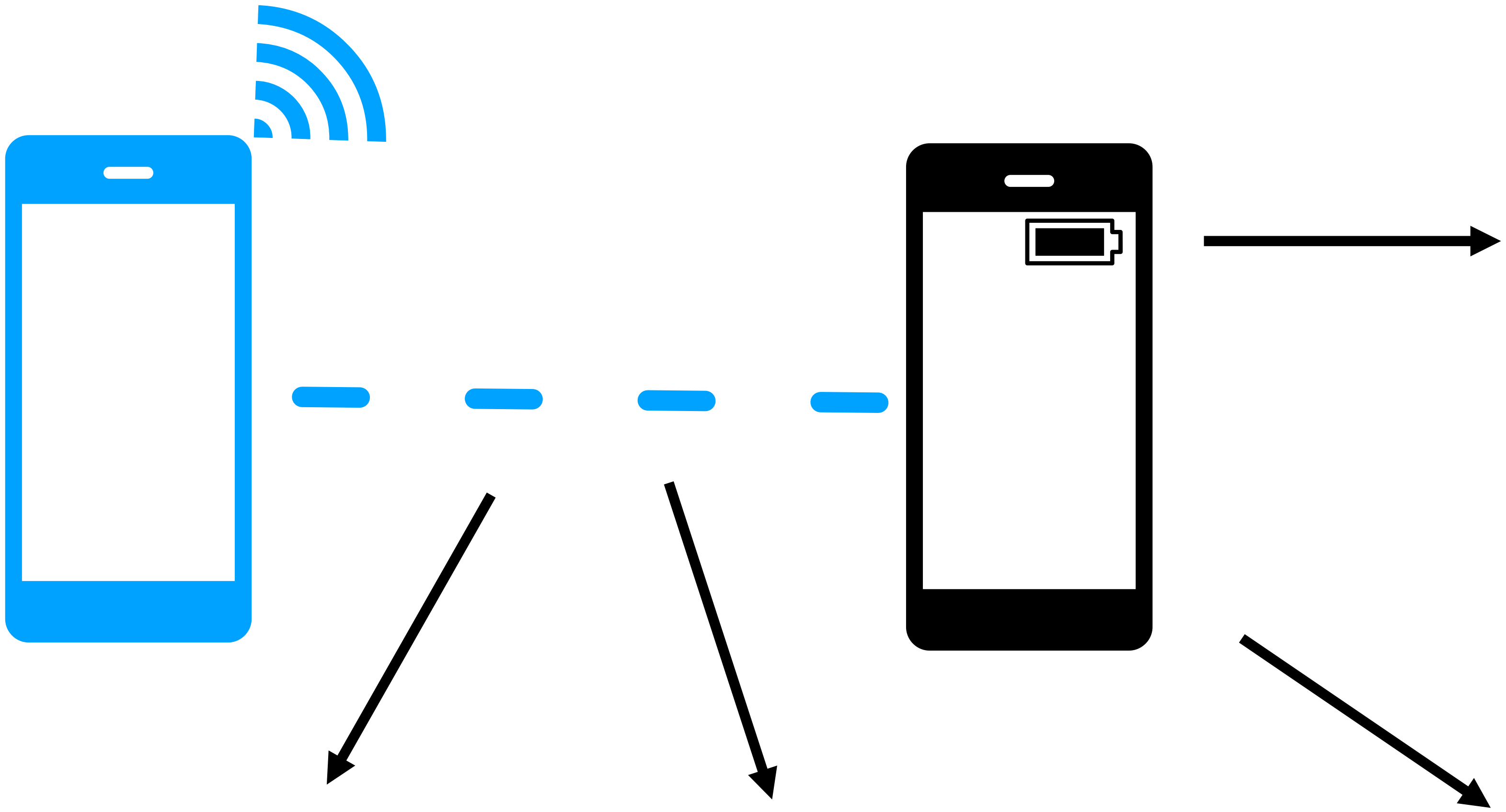
ContacTUM consortium 07.05.2020

Improving decentralized Digital Contact Tracing System (DCTS) & How to keep your privacy

Advantages of Bluetooth



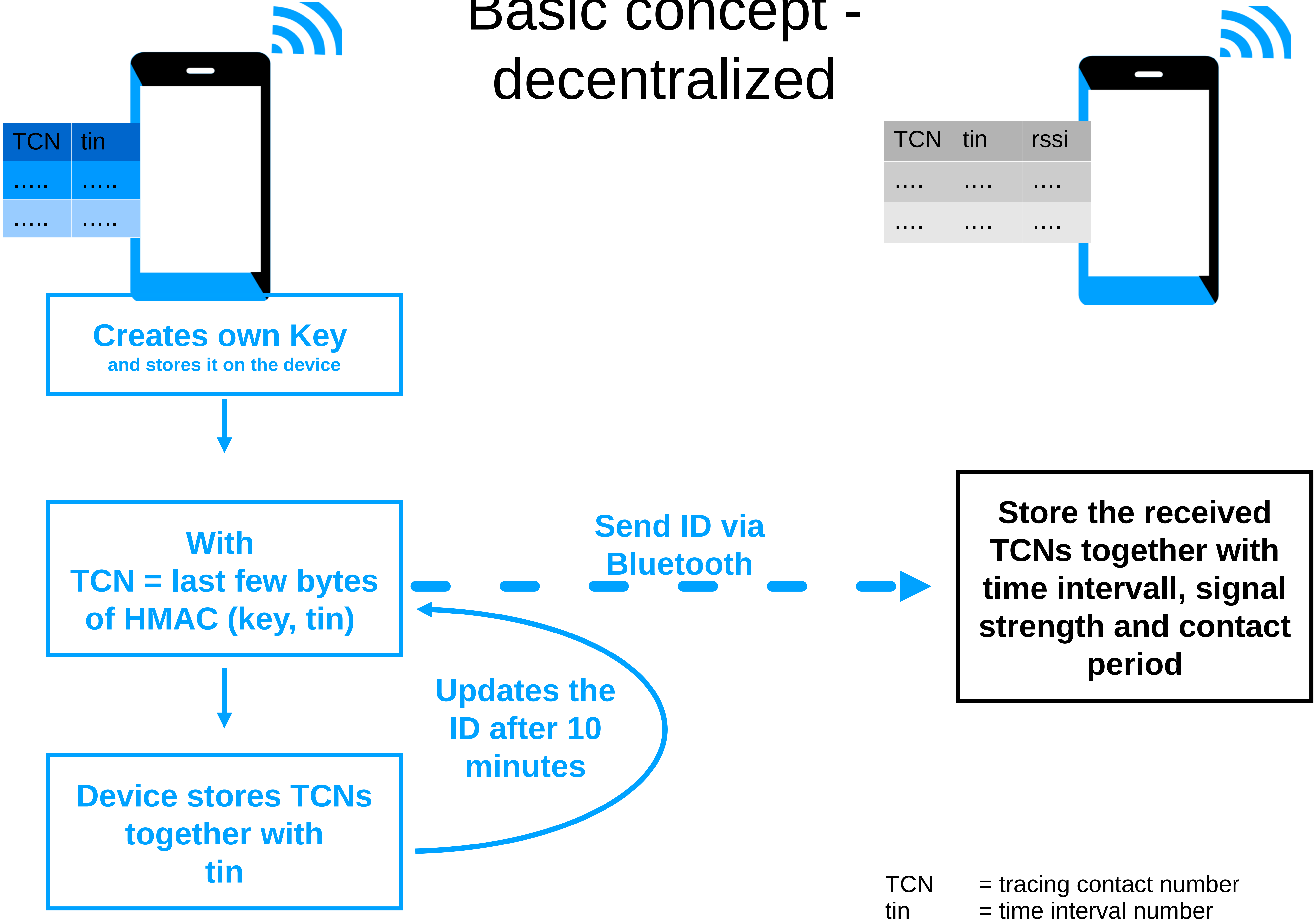
Low power consumption (respect to GPS/WIFI)

Measure only real encounters below 10m (also in buildings, subway, ...)

For Android ≥ 8: Android randomizes Mac addresses → no tracking without pairing

No absolute position is tracked – only vicinity

Basic concept - decentralized



TCN	tin
…..	…..
…..	…..

TCN	tin	rssi
….	….	….
….	….	….

Creates own Key
and stores it on the device

**With
TCN = last few bytes of HMAC (key, tin)**

Device stores TCNs together with tin

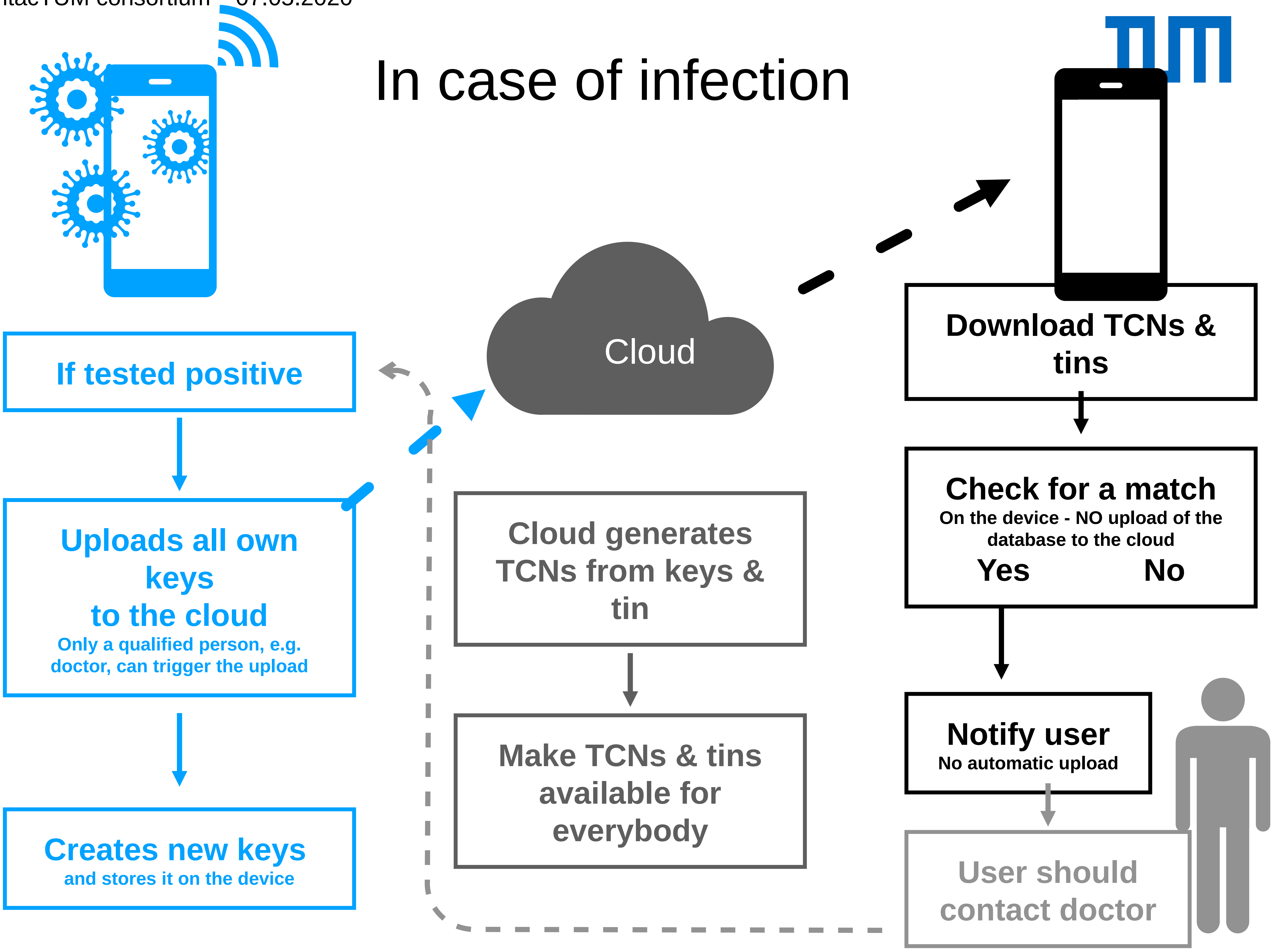
Updates the ID after 10 minutes

Send ID via Bluetooth

Store the received TCNs together with time intervall, signal strength and contact period

