laymen)

Public key cryptography

- One key to *close* the safe (*public key*) and another key to open the safe (*private key*).
- Patient **Patty** publishes her public key allowing **dr. Bob** to get hold of it.
- Patient **Patty** keeps her private key secret.
- Dr. **Bob** encrypts data for **Patty** with her public key; only **Patty** can decrypt this with her private key.
- To be sure that the public key really belongs to **Patty**, this key is associated with **Patty**'s identity and signed by a "TTP": a **Public Key certificate**



Hospital

Patient

- RDE Public Key certificate is bound to all printed information:
  - First and last name
  - Date of birth
  - Place of birth
  - Facial image (in colour!)

18

18

19

RDE registration

Public Key +
"certificate" +
email address

PubK

"Registration"

Private
Key ***x***

Dr. Bob
(sender)

Patty
(receiver)

19

19

RDE encryption ideally

Public Key +
"certificate"

PubK

1. ***E***=ENC_P (**K**)
   ENC_K (Message)

2. Email ***E*** + ENC_K (Message)

Private
Key ***x***

3. ***E***

4. K

Dr. Bob
(Sender)

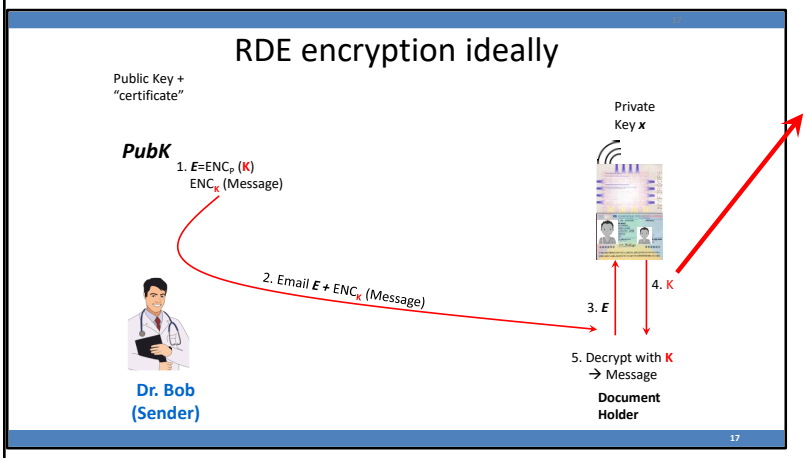
5. Decrypt with **K**
→ Message

Document
Holder

20

20

RDE encryption

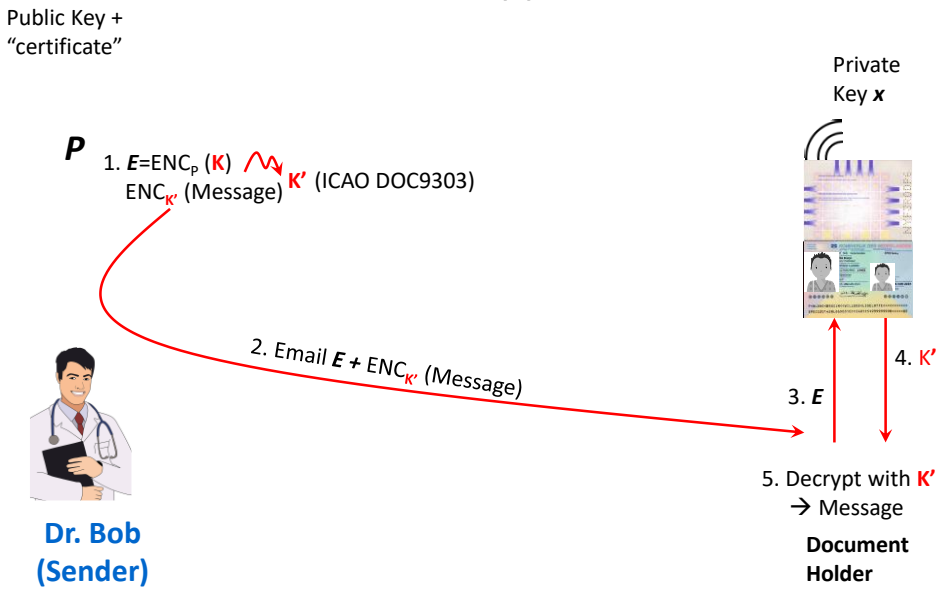


Unfortunately: passport delivers *different* K than the original
But: sending party can predict what the holder passport will deliver, namely ***K'***.
RDE Crux: let sending party and holder use *that* key, i.e. ***K'***, as encryption key.

