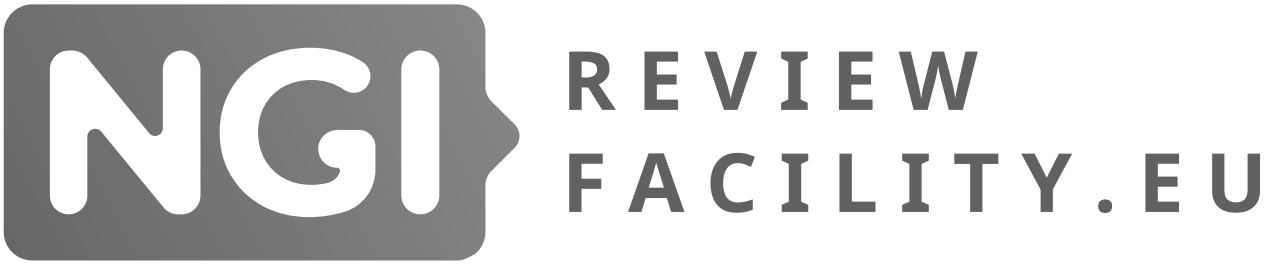


Name: NGI Emergency Tech Review Facility

Contract nr: LC-01499045

Date of signature: 30-04-2020

Name of the deliverable: Deliverable 3.10 –
In-depth assessment of the EU Digital COVID Certificates Gateway – part 2
(Reference Implementations - Source Code Evaluation)

Authors: Tim Hummel, Marcus Bointon (Radically Open Security)

Level of distribution: Confidential, only for members of the facility
(including the Commission Services)

Confidential: Yes

EU Digital COVID Certificate Reference Implementations - Source Code Evaluation

European Commission – Directorate General CONNECT

V 1.0
Amsterdam, April 23rd, 2022
Confidential

Document Properties

Client	European Commission – Directorate General CONNECT
Title	EU Digital COVID Certificate Reference Implementations - Source Code Evaluation
Targets	In the context of the recently added revocation feature: Source Code Audit of the Digital Covid Certificate Reference Implementation Repositories
Version	1.0
Pentesters	Tim Hummel, Robin Peraglie
Authors	Tim Hummel, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	March 14th, 2022	Tim Hummel	Initial draft
0.2	March 22nd, 2022	Tim Hummel	Additional findings
0.3	April 6th, 2022	Tim Hummel	Additional findings + review of Android apps
0.4	April 11th, 2022	Tim Hummel	Additional findings + review of validation rule components
0.5	April 12th, 2022	Tim Hummel	Pre-review
0.6	April 15th, 2022	Marcus Bointon	Review
0.7	April 17th, 2022	Tim Hummel	Minor fixes
1.0	April 23rd, 2022	Marcus Bointon	Further review

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between February 28, 2022 and April 12, 2022, Radically Open Security B.V. carried out a source code evaluation for the European Commission – Directorate General CONNECT.

This report contains our analysis as well as detailed explanations of any findings discovered.