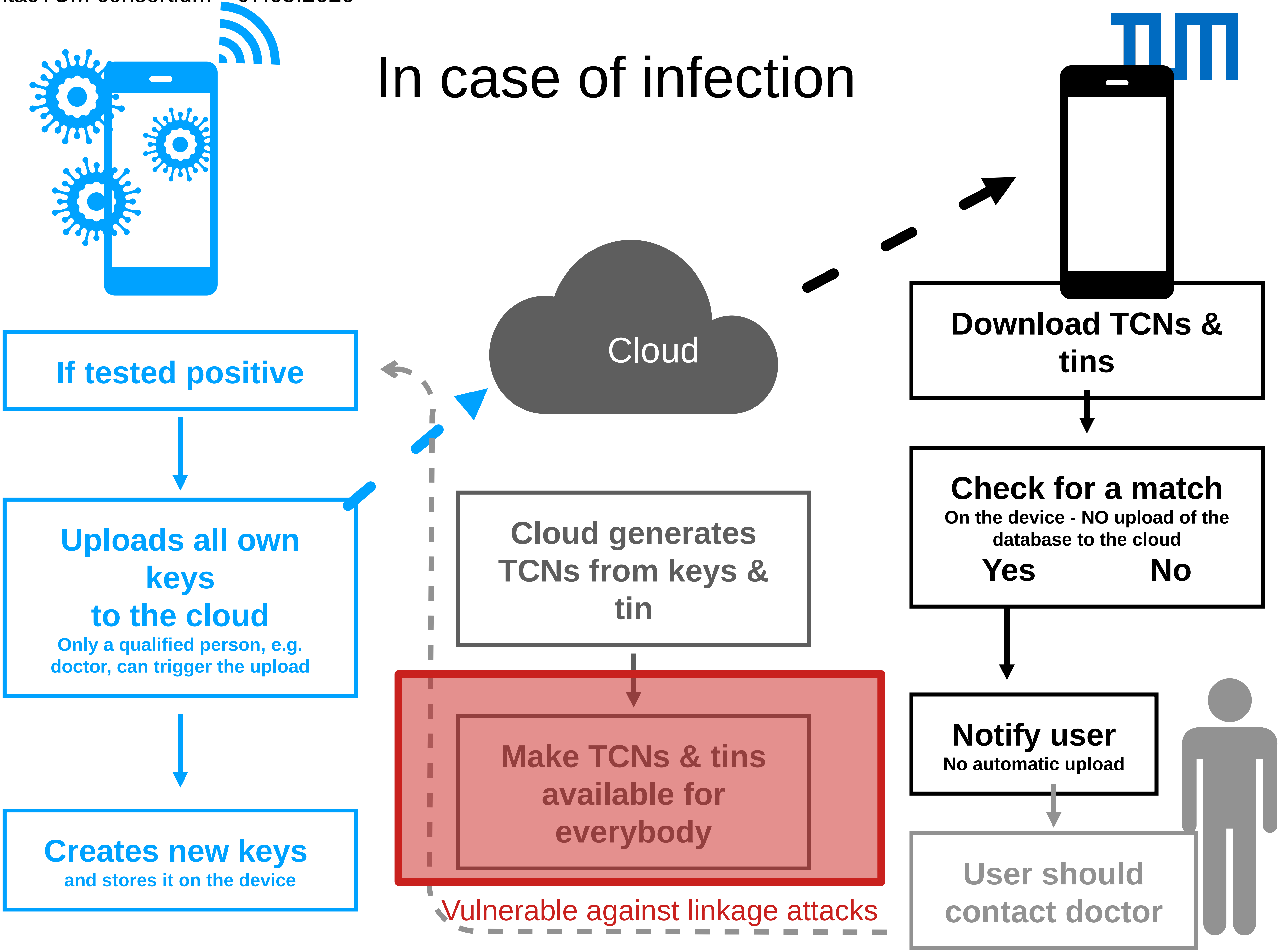
TCN = tracing contact number
tin = time interval number

In case of infection



If tested positive
Uploads all own keys to the cloud
Only a qualified person, e.g. doctor, can trigger the upload
Creates new keys
and stores it on the device
Cloud
Cloud generates TCNs from keys & tin
Make TCNs & tins available for everybody
Download TCNs & tins
Check for a match
On the device - NO upload of the database to the cloud
Yes
No
Notify user
No automatic upload
User should contact doctor

In case of infection



If tested positive

Uploads all own keys to the cloud
Only a qualified person, e.g. doctor, can trigger the upload

Creates new keys
and stores it on the device

Cloud

Cloud generates TCNs from keys & tin

Make TCNs & tins available for everybody

Vulnerable against linkage attacks

Download TCNs & tins

Check for a match
On the device - NO upload of the database to the cloud
Yes No

Notify user
No automatic upload

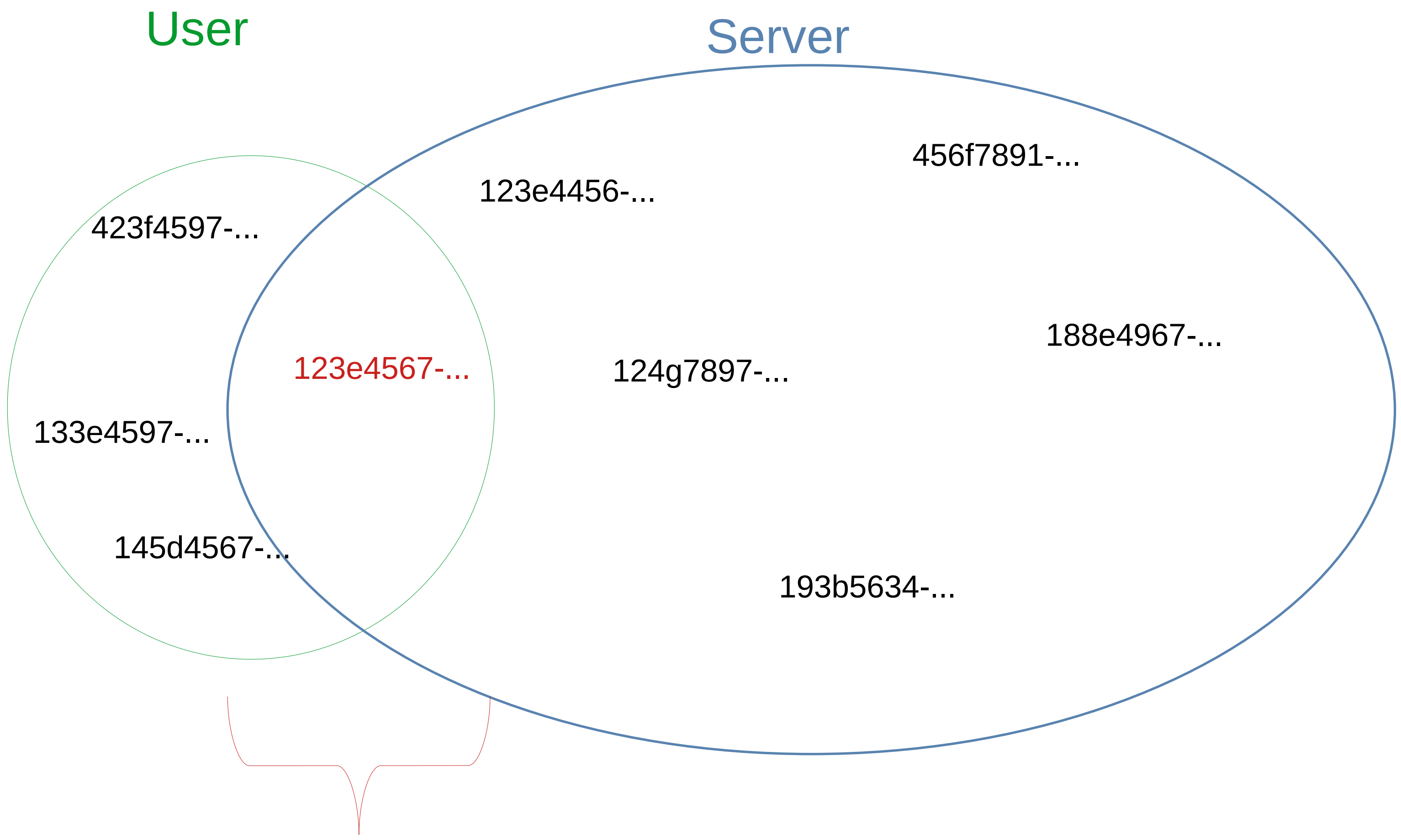
User should contact doctor

Interlude – private set intersection cardinality (PSI-CA)

How to protect the identities of infected people?