21

21

RDE encryption!

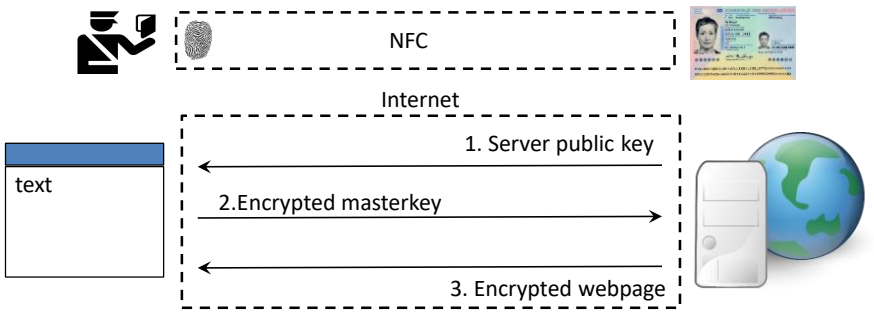


22

22

RDE explanation using TLS metaphore

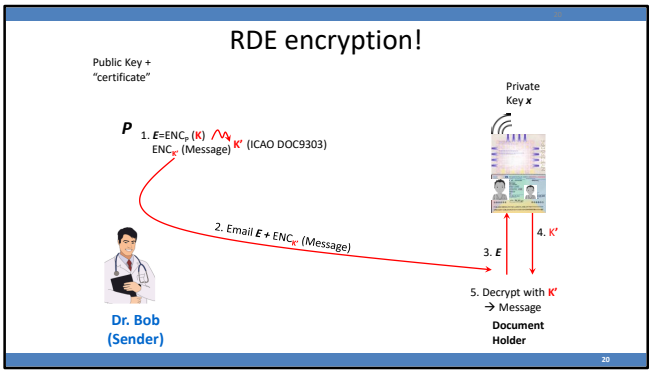
- As part of reading fingerprints by border control, the reader application needs to setup a Chip Authentication (CA) tunnel, similar to one-sided TLS.
- Suppose the masterkey only includes randomness of the reader (browser)
- Then the reader could precompute/predict the encrypted message in Step 3 and use that as a derived cryptographic key
- In RDE the derived cryptographic key is the precomputed/predicted encrypted contents of a known file (webpage in the metaphore)



23

23

RDE PIN (two factor encryption)



- By additionally encrypting ENC_P (**K**) with a *Personal Encryption Number* (PEN) one gets two factor encryption (possession and knowledge)
- PEN can only be *brute forced* with the passport!
- By making PEN 'long enough' the brute force risk is controllable.
- PEN is an interesting intermediary form of PIN and password.

24

24

25

Example RDE applications

- **Secure email**
  RDE encrypt messages and send them through email. Already tested in pilot in 2018 with a developed APP.
- **Secure password managers, e.g. Keepass**
  Weak spot is the encryption of the password database. In practice this encryption is based on a guessable password making cloud archiving a bad idea. With RDE the password database can be adequately encrypted, allowing secure cloud archiving.
- **Hardware based disk encryption**
  Weak in disk encryption is the encryption key which is typically manually entered during booting. With RDE the key can be derived from the passport during booting through NFC.
- **Secure personal health environments**
  Within Dutch healthcare it is facilitated that patients can have their medical records sent from their healthcare provider to a Personal Health Environment (PHE). With RDE the healthcare provider can encrypt the data ensuring that only the patient has access to them (and not the PHE).
- **End-to-end secure SURF FileSender (next slides)**
  See surffilesender.nl. SURF intents to implement RDE in its Filesender instance in a 2019 pilot. This pilot will be done in cooperation with Dutch government (RDW and RvIG).