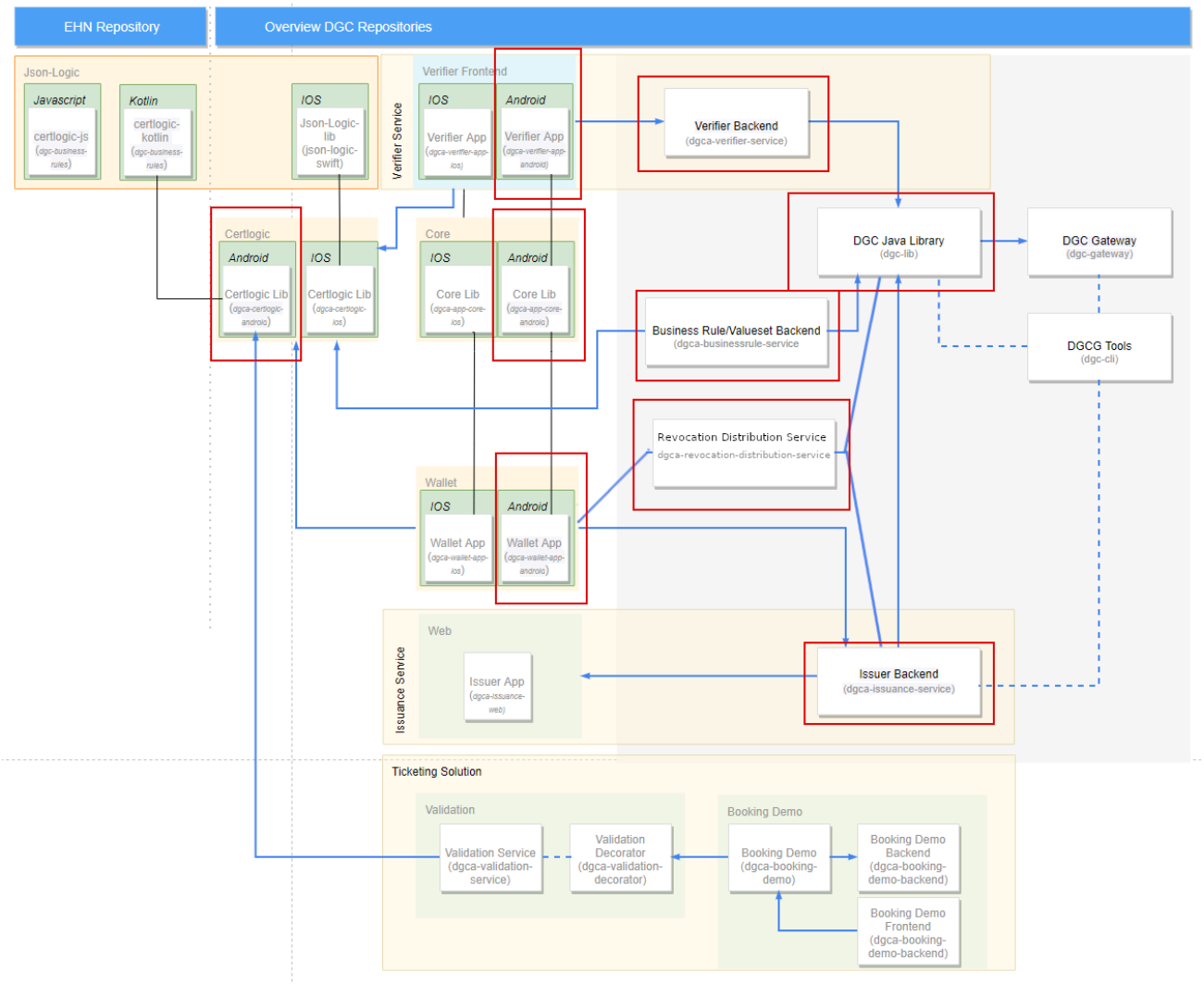
1.2 Scope of work

This is a source code evaluation of the Digital Covid Certificate (DCC) reference implementation repositories the European Commission makes available to member states. These reference implementations include the national services and phone apps. They are not intended to not be used directly, but to be adapted by member states for their own needs. The review was time-boxed and we spent 18 days on the review. This evaluation focused primarily on the revocation feature, but also took a look at the other features in the remaining time available.

Within the scope of this review:

- We evaluated the source code and included documentation in the public repositories on GitHub [1] listed below.
- We used the EC eHealth and COVID-19 documentation [5] as reference where applicable.
- We did not examine any potentially existing non-public developer repositories or take any non-public documentation into account, with the sole exception of two draft design documents: the "Backend to Backend revocation feature" [2] and Revocation B2A Proposal [3].
- All other repositories and documents were excluded from the scope.

The graphic below provides an overview. The in-scope repositories are marked by a red rectangle. The DGC gateway was evaluated previously in a separate report [4]. (Source: adapted from dgc-overview).



The following repositories and versions were reviewed:

- dgca-verifier-service commit-id: `5f0fc3feec926ba41005fd79d8ef18983769fea5` 6 commits since version `1.0.5` Link
- dgca-issuance-service release-version: `1.0.8` Link
- dgca-revocation-distribution-service commit-id: `bba6ebf159438f268eb1309549a1ed3bd36e7617` 3 commits since version `1.0.0` Link
- dgc-lib release-version: `1.1.13` Link
- dgca-verifier-app-android release-version: `1.3.0` Link
- dgca-wallet-app-android release-version: `1.3.0` Link
- dgca-app-core-android release-version: `1.3.0` Link
- dgc-certlogic-android commit-id: `93a32104fb084af22b9c6b50aea9ff245d8bd039` no release yet Link
- dgca-businessrule-service release-version: `1.1.5` Link

In order to use our time most efficiently, we choose to not evaluate:

- iOS components
- The ticketing system
- The certlogic
- The Issuer app and DGCD Tools
- The alternative SAP credentials storage parts of the in-scope components.

1.3 Project objectives

The objective of this project was to evaluate the reference implementations from a security perspective following the rollout of the recently added revocation feature, and use any remaining time-budget to evaluate other features.

1.4 Timeline

The audit took place between February 28, 2022 and April 12, 2022.

1.5 Results In A Nutshell

We discovered 7 Moderate, 11 Low and 1 Unknown-severity issues during this penetration test. Overall it is hard to say what the actual impact of these findings really is; It all depends on how these reference implementations are used in projects derived from them. A minor issue here could become a severe issue in a full implementation.

Missing documentation and guidance, not following best practices, missing checks in various components, and several moderate/minor issues make it harder for implementers to create a secure product based on these references.