User: never gets to know the server's entries

Server: never gets to know the user's entries

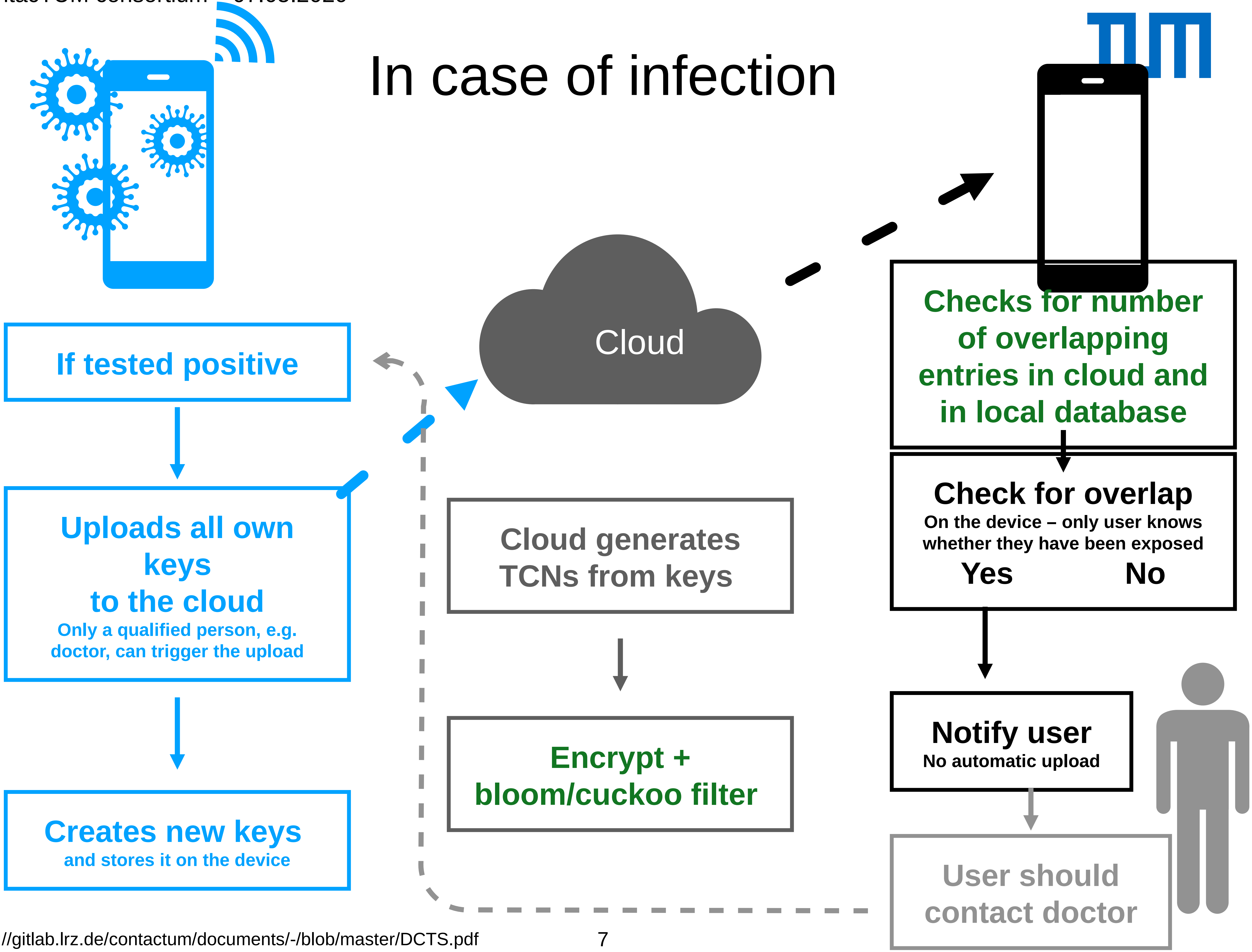


Number of entries in intersection:
1

User: knows the intersection cardinality

Server: never gets to know the intersection cardinality

In case of infection



If tested positive

Uploads all own keys to the cloud

Only a qualified person, e.g. doctor, can trigger the upload

Creates new keys

and stores it on the device

Cloud generates TCNs from keys

Encrypt + bloom/cuckoo filter

Checks for number of overlapping entries in cloud and in local database

Check for overlap

On the device – only user knows whether they have been exposed

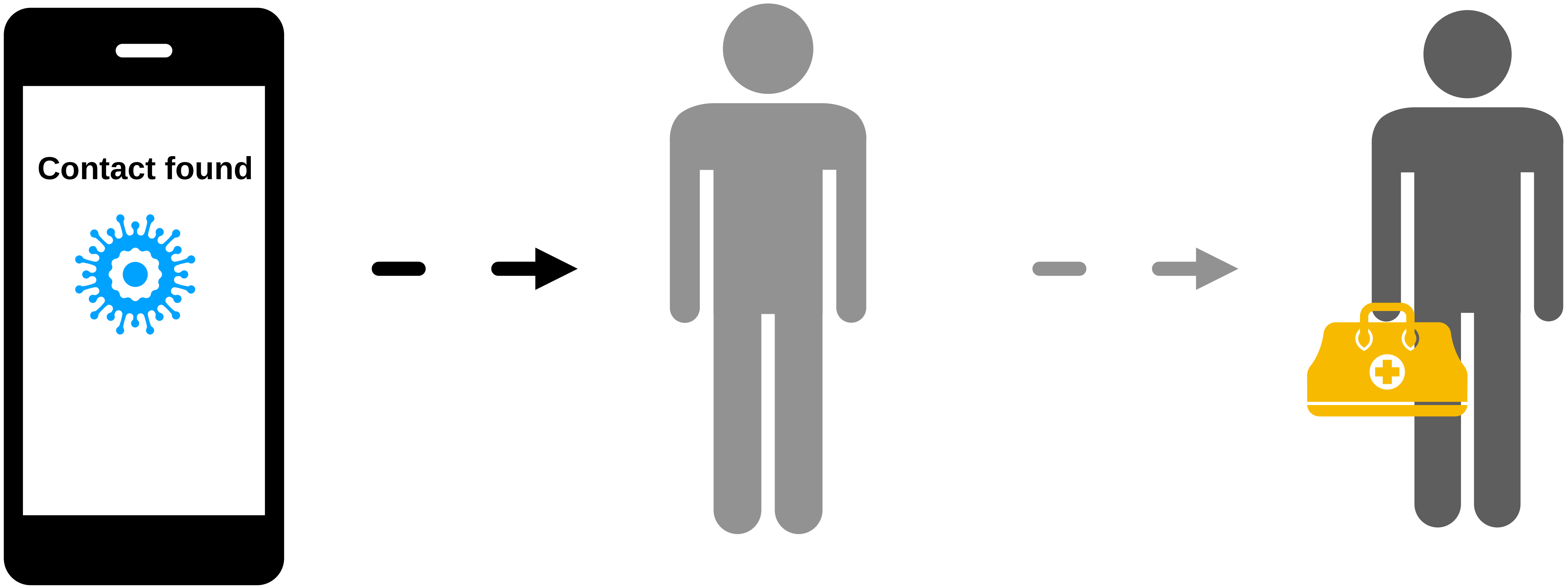
Yes No

Notify user

No automatic upload

User should contact doctor

User owns the data



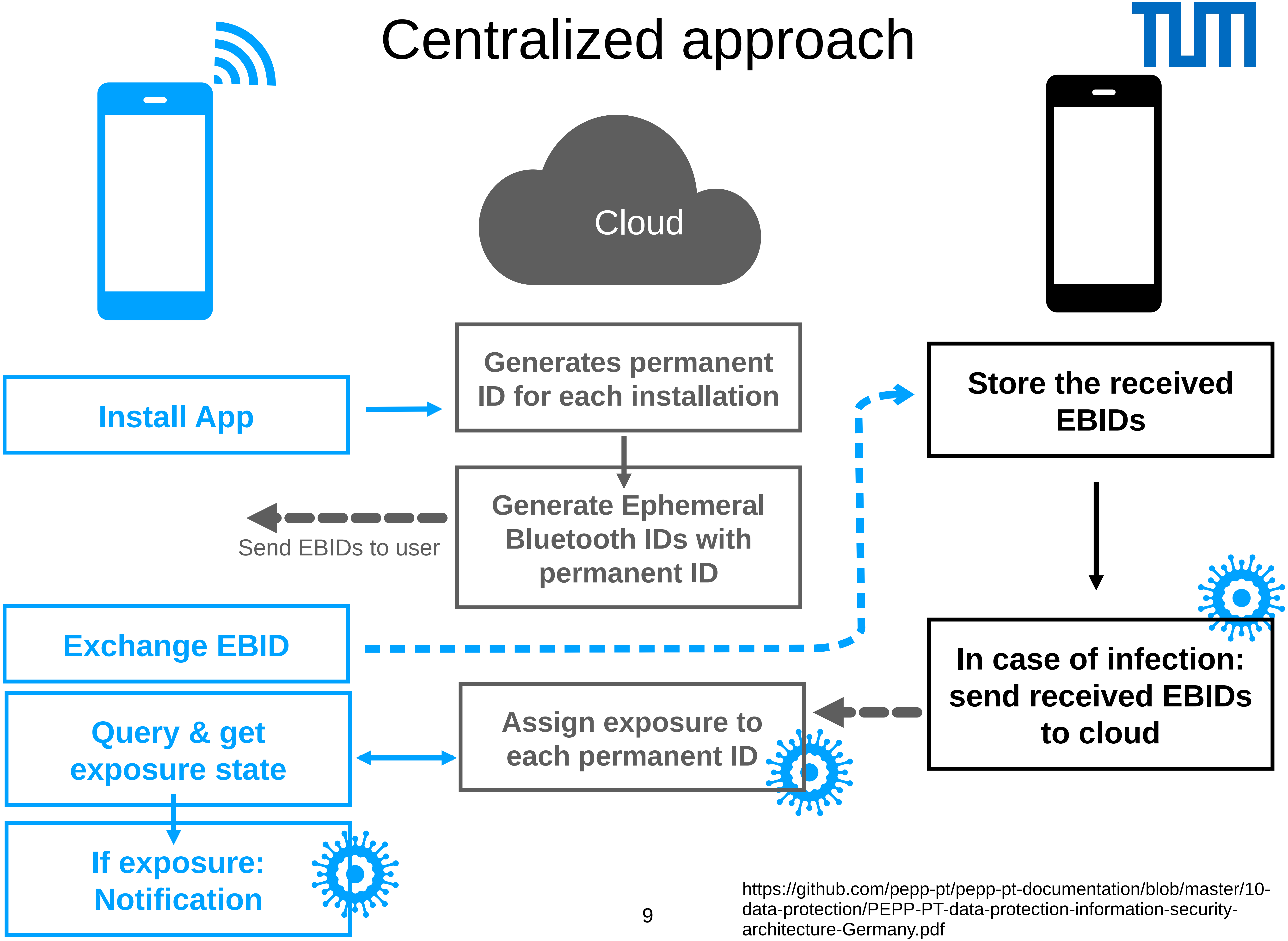
Analyses contact duration and signal strength

Value for the contact intensity

User can provide additional information

Decision for a test

Centralized approach



Cloud

Install App

Generates permanent ID for each installation

Generate Ephemeral Bluetooth IDs with permanent ID

Send EBIDs to user

Exchange EBID

Store the received EBIDs

In case of infection: send received EBIDs to cloud

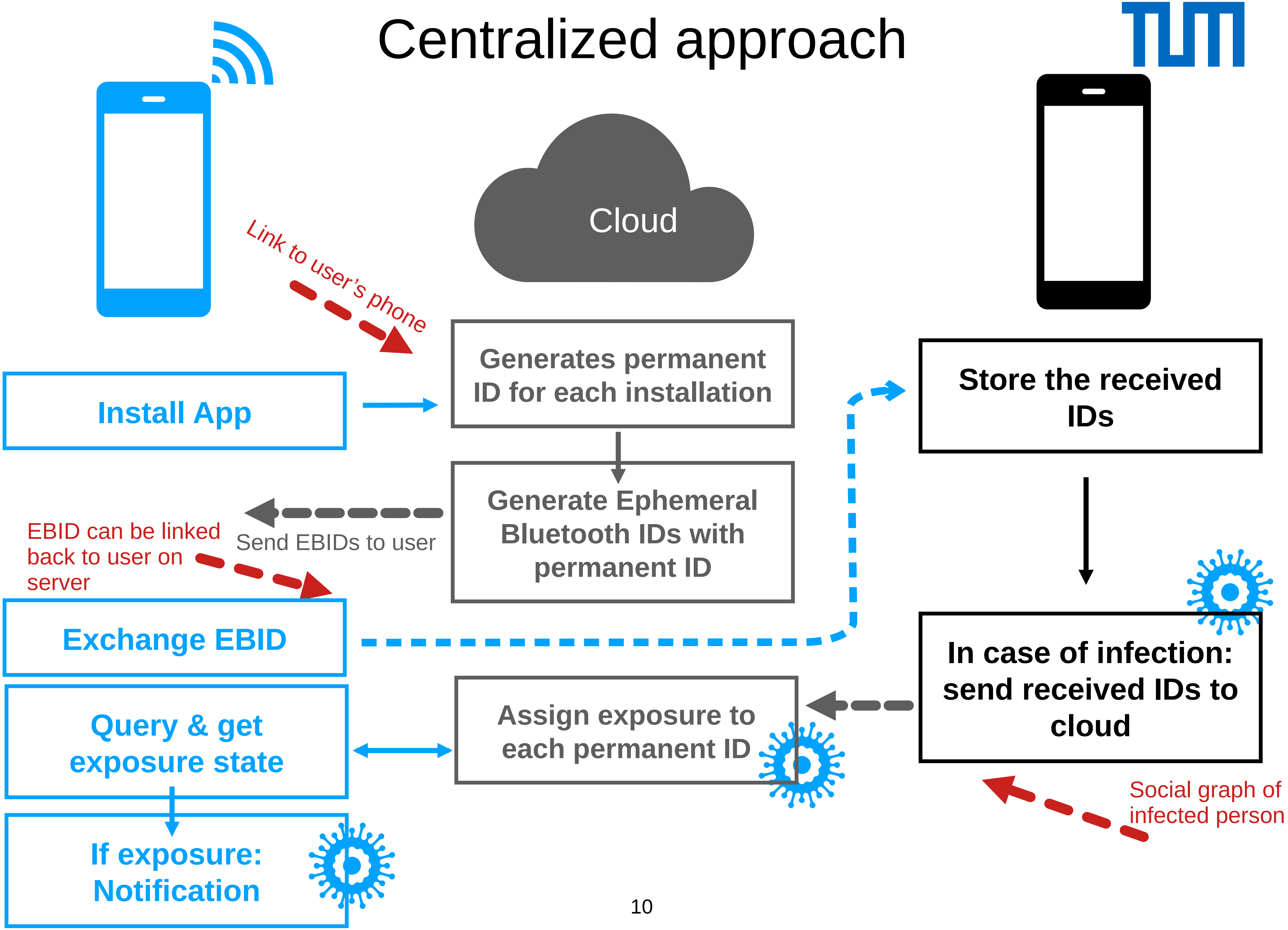
Assign exposure to each permanent ID

Query & get exposure state

If exposure: Notification

https://github.com/pepp-pt/pepp-pt-documentation/blob/master/10-data-protection/PEPP-PT-data-protection-information-security-architecture-Germany.pdf