25

25

26

RDE application: end-to-end secure SURF FileSender

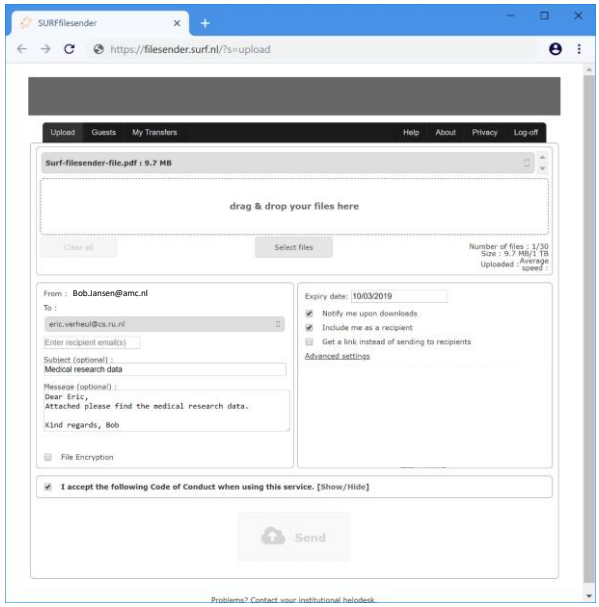
- FileSender:
  - is an open source, web based application for exchanging large files, see https://filesender.org/
  - currently supports the use of a password (decryption key) to encrypt the exchanged files.
  - encryption en decryption takes place in the browser (W3C Web Cryptography API JavaScript) providing end-to-end security.
- However, the (classical) problem is the password exchange among users in a secure and user-friendly way.
- To address this problem, SURF and Dutch government (RDW, RvIG) intent to supplement RDE to the FileSender open source project.
- In this way FileSender can conveniently support the secure exchange of personal data in education, research, healthcare. It also can facilitate the GDPR 'right of access' at institutions.

26

26

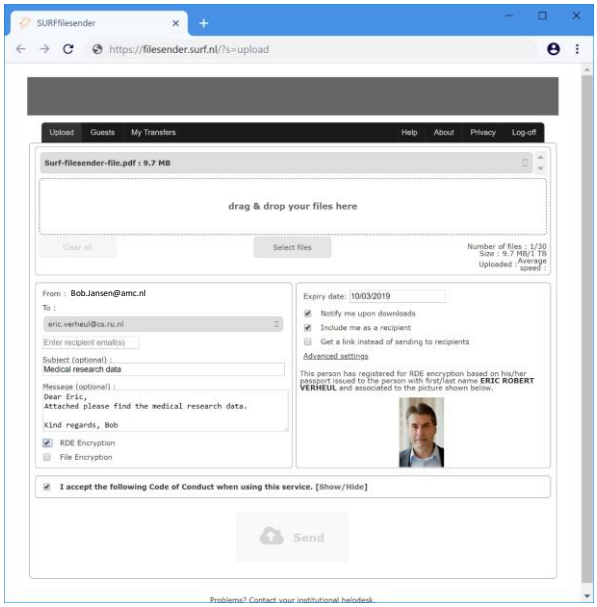
RDE application: end-to-end secure SURF FileSender