1.6 Summary of Findings

ID	Type	Description	Threat level
F-003	Authentication issue	dgca-revocation-distribution-service authentication bypass allows using the revocation lookup endpoint without proper authentication.	Moderate
F-008	Missing verification	dgca-revocation-distribution-service performs limited checks on downloaded data.	Moderate
F-012	Authentication issue	dgca-issuance-service DCCs can be reclaimed by stealing a TAN that should not exist, according to the documentation.	Moderate
F-013	Best practice	dgca-verifier-app-android debug options are not documented.	Moderate

F-014	Missing verification	dgca-verifier-app-android does not verify that the DCC's country matches the DCC signer's country.	Moderate
F-015	Missing verification	In dgca-verifier-app-android and revocation-data-service there is no verification that the DCC revocation is issued by the country that issued the DCC.	Moderate
F-017	Missing verification	In dgca-verifier-app-android and dgca-wallet-app-android there is no certificate pinning when accessing revocation-data-service.	Moderate
F-001	Best practice	Configuration files in the repositories contain passwords.	Low
F-004	Best practice	Test environment URLs are published in the repositories.	Low
F-007	Missing verification	dgca-verifier-service performs limited checks on downloaded data.	Low
F-010	Best practice	Limited documentation hampers creating secure implementations based on the reference implementations.	Low
F-016	Missing feature	No lookup proof implemented.	Low
F-018	Best practice	dgca-wallet-app-android user authentication is not coupled to stored data.	Low
F-019	Missleading interface	The dgca-wallet-app-android login dialogue wrongly claims to be biometric when it is not.	Low
F-020	Best practice	dgca-wallet-app-android uses an ECB encryption scheme.	Low
F-021	Crypto	The dgca-verifier-app-android uses AES-GCM with an hardcoded IV.	Low
F-022	Missing verification	dgca-businessrule-service does not verify the origin country of a rule.	Low
F-023	Integrity protection	dgca-businessrule-service offers its public key for signature function over an untrusted connection.	Low
F-002	Integrity protection	Services should enforce secure communication.	Unknown

1.7 Summary of Recommendations

ID	Type	Recommendation
F-003	Authentication issue	• Check that the provided payload values belong to the public key during revocation lookup in the `dgca-revocation-distribution-service`.
F-008	Missing verification	• Implement checks in the `dgca-revocation-distribution-service` that verify data received from the gateway, especially data that the gateway does not also verify internally.
F-012	Authentication issue	• Do not allow reclaiming a certificate in the `dgca-issuance-service`. • Do not issue a new TAN when a DCC is claimed.
F-013	Best practice	• Document debug features in `dgca-verifier-app-android`.

		• Remove debug features in normal builds.
F-014	Missing verification	• Check that the DSC's country matches the DCC's country that it signed in the `dgca-verifier-app-android`.
F-015	Missing verification	• Verify that countries do not try to revoke each other's DCCs either in the apps or at a higher level.
F-017	Missing verification	• Implement certificate pinning between the apps and the `revocation-distribution-service` or support revocation data signing.
F-001	Best practice	• Do not store credentials in a public repository. • Inject sensitive configuration values using environment variables without default values. • Provide guidelines that these credentials should be changed.
F-002	Integrity protection	• Specify that the services should be made available only via HTTPS. • Ensure in code that only HTTPS is used when connecting to other components. • Allow for certificate pinning.
F-004	Best practice	• Change the actual URLs used for testing and do not publish them. • Use dummy domains (e.g. `example.com` from RFC2606) for documentation purposes.
F-007	Missing verification	• Assume no trust and implement KID and country code checks in the `dgca-verifier-service`.
F-010	Best practice	• Provide documentation to help implementers: Document inner workings, additional features, and deviations from reference documents.
F-016	Missing feature	• Implement the lookup proof described in the reference documents or compensation measure.
F-018	Best practice	• Connect `dgca-wallet-app-android` logins to data keystore keys.
F-019	Missleading interface	• Change the login screen text and design accordingly on non-biometric devices in the `dgca-wallet-app-android`.
F-020	Best practice	• Use a more secure encryption scheme in the `dgca-wallet-app-android`.
F-021	Crypto	• Use AES-GCM properly with unique IVs in the `dgca-verifier-app-android`.
F-022	Missing verification	• Verify in the `dgca-businessrule-service` that each rule is signed by the country it applies to.
F-023	Integrity protection	• Provide documentation about the signing feature of the `dgca-businessrule-service` so it can be used correctly. • Make the public key available in a hard-coded or verifiable way instead of via an unprotected endpoint.

2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.
- **High**
  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**
  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**
  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Findings

We have identified the following issues:

3.1 F-003 — Authentication bypass

Vulnerability ID: F-003

Vulnerability type: Authentication issue

Threat level: Moderate

Description:

`dgca-revocation-distribution-service` authentication bypass allows using the revocation lookup endpoint without proper authentication.