Centralized approach



Cloud

Link to user's phone

Install App

Generates permanent ID for each installation

Generate Ephemeral Bluetooth IDs with permanent ID

Send EBIDs to user

EBID can be linked back to user on server

Exchange EBID

Store the received IDs

In case of infection: send received IDs to cloud

Assign exposure to each permanent ID

Query & get exposure state

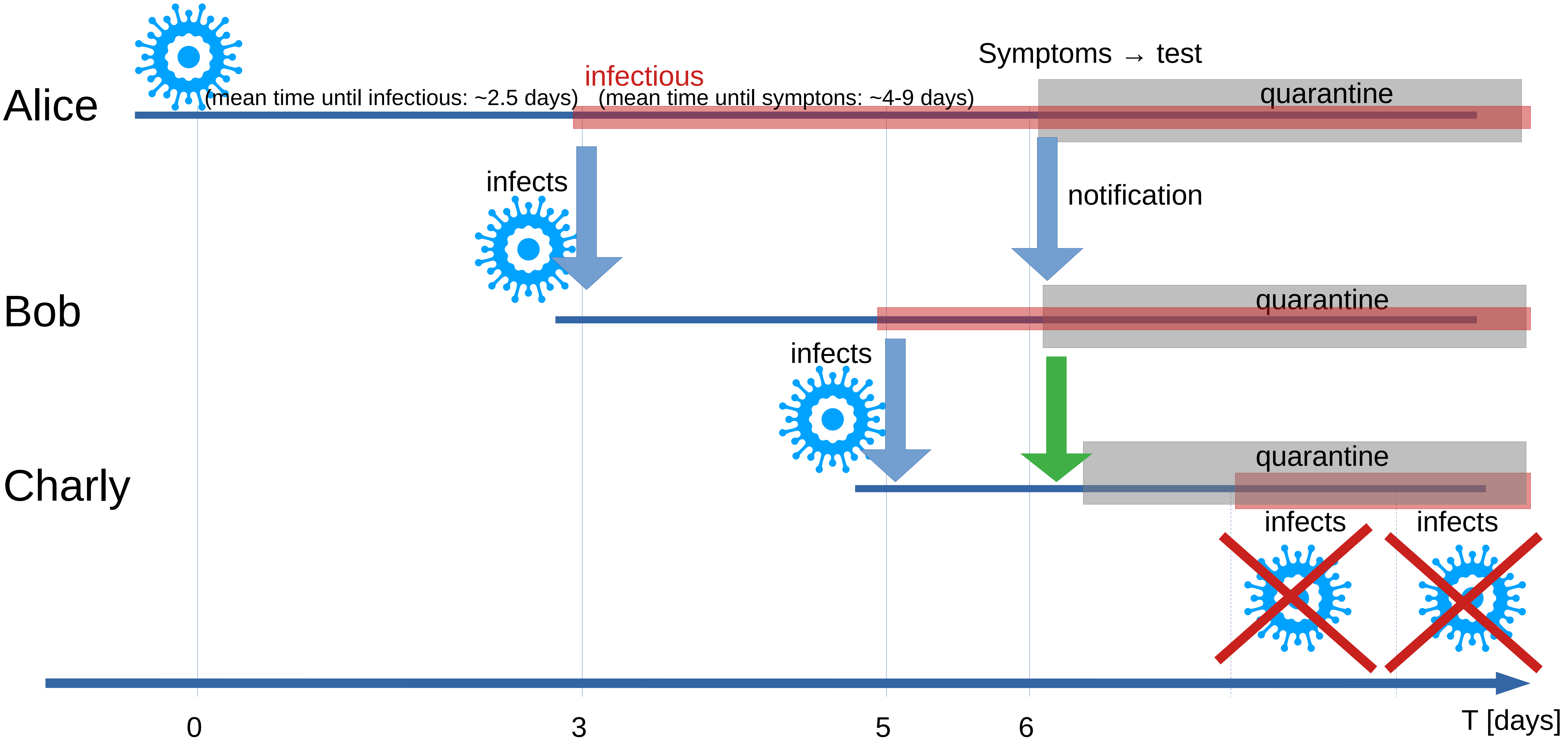
If exposure: Notification

Social graph of infected person

Centralized vs decentralized

Who knows what?	Centralized	Decentralized
Who did I see?	If infected: Server	Me
Who saw me?	If infected contact: Server	Me
Where have I been?	If I/contact infected:Server If infected: linkage attack	If infected: linkage attack (protection: PSI-CA)
Have I been exposed?	Server	Me
Who has been infected?	Server Linkage attack	Linkage attack (Protection with private set intersection cardinality (PSI-CA))
How many people have been infected?	Server Estimate with linkage attack	Estimate with linkage attack (Protection with PSI-CA)

Second order tracing

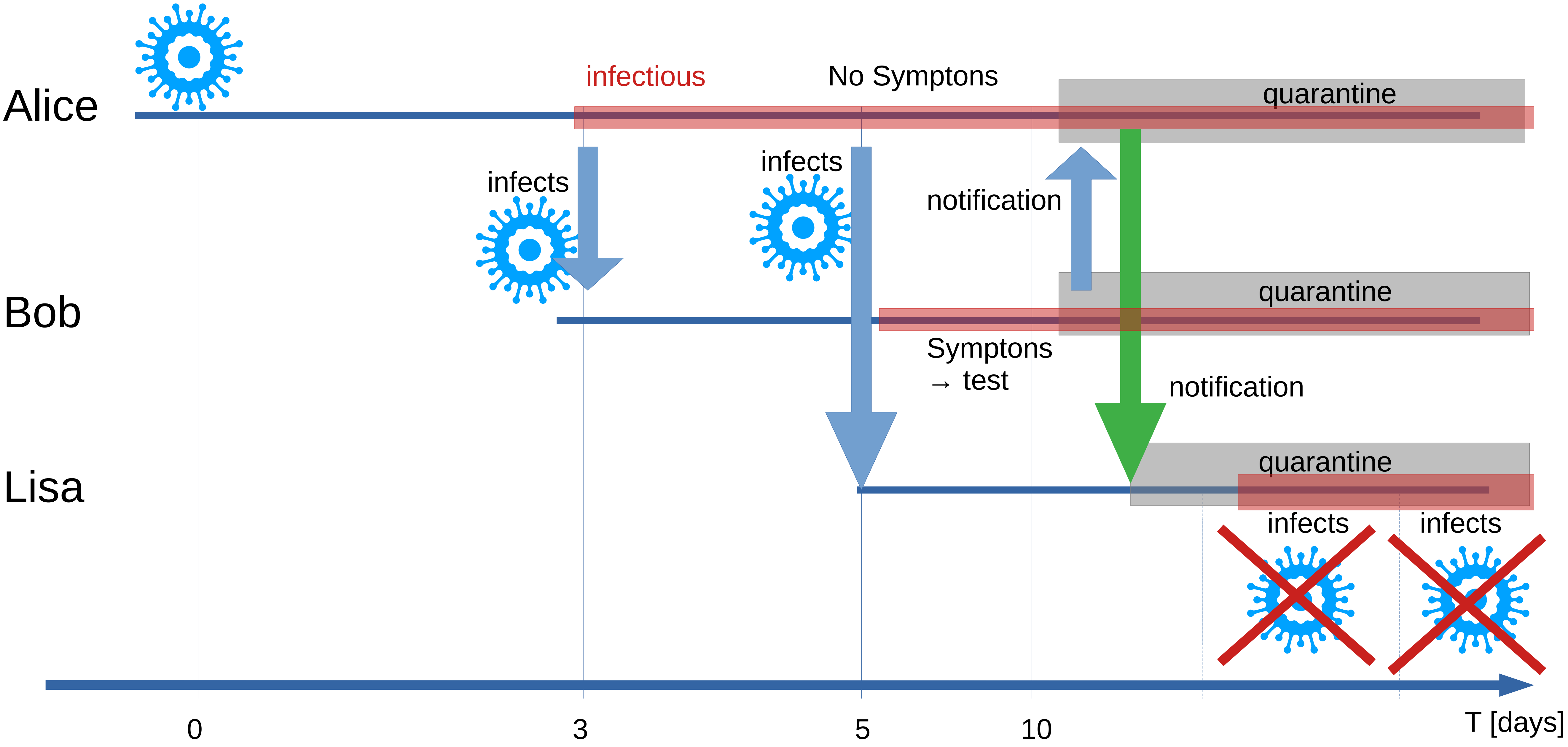


Tracing of second order contacts necessary in order to stop infection chain

doi: https://doi.org/10.1101/2020.03.21.20040329

https://www.sciencemediacenter.de/alle-angebote/fact-sheet/details/news/verlauf-von-covid-19-und-kritische-abschnitte-der-infektion/