Current Interface Sender



RDE application: end-to-end secure SURF FileSender

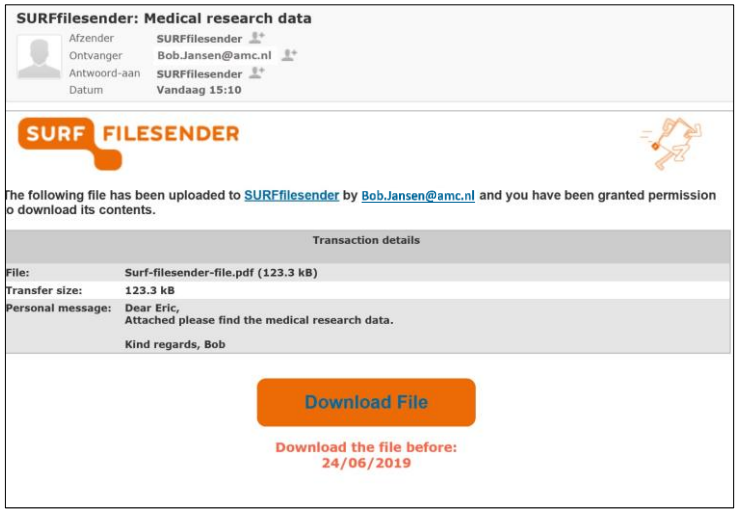
RDE Interface Sender



29

RDE application: end-to-end secure SURF FileSender

Notification Email

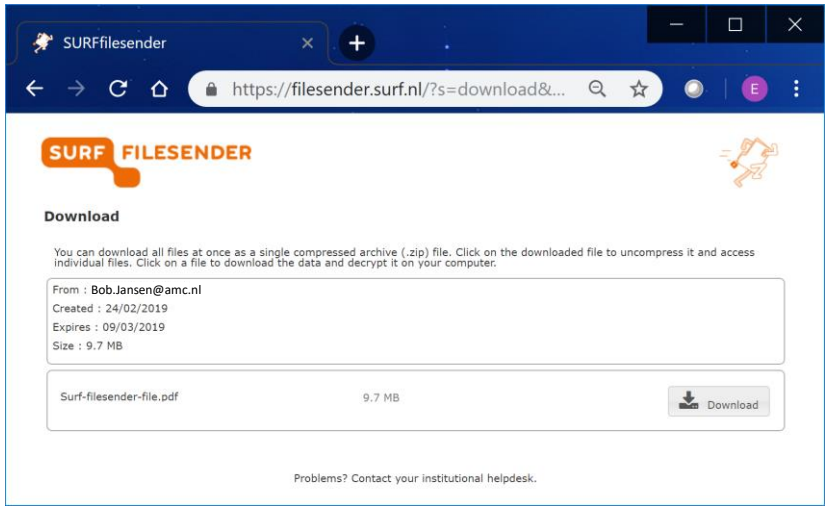


Encryption Mock-up

29

30

RDE application: end-to-end secure SURF FileSender



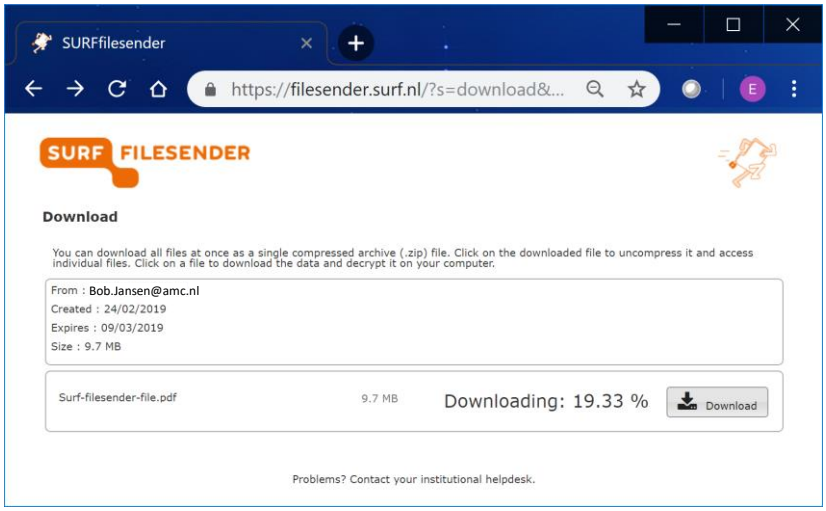
RDE Interface Receiver

Decryptie Mock-up

30

31

RDE application: end-to-end secure SURF FileSender

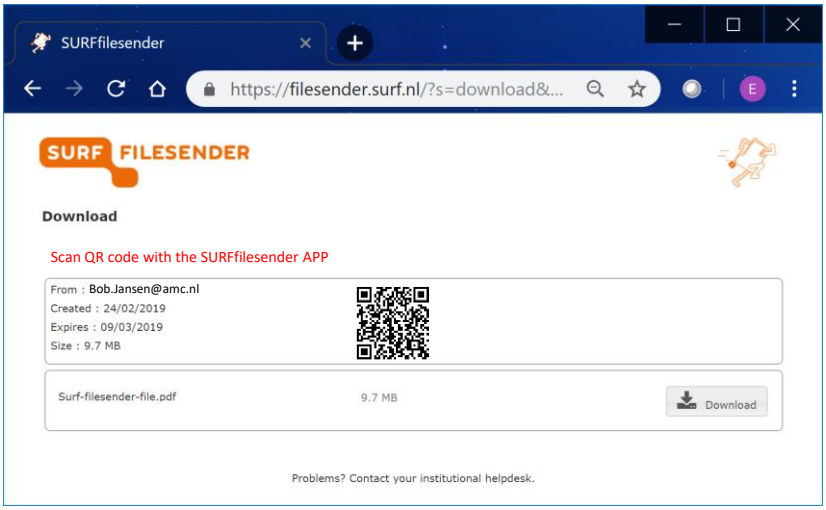


RDE Interface Receiver

31

32

RDE application: end-to-end secure SURF FileSender

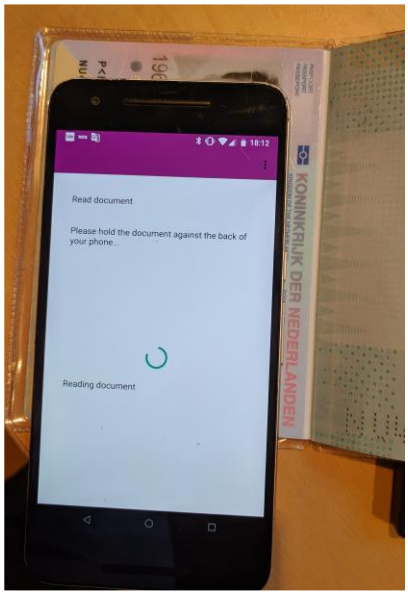


RDE Interface Receiver

32

33

RDE application: end-to-end secure SURF FileSender

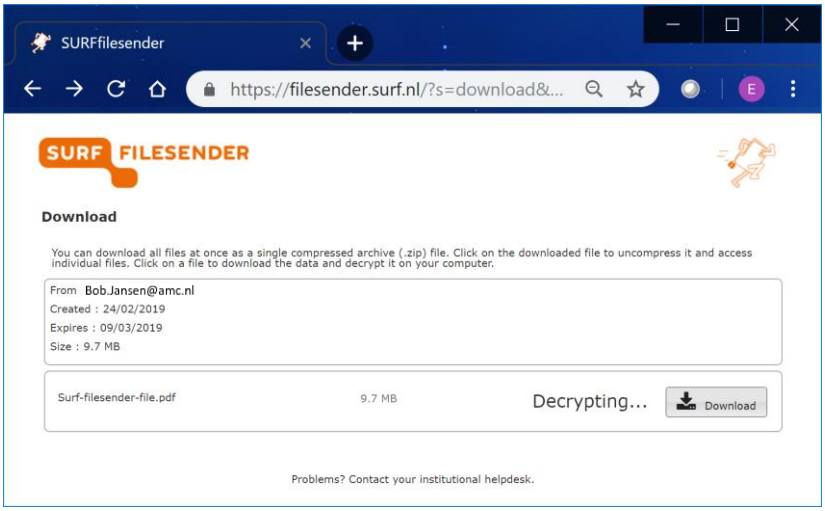


Decryptie Mock-up 33

33

34

RDE application: end-to-end secure SURF FileSender



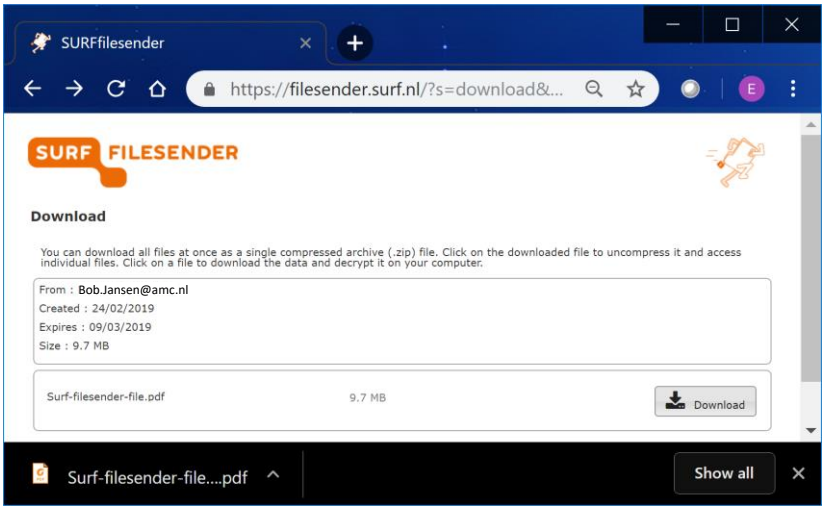
RDE Interface Receiver

Decryptie Mock-up 34

34

35

RDE application: end-to-end secure SURF FileSender



RDE Interface Receiver

Decryptie Mock-up

35

35

36

Implementation of RDE in SURF FileSender

- Cryptographic basis is symmetrical encryption based on AES in GCM mode. The AES-GCM operations take place within the internet browsers of the users.
- This is completely supported by the W3C Web Cryptography API (https://www.w3.org/TR/WebCryptoAPI/): AES-GCM can be called *natively* through JavaScript without requiring extra JavaScript *libraries*.
- By using a suitable AES-GCM configuration, encryption and decryption in (large) chunks is also easily possible; this is relevant for very large files.
- For the essential part of RDE (ECDH) support for so-called Brainpool elliptic curves is required. Alas W3C only supports NIST curves as P-256.
- It is therefore required that Brainpool based ECDH is separately implemented, e.g. by limited use of the Stanford Javascript Crypto Library (http://bitwiseshiftleft.github.io/sjcl/).
- RDE chunk based setup also usable for current password based Filesender, allowing sending very large file (>>2GB).

36

36

Conclusion