Technical description:

The endpoint `/revocation/lookup` in `dgca-revocation-distribution-service` seems to implement the Wallet App Lookup using "An authorization over the keypair which was used for claiming a certificate" as described in the [3]. This is because "it must be guaranteed that access is only limited to the current holder of the certificate". We write "seems", because there is no proper documentation on this endpoint, but it matches the idea given in the proposal document.

For this request a JWT token with the following structure is submitted to a `dgca-revocation-distribution-service` endpoint:

```
Header +
public class RevocationCheckTokenPayload {
    private String sub;
    private List<String> payload;
    private long exp;
}
+ Signature
```

Looking at the implementation in the `dgca-wallet-app-android` shows that both the `sub` and `payload` fields contain DCC hashes:

```
val jwtTokenBody = CertificateRevocationSignaturesEntity(
    sub = uvciSha256Full,
    payload = listOf(uvciSha256, coUvciSha256, signatureSha256),
    exp = System.currentTimeMillis() + TimeUnit.MINUTES.toMillis(15)
)
```

The revocation distribution service extracts the sub-claim from the JWT token. It then asks the `dgca-issuance-service` `/dgci/{hash}` endpoint for the public key of the provided sub-claim. The received public key is used to verify the JWT token. This ensures that the JWT request is signed by a public key known to the `dgca-issuance-service` for the provided sub-claim. After verification, the `dgca-revocation-distribution-service` performs a lookup to check if any of the provided hashes in the payload list have been revoked.

However there is no check in the `dgca-revocation-distribution-service` that the `sub` and `payload` list hashes belong to each other, thus there is thus no guaranteed relationship between the two.

This makes it possible for anybody holding any valid DCC to use the endpoint to look up anybody else's revocation status, which is contrary to the statement in the design document.

Impact:

Anybody with a valid DCC can use the supposedly restricted revocation lookup endpoint to look up anybody's status. The impact is limited because one could also just download the freely available revocation list and perform the lookup directly.

Recommendation:

- Check that the provided payload values belong to the public key during revocation lookup in the `dgca-revocation-distribution-service`.

3.2 F-008 — Data download in dgca-revocation-distribution-service

Vulnerability ID: F-008

Vulnerability type: Missing verification

Threat level: Moderate

Description:

`dgca-revocation-distribution-service` performs limited checks on downloaded data.

Technical description:

This `dgca-revocation-distribution-service` is essentially a national cache for the revocation data. It downloads revocation data from the gateway and repackages it for easier distribution.

The revocation data is downloaded in signed batches together with some metadata. There are very limited checks on the downloaded data:

- The signature of the batches is checked, but only against the CMS internal certificate and not against a trust anchor or any known country certificate, before the contained hashes are added to the `HashesRepository`.
- The metadata is added to the `BatchListRepository` without checking, for example whether the metadata matches the signed data.
- JSON structure and whether values within it are valid base 64 is not checked.

This is not helped by the gateway already performing limited checks, see "F-004 Limited checks on revocation data" "F-007 JSON parsing problems" in [4]. In conjunction that means that the latter two points are checked by neither the gateway nor this service. The first point is partially mitigated if the service is used with the official gateway, because that one does check that data is signed by a known country.

It's possible that we overlooked the checks in this nested code structure.

Impact:

In theory not being aware of these limited checks could lead to parsing errors or putting the wrong level of trust in revocation data forwarded to the apps. In practice this probably has only limited impact as all members of this network should be trusted members states and the official gateway is used.

Recommendation:

- Implement checks in the `dgca-revocation-distribution-service` that verify data received from the gateway, especially data that the gateway does not also verify internally.

3.3 F-012 — DCCs can be reclaimed

Vulnerability ID: F-012

Vulnerability type: Authentication issue

Threat level: Moderate

Description:

`dgca-issuance-service` DCCs can be reclaimed by stealing a TAN that should not exist, according to the documentation.

Technical description:

The function `claim` in the `dgca-issuance-service` should, according to https://ec.europa.eu/health/system/files/2022-02/digital-covid-certificates_v4_en_0.pdf, claim a DCC once and then delete the TAN. See Figure 7: TAN Validation.

The code below deviates from this approach in multiple critical ways:

1. If a DCC has already been claimed the code does not abort. Worse, it also skips the TAN expiration check. The code allows claiming already-claimed DCCs without an expiration limit.
2. Instead of deleting the TAN, a new TAN is generated, which is fine as an alternative to deletion invalidating the old TAN, but for unknown reasons this TAN is passed to the Wallet apps in a response.

```
public ClaimResponse claim(ClaimRequest claimRequest) {
 log.debug("Claim certificate with ID '{}'", claimRequest.getDgci());
 if (!verifySignature(claimRequest)) {
  throw new WrongRequest("Signature verification failed");
 }
 Optional<DgciEntity> dgciEntityOptional = dgciRepository.findByDgci(claimRequest.getDgci());
 if (dgciEntityOptional.isPresent()) {
  DgciEntity dgciEntity = dgciEntityOptional.get();
  if (dgciEntity.getRetryCounter() > MAX_CLAIM_RETRY_TAN) {
   throw new WrongRequest("Claim max try exceeded");
  }
  if (!dgciEntity.getCertHash().equals(claimRequest.getCertHash())) {
   throw new WrongRequest("Cert hash mismatch");
  }
  if (!dgciEntity.getHashedTan().equals(claimRequest.getTanHash())) {
   dgciEntity.setRetryCounter(dgciEntity.getRetryCounter() + 1);
   dgciRepository.saveAndFlush(dgciEntity);
   throw new WrongRequest("TAN mismatch");
  }
  if (!dgciEntity.isClaimed()) {
   ZonedDateTime tanExpireTime = dgciEntity.getCreatedAt()
    .plus(issuanceConfigProperties.getTanExpirationHours());
   if (tanExpireTime.isBefore(ZonedDateTime.now())) {
    throw new WrongRequest("TAN expired");
   }
  }
  dgciEntity.setClaimed(true);
  dgciEntity.setRetryCounter(dgciEntity.getRetryCounter() + 1);
  dgciEntity.setPublicKey(asJwk(claimRequest.getPublicKey()));
  Tan newTan = Tan.create();
  dgciEntity.setHashedTan(newTan.getHashedTan());
  dgciEntity.setRetryCounter(0);
  dgciRepository.saveAndFlush(dgciEncomflicts innconcurrent situationstity);
  log.info("Certificate with ID '{}' successfully claimed.", dgciEntity.getDgci());
```

```
  ClaimResponse claimResponse = new ClaimResponse();
  claimResponse.setTan(newTan.getRawTan());
  return claimResponse;
 } else {
  log.warn("Cannot find certificate with ID '{}'", claimRequest.getDgci());
  throw new DgciNotFound("Cannot find DGCI: " + claimRequest.getDgci());
 }
}
```

We checked the code of `dgca-wallet-app-android` and found that the TAN is received and stored in the local database.

If an attacker gets hold of that TAN and DCC ID (DGCI) he can call the endpoint again and claim the DCC. That arguably isn't very easy and in most cases requires privileged access to the victim device, but then extracting the TAN is very easy compared to the normal level of protection, which would be the private key being protected by an Android phone's keystore. This is significantly less secure.

Additionally there are other bugs:

1. The last `dgciEntity.setRetryCounter(dgciEntity.getRetryCounter() + 1);` is not immediately followed by a `dgciRepository.saveAndFlush(dgciEntity);`, but a line that could throw an exception and abort. This could mean the increment of the `RetryCounter` is not saved. However, we don't see how this would benefit an attacker.
2. The `RetryCounter` is limited. There is no database lock applied in this critical section, leaving it open to problems in concurrent situations. If this service is run on multiple nodes, this could enable an attacker to get a few more TAN tries by issuing requests to multiple instances in parallel. This is unlikely to create any real benefit, because a few more parallel tries are still a very long way short of being able to guess the TAN within a reasonable probability.

Impact:

It allows reclaiming a DCC. It allows re-claiming a DCC when having access to a phone's data store, instead of the more protected key storage. The claimed DCC can be used for travel request and revocation lookups.

Recommendation:

- Do not allow reclaiming a certificate in the `dgca-issuance-service`.
- Do not issue a new TAN when a DCC is claimed.

3.4 F-013 — Verifier app debug options

Vulnerability ID: F-013

Vulnerability type: Best practice

Threat level: Moderate

Description:

`dgca-verifier-app-android` debug options are not documented.

Technical description:

`dgca-verifier-app-android` handles sensitive information of DCC holders. It contains debug features (see `DebugModeState`) to share/log this data. It is evident through functions such as `anonymizationManager.anonymizeDcc(certificateModel, state)` and the different debug levels that the developer thought about user's privacy. There is however no documentation about these debug features and they are included in normal non-debug app builds.

Impact:

Debug features might be abused or accidentally be used to expose and harvest personal information

Recommendation:

- Document debug features in `dgca-verifier-app-android`.
- Remove debug features in normal builds.

3.5 F-014 — Verifier app country of origin

Vulnerability ID: F-014

Vulnerability type: Missing verification

Threat level: Moderate

Description:

`dgca-verifier-app-android` does not verify that the DCC's country matches the DCC signer's country.

Technical description:

The `dgca-verifier-app-android` DCC signature verification is done by class `VerificationViewModel` and class `VerificationCryptoService`. They check that there is a KeyID (KID) embedded in the DCC. They use this KID to find the Digital Signer Certificate (DSC) that will be used to verify the DCC signature. They do not verify that the DSC country code used to verify the DCC matches the DCC country code.