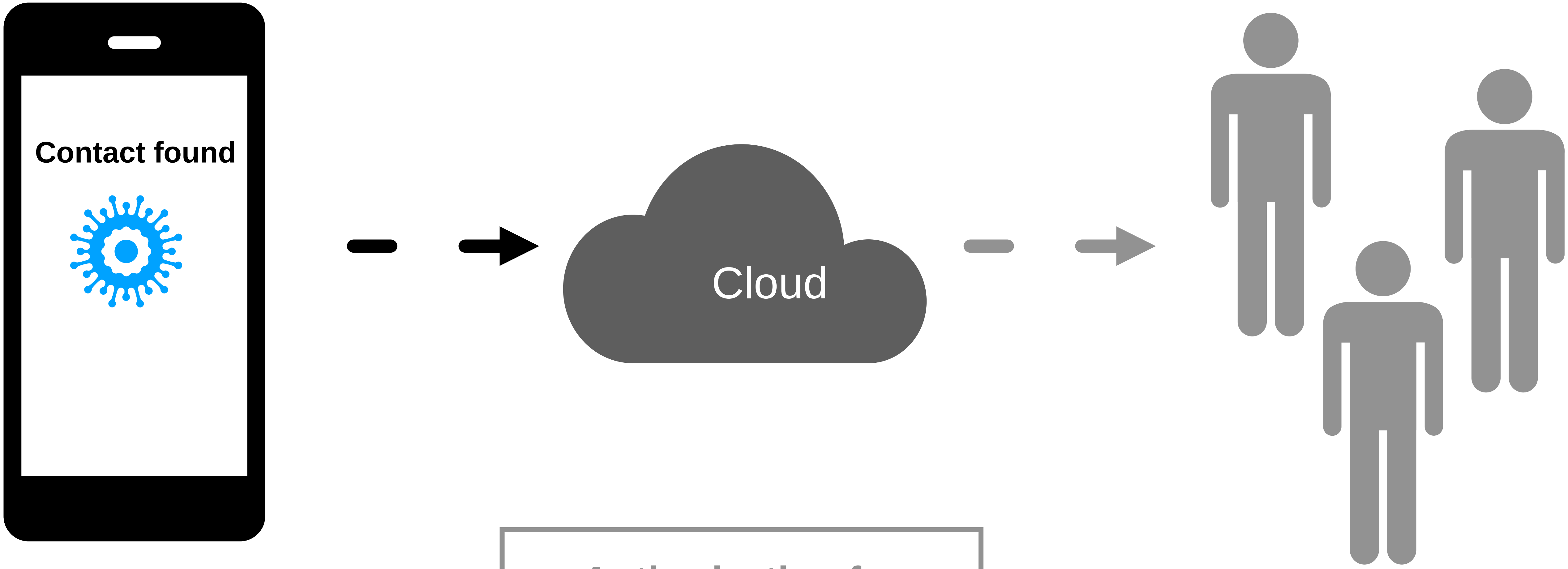
Second order tracing



Tracing of asymptomatic source with second order contacts

doi: https://doi.org/10.1101/2020.03.21.20040329

Second order tracing



Exposure notification

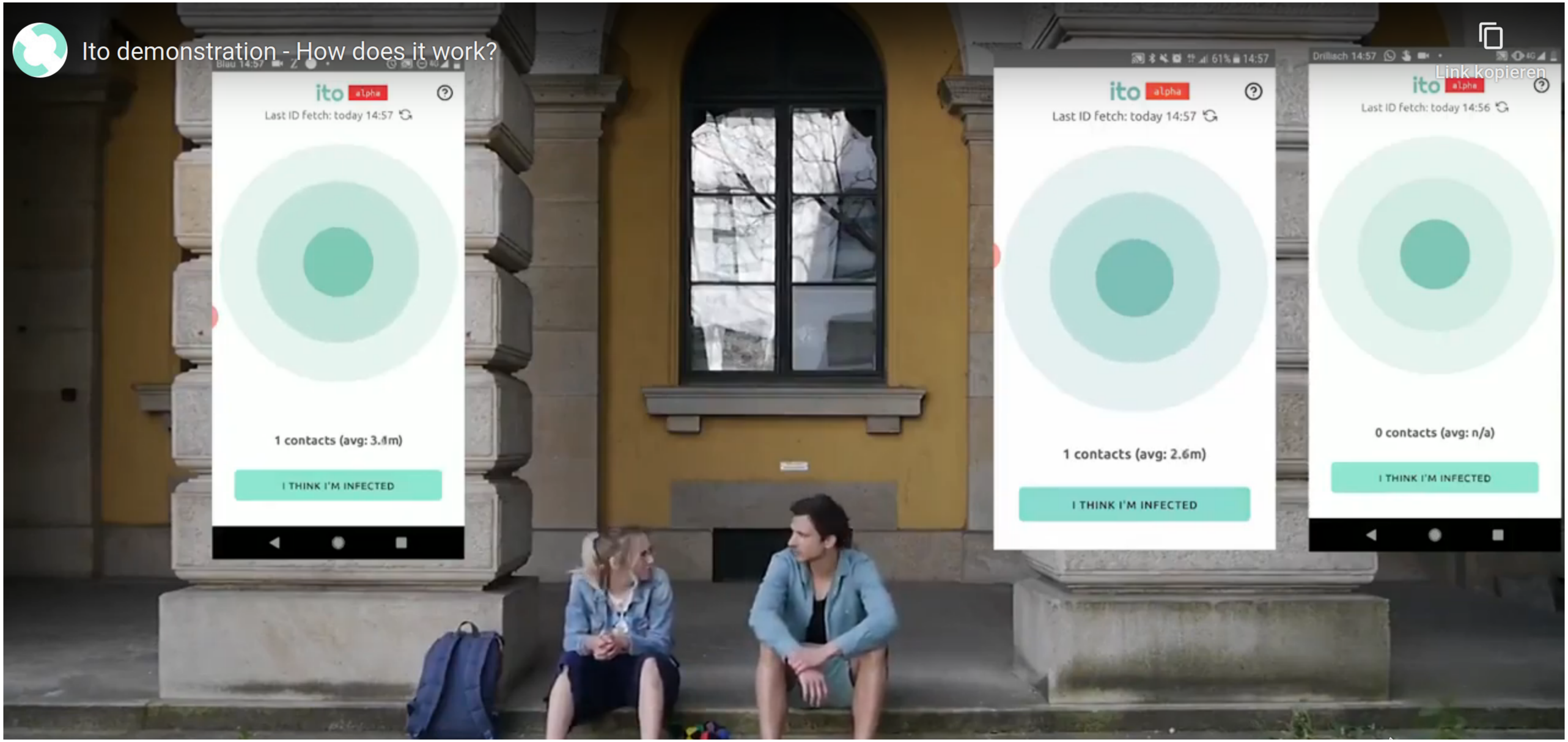
↓

Upload of own keys → second order tracing

Authorization for upload: prove knowledge of “infected TCN” & tin (with zero-knowledge proof)

Notify second order contacts

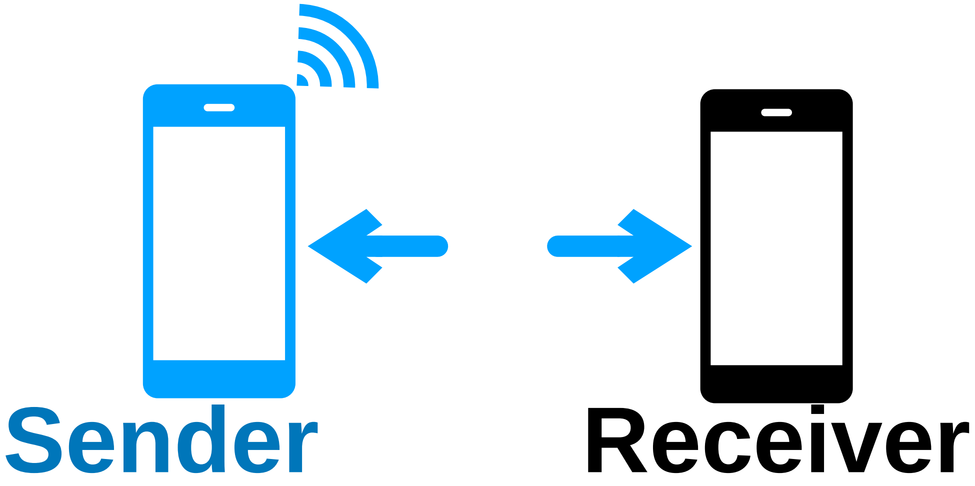
Prototype



https://www.ito-app.org/

Conclusion & outlook

- Privacy preserving contact tracing approach using Bluetooth decentralized design
- Improved design with private set intersection
- Contain infection chain with second order tracing
- Currently testing accuracy of distance measurements



Backup Slides

Private set intersection cardinality

User

scanned IDs (= ID_U, $|ID_U| = a$)

Local public/secret key: pk_U, sk_U

1) Shuffle $Enc_{pku}(ID_U)$

2) Send to server

5) decrypt → $Enc_{pks}(ID_U)$

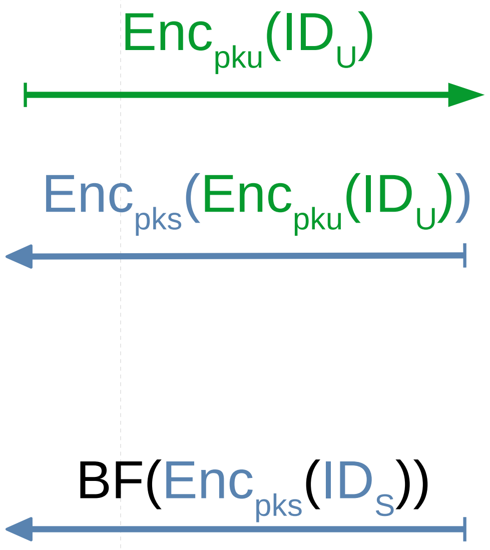
Commutative encryption:
$Enc_A(Enc_B(x)) = Enc_B(Enc_A(x))$

7) Calculate BF($Enc_{pks}(ID_{U,i})$) for each entry in ID_U

8) Check for matches with BF($Enc_{pks}(ID_S)$)

(0 0 0 1 1 0 0 0 1 0) ↔ (0 1 0 0 0 1 0 0 1 0,
0 0 0 1 1 0 0 0 1 0,
1 0 0 0 1 0 0 1 0 0,
……………………..)

9) Get cardinality of intersections



Server

infected IDs (= ID_S, $|ID_S| = b$)