Impact:

Other countries could create and sign DCCs that would be treated and displayed as belonging to another country.

Recommendation:

- Check that the DSC's country matches the DCC's country that it signed in the `dgca-verifier-app-android`.

3.6 F-015 — DCCs revocation by other countries

Vulnerability ID: F-015

Vulnerability type: Missing verification

Threat level: Moderate

Description:

In `dgca-verifier-app-android` and `revocation-data-service` there is no verification that the DCC revocation is issued by the country that issued the DCC.

Technical description:

The DGC gateway blindly forwards revocation data and does not verify that the country providing them is also the owner of the certificate with the matching KID. This is by design, which we also mentioned as finding in the DGC gateway report [4].

The `dgca-verifier-app-android` has knowledge of the KID, DCC hash (DGCI), and country that a DCC is issued for. The national `revocation-data-service` knows which DCC hash revocation list is published/signed by which

country, but does not forward this information to the apps. Combined, this information could be used to verify if DCC revocation is issued by the DCC-issuing country.

Impact:

Countries can revoke other countries' DCCs

Recommendation:

- Verify that countries do not try to revoke each other's DCCs either in the apps or at a higher level.

3.7 F-017 — Revocation service certificate pinning

Vulnerability ID: F-017

Vulnerability type: Missing verification

Threat level: Moderate

Description:

In `dgca-verifier-app-android` and `dgca-wallet-app-android` there is no certificate pinning when accessing `revocation-data-service`.

Technical description:

Neither the `dgca-verifier-app-android` nor the `dgca-wallet-app-android` use certificate pinning when accessing the `revocation-distribution-service`.

The apps use certificate pinning to access the other national services `verifier-service`, `issuance-service`, and `businessrule-service`, but not the `revocation-distribution-service`.

The Revocation B2A Proposal [3] states that revocation data should be signed so it is verifiable by the app, which is optional if certificate pinning is used. Neither is implemented.

Impact:

The trust level of revocation data expected by the design documents is not reached. MitM attacks are made easier.

Recommendation:

- Implement certificate pinning between the apps and the `revocation-distribution-service` or support revocation data signing.

3.8 F-001 — Repositories contain passwords

Vulnerability ID: F-001

Vulnerability type: Best practice

Threat level: Low

Description:

Configuration files in the repositories contain passwords.

Technical description:

Various files in national service reference implementation repositories contain passwords. Usually in:

- `docker-compose.yml`
- `application.yml`
- `*.yml`

These credentials are handy for testing, but in the absence of clear deployment guidelines there is a risk that these values are actually used by implementers in real projects.

In some files we saw that environment variables without default values are used, and this is a correct and safe approach.

Impact:

Adding "example" credentials to a repository carries a risk that they will be used in production. However, even if these credentials are made public they carry limited risk, as they are typically not used in components that are publicly accessible.

Recommendation:

- Do not store credentials in a public repository.

- Inject sensitive configuration values using environment variables without default values.
- Provide guidelines that these credentials should be changed.

3.9 F-002 — Communication security

Vulnerability ID: F-002

Vulnerability type: Integrity protection

Threat level: Unknown

Description:

Services should enforce secure communication.

Technical description:

The different components of the system (national services, phone apps and EU gateway) rely on data to be transmitted securely between them. For example the `dgca-revocation-distribution-service` needs to trust some data provided by the `dgca-verifier-service` and also the EU gateway.

HTTPS would prevent modification of data during transport. The provided reference implementations do not specify that they should offer their endpoints via HTTPS only. We checked this for `dgca-revocation-distribution-service` and `dgca-verifier-service`.

They also do not enforce HTTPS when making requests themselves, however they would likely use HTTPS when a HTTPS URL is provided.

Certificate pinning can prevent some attack scenarios arising.

Impact:

Communication between components might be tampered with or read by MitM attackers.

Recommendation:

- Specify that the services should be made available only via HTTPS.
- Ensure in code that only HTTPS is used when connecting to other components.

- Allow for certificate pinning.

3.10 F-004 — Publishing of test URLs

Vulnerability ID: F-004

Vulnerability type: Best practice

Threat level: Low

Description:

Test environment URLs are published in the repositories.

Technical description:

The acceptance environment URLs are published in the code in several places, for example:

- `https://github.com/eu-digital-green-certificates/dgca-wallet-app-ios/search?q=cfapps.eu10.hana.ondemand.com`
- `https://github.com/eu-digital-green-certificates/dgca-verifier-app-android/search?q=cfapps.eu10.hana.ondemand.com`
- `https://github.com/search?q=org%3Aeu-digital-green-certificates+cfapps.eu10.hana.ondemand.com&type=code`

Impact:

This could lead to random people and attackers alike trying to play with test services, which might not be desirable.

Recommendation:

- Change the actual URLs used for testing and do not publish them.
- Use dummy domains (e.g. `example.com` from RFC2606) for documentation purposes.