Local public/secret key: pk_S, sk_S

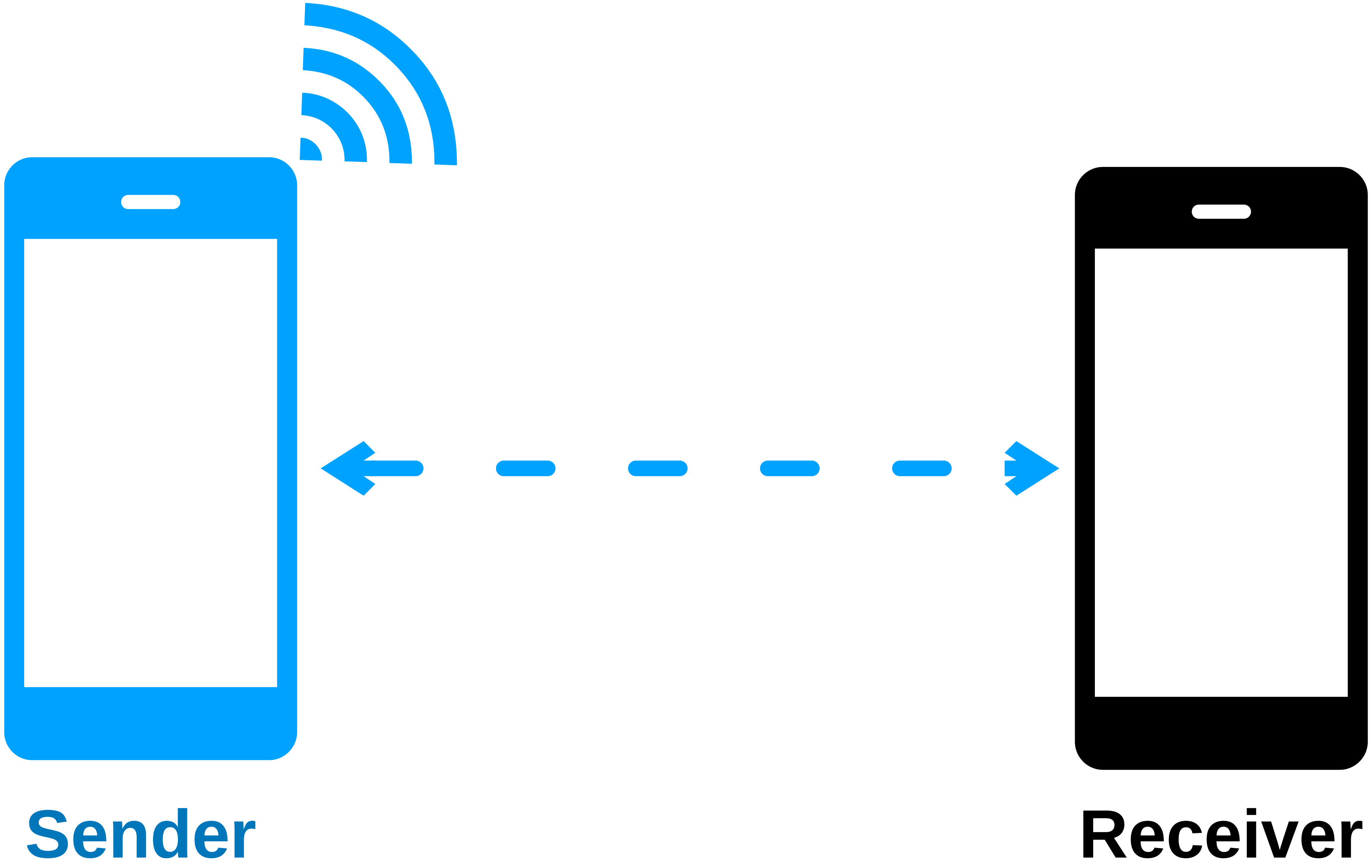
3) Shuffle $Enc_{pku}(ID_U)$

4) Encrypt and send to User

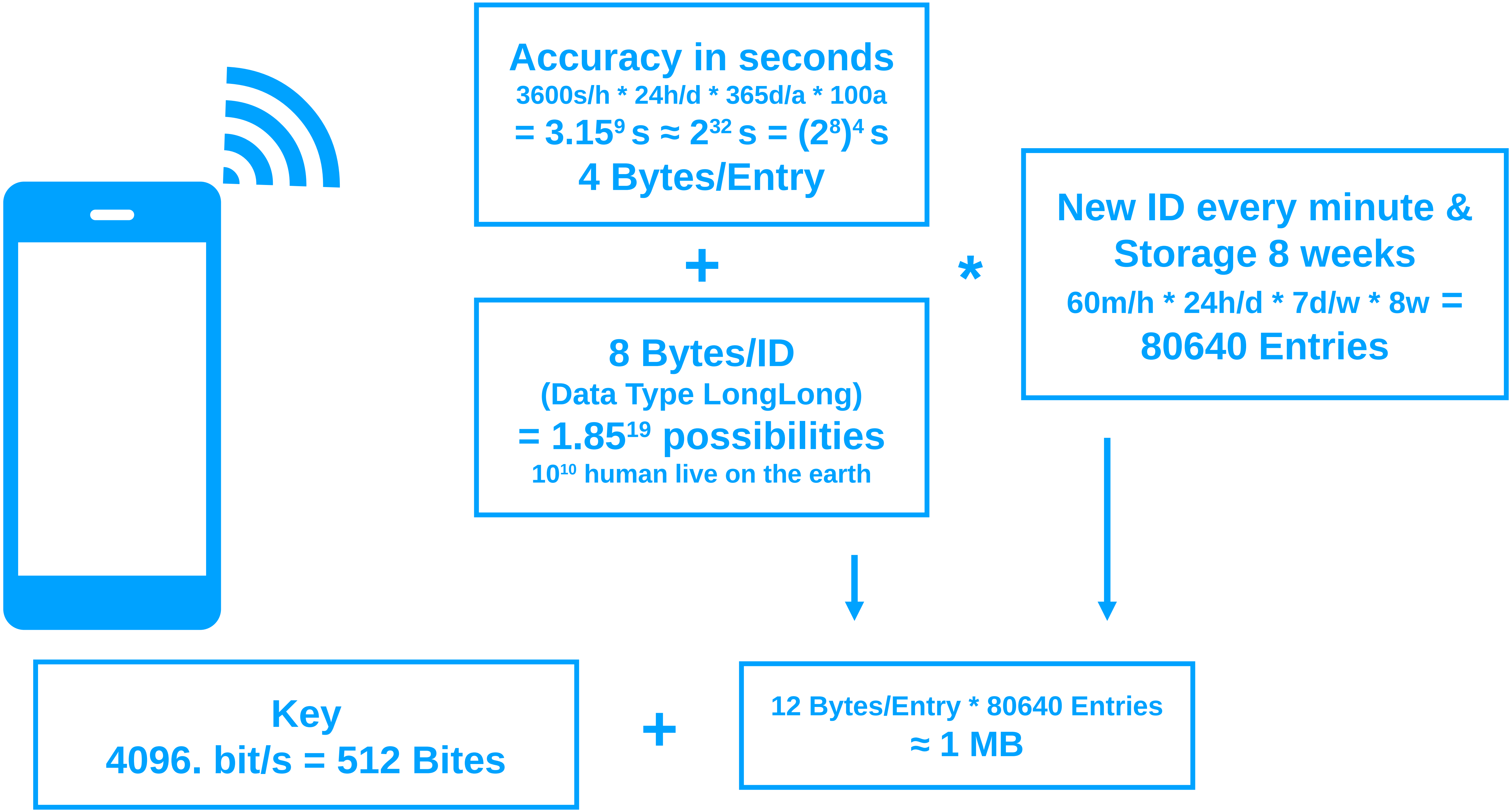
6) Send Bloom filter: BF($Enc_{pks}(ID_S)$)
(BF can be pre-computed)

Bloom filter:
use *k* hash functions on entry, save results in bit array $[0,1]^m$

Only Bluetooth is required



Storage Size Own IDs



Accuracy in seconds

3600s/h * 24h/d * 365d/a * 100a

= 3.15^{9} s ≈ 2^{32} s = $(2^{8})^{4}$ s

4 Bytes/Entry

+

8 Bytes/ID

(Data Type LongLong)