3.11 F-007 — Data download in dgca-verifier-service

Vulnerability ID: F-007

Vulnerability type: Missing verification

Threat level: Low

Description:

`dgca-verifier-service` performs limited checks on downloaded data.

Technical description:

The `dgca-verifier-service` is essentially a frequently updated national cache for Digital Signer Certificates (DSCs). It downloads the Country Signing Certificate Authority certificates (CSCAs) and country upload certificates from the gateway, and checks them against the configured trust anchor public key. The trust anchor should be set to the gateway public key. It downloads the DSCs from the gateway and checks that they are signed by a CSCA and Upload certs (the latter can be optionally deactivated).

It does not however verify, that:

- The DSC's country code matches the country code of the Upload cert used to sign the CMS downloaded from the gateway. See `fetchTrustListAndVerifyByCscaAndUpload`.
- The DSC's country code matches the country code of the CSCA cert used to sign. See `fetchTrustListAndVerifyByCscaAndUpload`.
- The `TrustListItemDto` country and KID values downloaded alongside the DSC from the gateway match the data in the DSC.

In theory the `dgca-verifier-service` code would accept DSCs for one country being signed by another.

All of the above should not be an issue when used in conjunction with the official DGC gateway as these values are checked before being put in the gateway database by the reference gateway code. However, this is inconsistent. If the developers trust the gateway to do all these checks, why even check if the DSCs are valid locally at all? We recommend assuming as little trust as possible and verifying all data locally.

It's possible that we overlooked the checks in this nested code structure.

Impact:

The `dgca-verifier-service` would accept DSCs for one country being signed by another, and accepts arbitrary KID values from the connected gateway.

Recommendation:

- Assume no trust and implement KID and country code checks in the `dgca-verifier-service`.

3.12 F-010 — Limited documentation

Vulnerability ID: F-010

Vulnerability type: Best practice

Threat level: Low

Description:

Limited documentation hampers creating secure implementations based on the reference implementations.

Technical description:

There is barely any documentation in the provided repositories. While higher-level functions are described by the EU documents [5], there is a general lack of user guidance or more technical documentation. The EU documents sometimes give multiple options and only reading the code reveals what was really implemented.

For example these repositories have no documentation or guidance other than a short introductory text:

- `https://github.com/eu-digital-green-certificates/dgca-wallet-app-android#documentation`
- `https://github.com/eu-digital-green-certificates/dgca-revocation-distribution-service`
- `https://github.com/eu-digital-green-certificates/dgca-verifier-app-android`

This is combined with "surprise" features such as local data encryption in the apps or certificate pinning. These are good security features, but without documentation implementers would have to read the code to find out these measures even exist. They increase security, but why and with which security goals in mind these features were added is not documented.

The reference implementations are used for critical (security) services and only having the code as documentation is a weak starting point for all subsequent development.

Impact:

Secure development is hampered by missing documentation.

Recommendation:

- Provide documentation to help implementers: Document inner workings, additional features, and deviations from reference documents.

3.13 F-016 — Lookup proof

Vulnerability ID: F-016

Vulnerability type: Missing feature

Threat level: Low

Description:

No lookup proof implemented.

Technical description:

The `dgca-wallet-app-android` checks its own DCC and knows its own revocation status.

It however lacks the ability to prove that to a `dgca-verifier-app-android`, whose bloom filter, by design, contains false positives. To trap these rare occurrences, the reference document [3] describes Lookup Proofs:

> 4.8.6. Lookup Proof Normally the lookup of certificate revocation status is normally made by the holder in situations where any kind of check fails (e.g. at the Border). In these situations, the verifier needs to be sure that the challenge of the holder was justified. This can be realised over generating lookup proofs which are signed by the revocation list provider. The best lookup proof would be in this case a new issued DCC or a token provided by a validation service to fulfil the claim of compensation.

We did not find this or a compensating measure implemented in the reviewed code. The reference document states:

The probability for a false positive is 1/10^10

Impact:

Some valid DCCs may be misidentified as invalid and holders may be denied access to e.g. flights or borders.

Recommendation:

- Implement the lookup proof described in the reference documents or compensation measure.

3.14 F-018 — Wallet app data storage

Vulnerability ID: F-018

Vulnerability type: Best practice

Threat level: Low

Description:

`dgca-wallet-app-android` user authentication is not coupled to stored data.

Technical description:

The `dgca-wallet-app-android` asks the user for authentication when starting. This authentication is required to progress to the main menu.

The application stores data using keys protected by the keystore (see the `KeyStoreCryptor` interface). The application also has a key that is used to sign revocation lookups (see the `JwtTokenGenerator` interface).

However, the keys are not coupled to the authentication results and do not have user authentication requirements. That allows advanced attackers instrumenting the application to extract the data, by just calling the data decryption function directly. They are not stopped by the login screen for the app.