= 1.85^{19} possibilities

10^{10} human live on the earth

*

New ID every minute & Storage 8 weeks

60m/h * 24h/d * 7d/w * 8w =

80640 Entries

Key

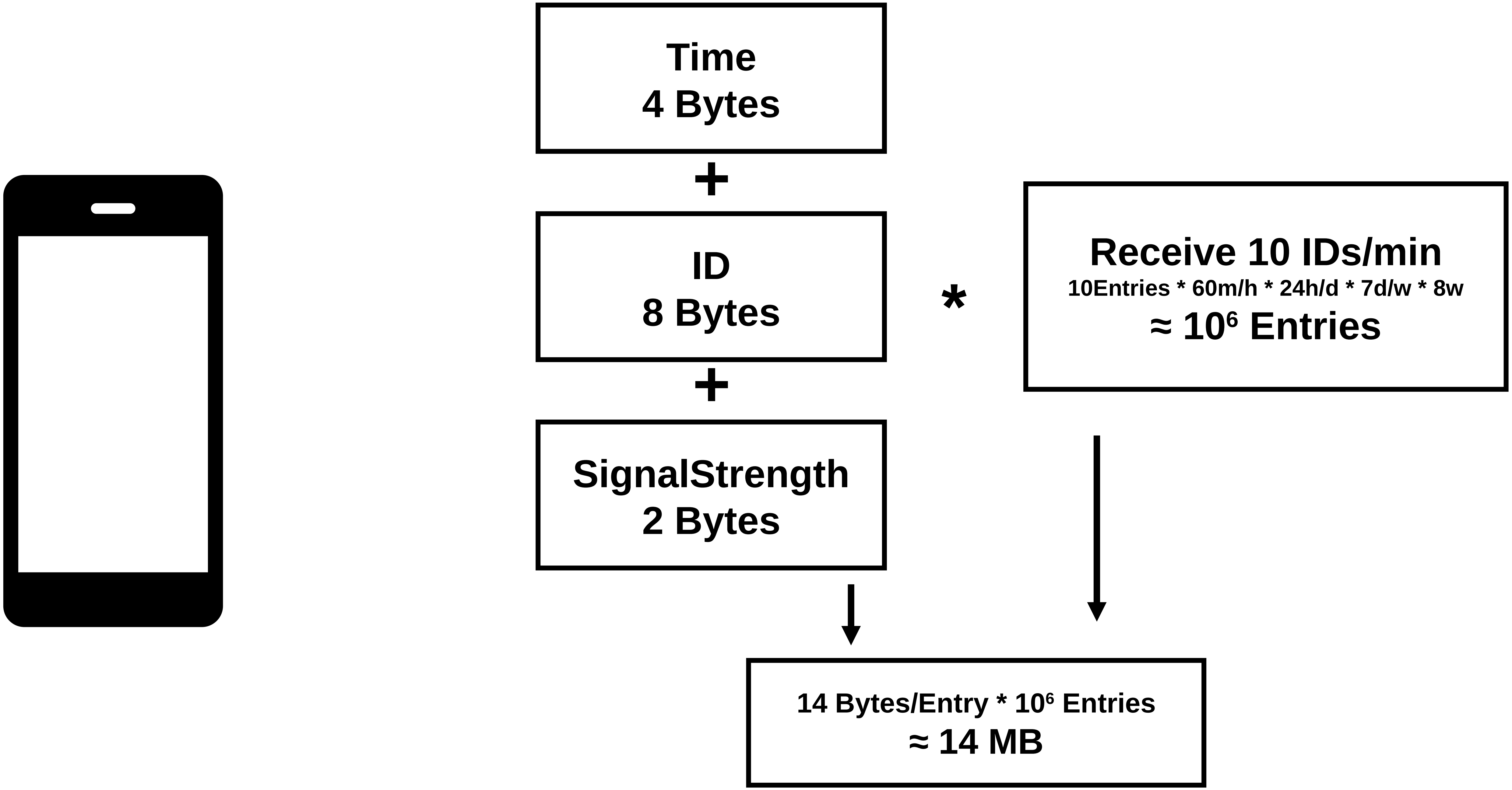
4096. bit/s = 512 Bites

+

12 Bytes/Entry * 80640 Entries

≈ 1 MB

Storage Size Recorded IDs



Time
4 Bytes

+

ID
8 Bytes

+

SignalStrength
2 Bytes

*****

Receive 10 IDs/min
10Entries * 60m/h * 24h/d * 7d/w * 8w
≈ 10^6 Entries

14 Bytes/Entry * 10^6 Entries
≈ 14 MB

Total Storage Size

Own IDs & Key

12 Bytes/Entry * 80640 Entries

≈ 1 MB

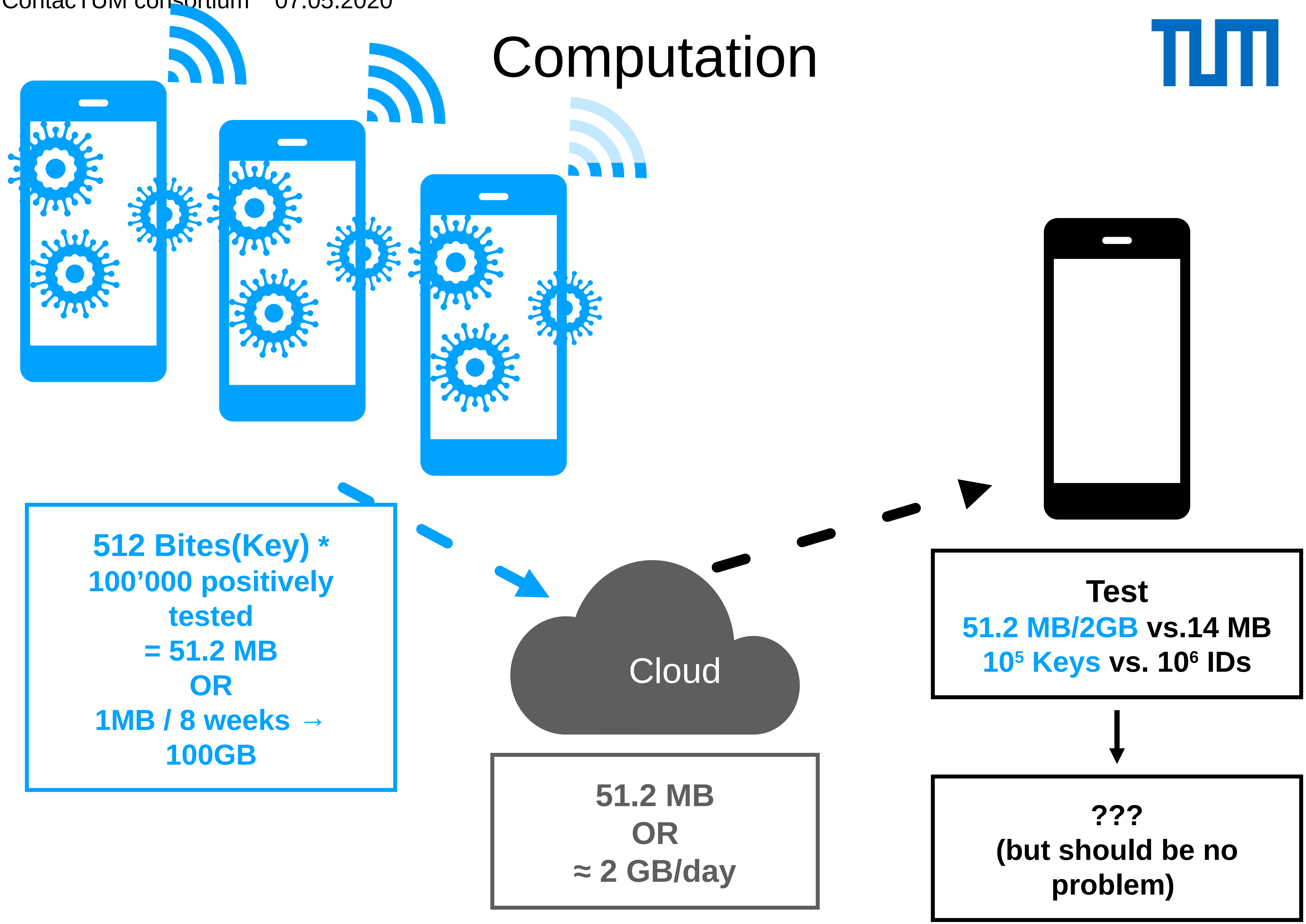
Tracked IDs

14 Bytes/Entry * 10^6 Entries

≈ 14 MB

15 MB / 8 weeks

Computation



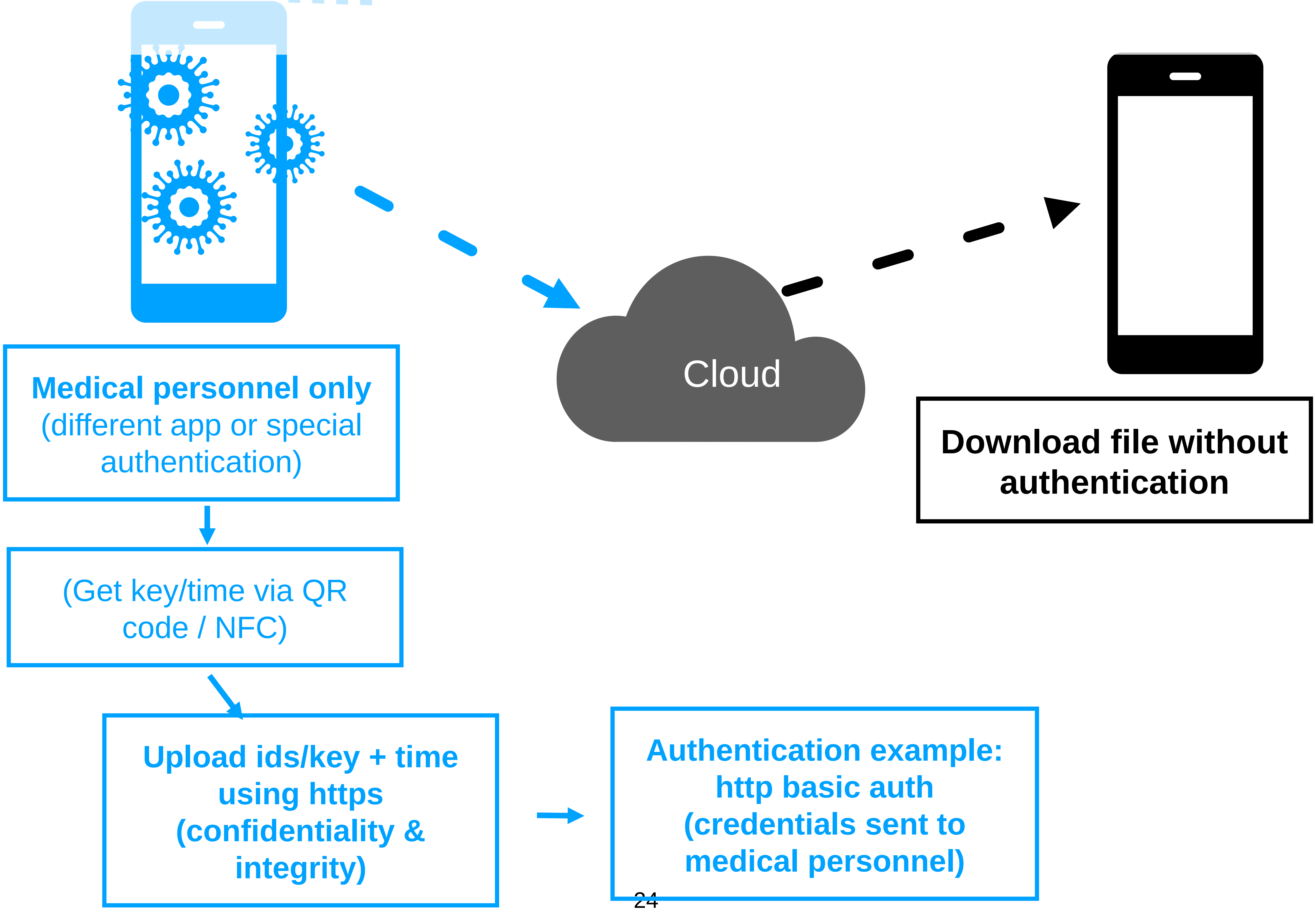
512 Bites(Key) *
100’000 positively tested
= 51.2 MB
OR
1MB / 8 weeks → 100GB

51.2 MB
OR
≈ 2 GB/day

Test
51.2 MB/2GB vs.14 MB
10^5 Keys vs. 10^6 IDs

???
(but should be no problem)

Example protocol for server up- and download



Medical personnel only
(different app or special authentication)

(Get key/time via QR code / NFC)

Upload ids/key + time using https (confidentiality & integrity)

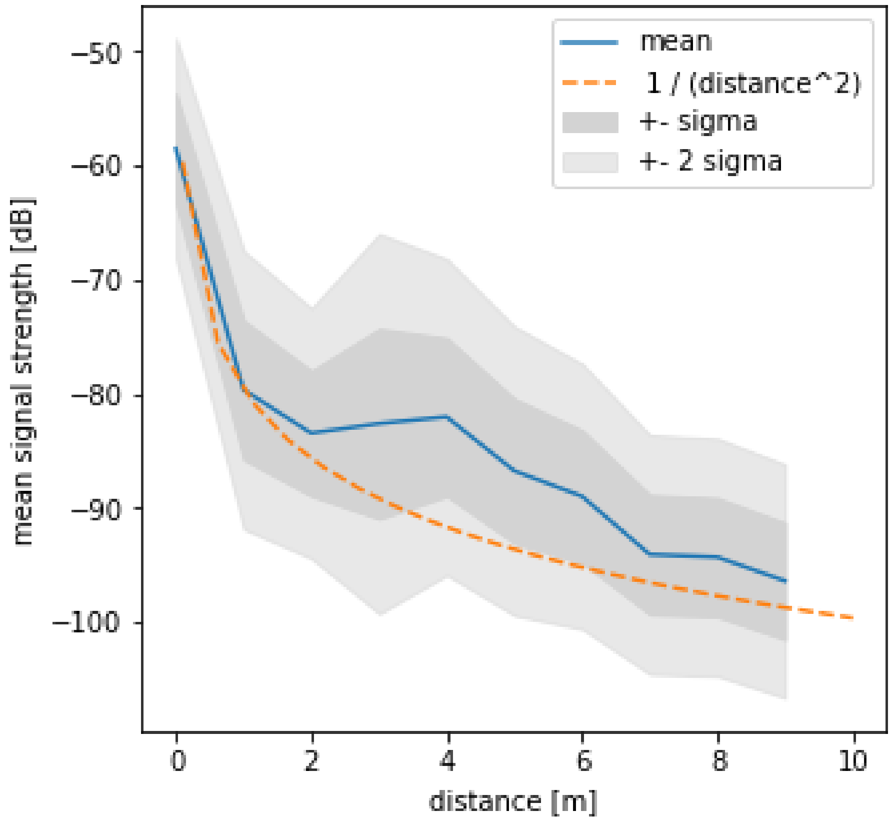
Authentication example: http basic auth (credentials sent to medical personnel)

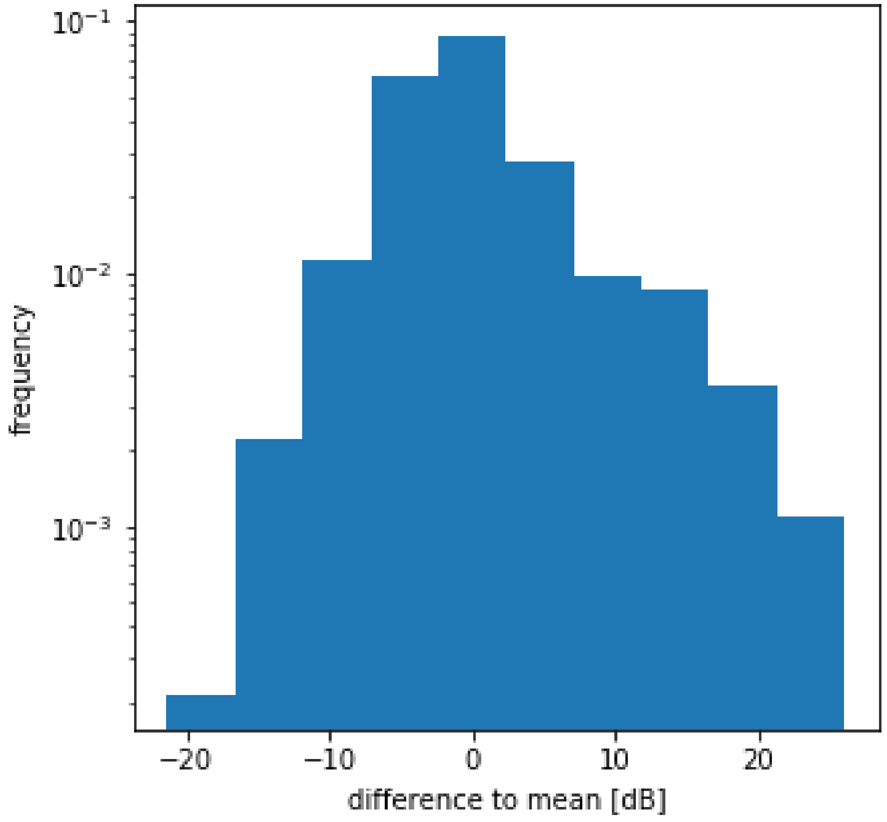
Download file without authentication

Bluetooth tests

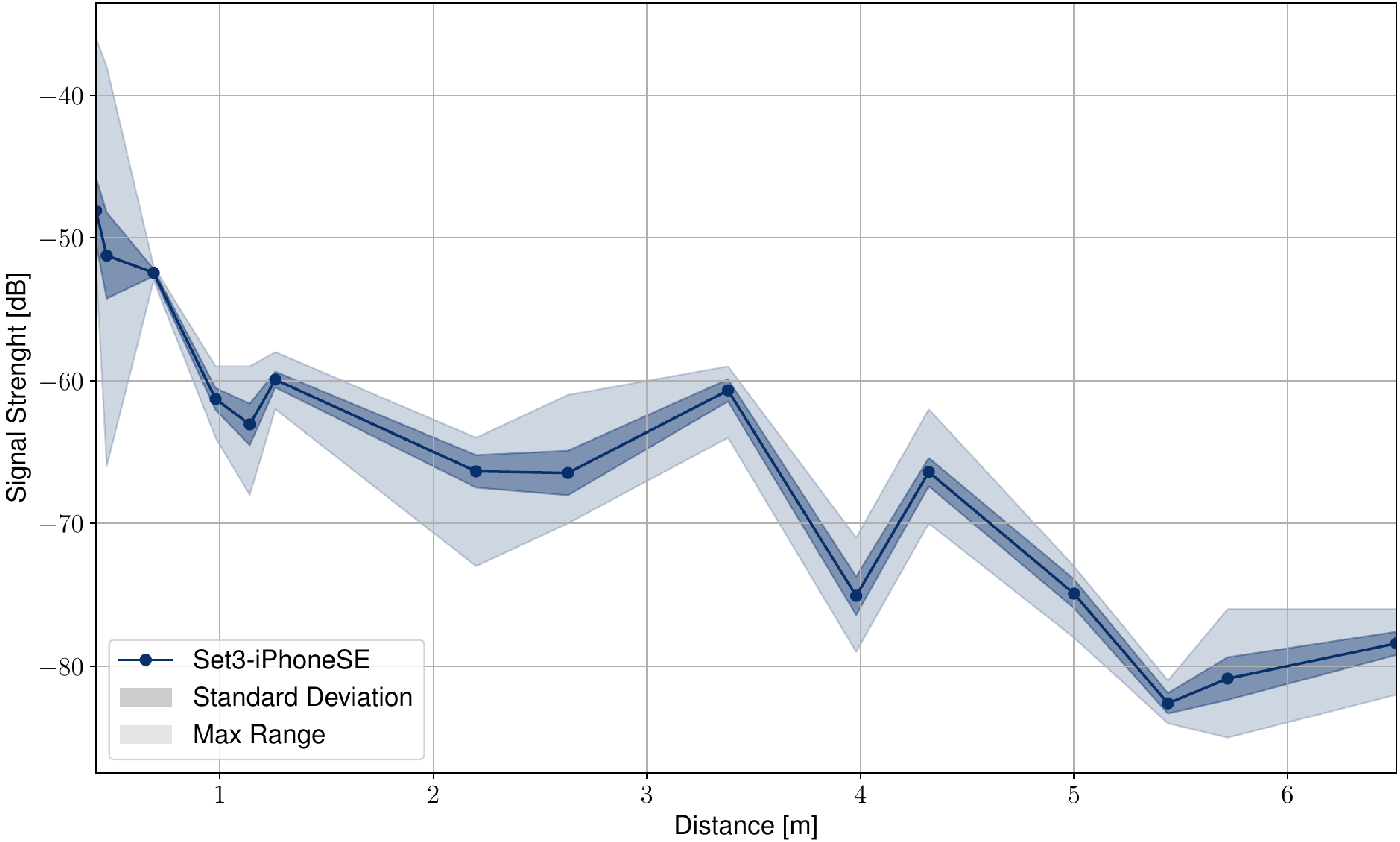
Distances from few cm to ~9 m.

for t > 20 minutes

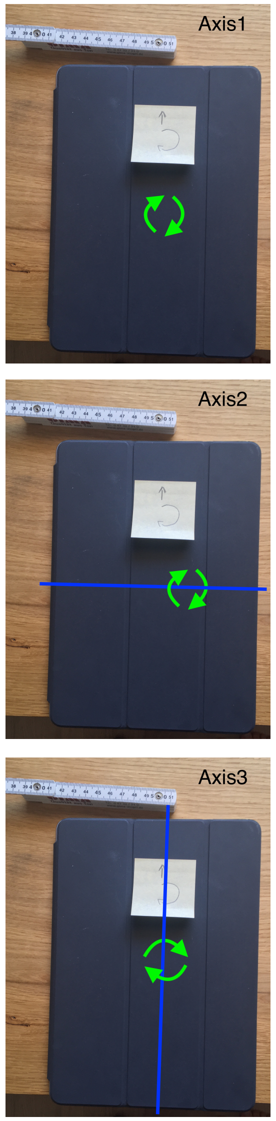


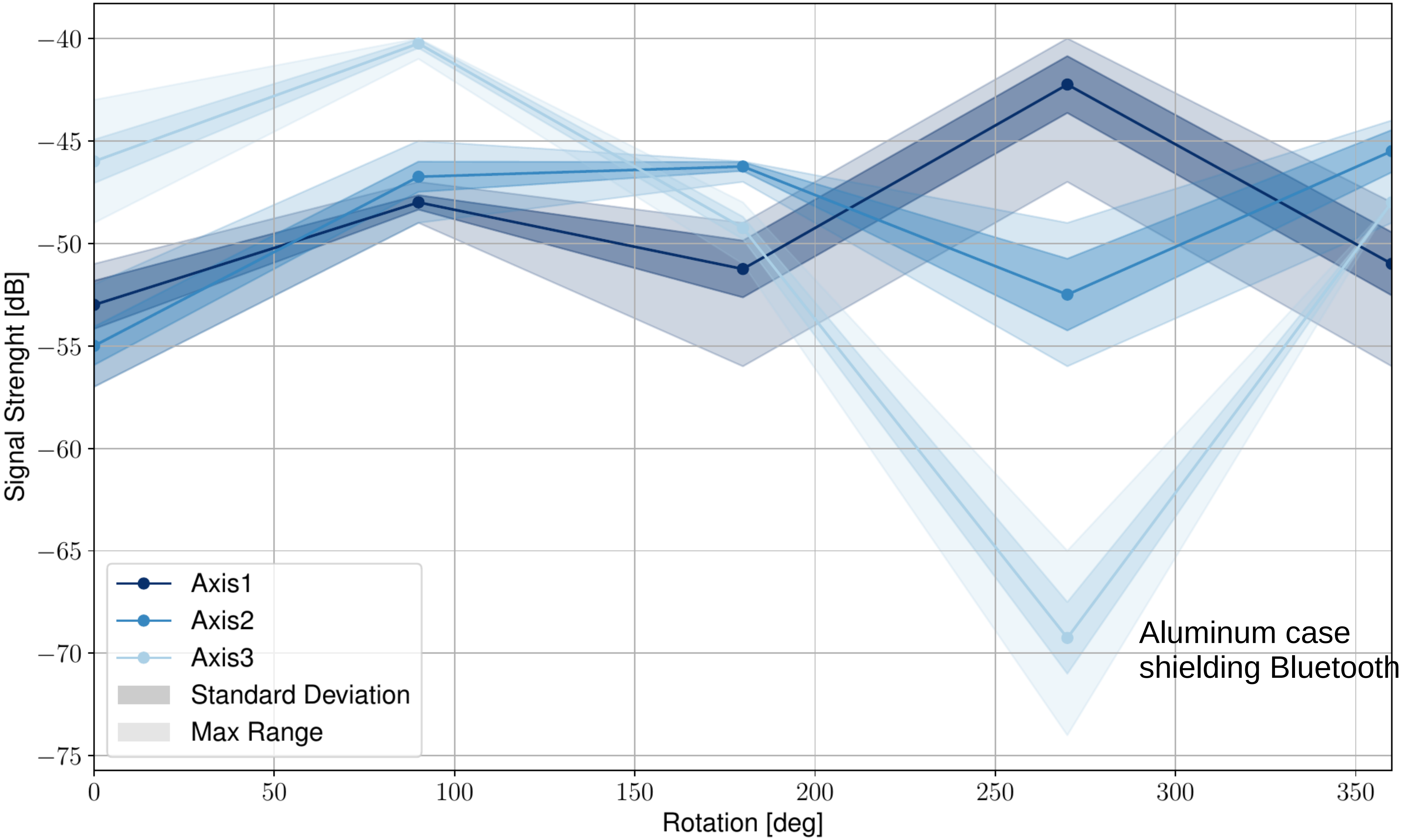


Bluetooth – strength vs distance



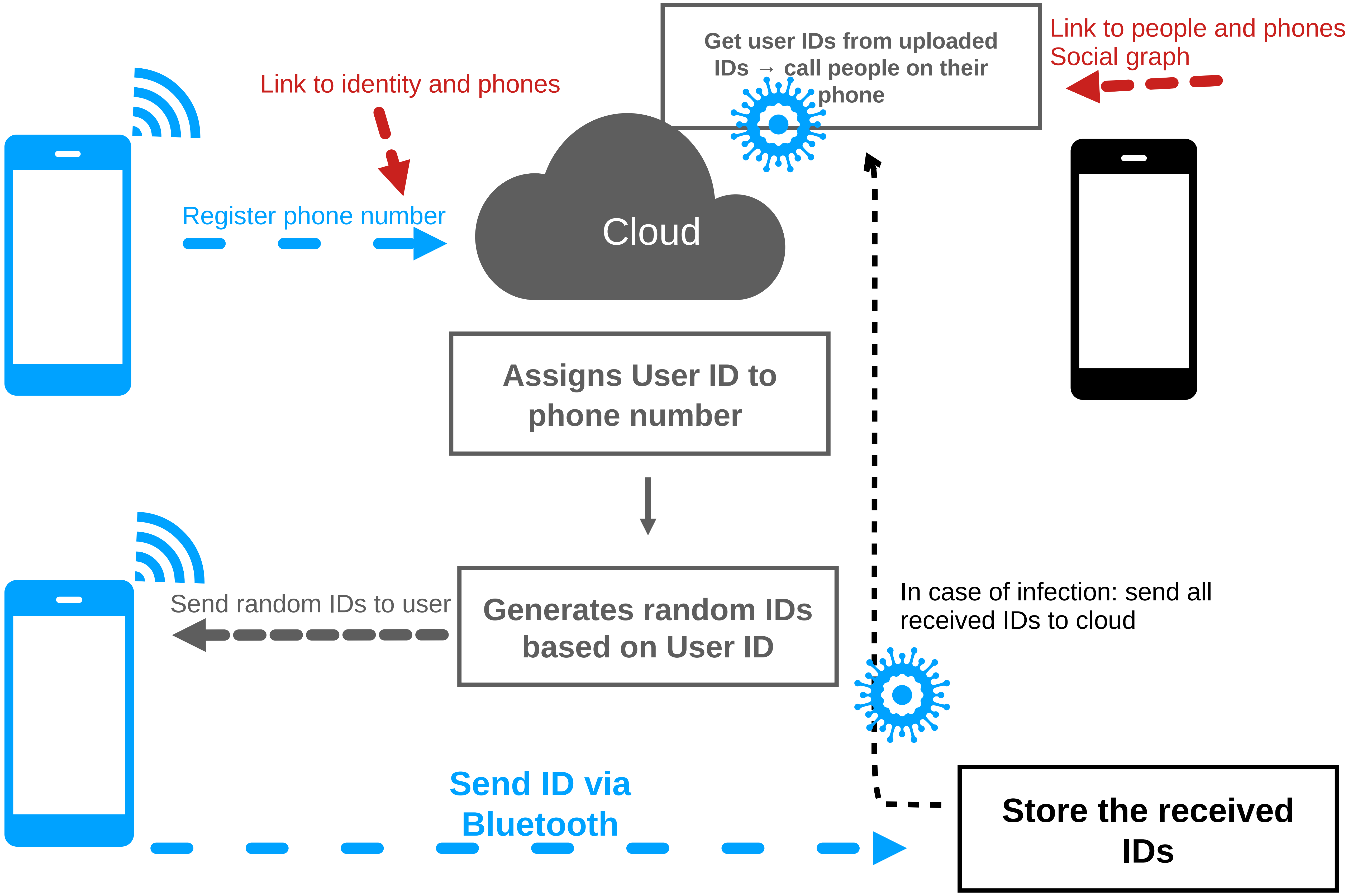
Bluetooth - isotropy





Aluminum case shielding Bluetooth

Comparison with Singapore



Get user IDs from uploaded IDs → call people on their phone

Link to people and phones
Social graph

Link to identity and phones

Register phone number

Cloud

Assigns User ID to phone number

Send random IDs to user

Generates random IDs based on User ID

In case of infection: send all received IDs to cloud

Send ID via Bluetooth

Store the received IDs