See also: https://labs.f-secure.com/blog/how-secure-is-your-android-keystore-authentication/

Impact:

Advanced local attackers do not need user presence to extract data. Data encryption is not implemented according to best practice.

Recommendation:

- Connect `dgca-wallet-app-android` logins to data keystore keys.

3.15 F-019 — Wallet-app login dialog

Vulnerability ID: F-019

Vulnerability type: Missleading interface

Threat level: Low

Description:

The `dgca-wallet-app-android` login dialogue wrongly claims to be biometric when it is not.

Technical description:

The `dgca-wallet-app-android` is protected by a login dialogue. The dialogue depends on the device features protected by biometrics or another device unlock method, but it always claims to be a biometric login even on devices that don't support biometrics.

The class `AuthFragment`:

```
val prompt = BiometricPrompt.PromptInfo.Builder()
    .setTitle(getString(R.string.biometric_dialog_title))
    .setSubtitle(getString(R.string.biometric_dialog_subtitle))
```

always uses these hardcoded values:

```
<string name="biometric_dialog_title">Biometric login</string>
<string name="biometric_dialog_subtitle">Log in using your biometric credential</string>
```

This security feature is advertised even when it is not present, giving a false sense of security.

Impact:

`dgca-wallet-app-android` login suggests it has biometric security on non-biometric devices.

Recommendation:

- Change the login screen text and design accordingly on non-biometric devices in the `dgca-wallet-app-android`.

3.16 F-020 — ECB

Vulnerability ID: F-020

Vulnerability type: Best practice

Threat level: Low

Description:

`dgca-wallet-app-android` uses an ECB encryption scheme.

Technical description:

The `dgca-wallet-app-android` encrypts internal data, the QR code and TAN using the keystore. The chosen encryption scheme in class `SecurityKeyWrapper` and `DefaultKeyStoreCryptor` is ECB.

ECB produces identical encrypted data and is thus not recommended for multi-block data.

There is no documentation of the security model this security feature is intended for, so it's not possible to say whether the feature is faulty. While potentially not a meaningful issue for the presented data, it's not best practice and might become an issue later if implementations extend the use case beyond the reference implementations.

Impact:

`dgca-wallet-app-android` data storage encryption scheme leaks protected data slightly.

Recommendation:

- Use a more secure encryption scheme in the `dgca-wallet-app-android`.

3.17 F-021 — AES-GCM

Vulnerability ID: F-021

Vulnerability type: Crypto

Threat level: Low

Description:

The `dgca-verifier-app-android` uses AES-GCM with an hardcoded IV.

Technical description:

The `dgca-verifier-app-android` uses AES-GCM to protect the DSCs used to verify DCCs. To fulfil its security promises AES-GCM messages need to be created with a unique, never-to-be-repeated initialisation vector, but the `dgca-verifier-app-android` uses a hard coded IV.

There also no documentation of the security model this security feature is modelled for, so it's not possible to say whether this feature is faulty. While potentially not a meaningful issue for the use case, given that DSCs are essentially public information, it is not best practice and might become an issue later if implementations extend the use case beyond the reference implementation.

See class `SecurityKeyWrapper`:

```
private fun getCipher(mode: Int) = Cipher.getInstance(AES_GCM_NO_PADDING).apply {
    init(
        mode,
        secretKey,
        GCMParameterSpec(128, AES_GCM_NO_PADDING.toByteArray(), 0, 12)
    )
}

companion object {
    private const val AES_GCM_NO_PADDING = "AES/GCM/NoPadding"
}
```

See https://crypto.stackexchange.com/questions/26790/how-bad-it-is-using-the-same-iv-twice-with-aes-gcm.

Impact:

The secure storage leaks key(-stream) and plaintext data.

Recommendation:

- Use AES-GCM properly with unique IVs in the `dgca-verifier-app-android`.

3.18 F-022 — Origin country of a rule

Vulnerability ID: F-022

Vulnerability type: Missing verification

Threat level: Low

Description:

`dgca-businessrule-service` does not verify the origin country of a rule.

Technical description:

The `dgca-businessrule-service` checks that rules downloaded from the DGC gateway are signed by a trusted certificate DSC. However, it does not verify that the signature is from the same country that the rules apply to and/or the country that it requested information for.

See class `DgcGatewayValidationRuleDownloadConnector`:

```
return trustedUploadCertificates
.stream()
.anyMatch(uploadCertificate::equals);
```

All the above shouldn't be an issue when used in conjunction with the official DGC gateway, as these values are checked before being stored in the gateway database by the gateway code.

However this is inconsistent. If the developers trust the gateway to do all these checks, why then even check if the data is signed locally at all. We recommend assuming as little trust as possible and verify all data locally.

Impact:

The `dgca-businessrule-service` would accept rules for one country being signed by another.

Recommendation:

- Verify in the `dgca-businessrule-service` that each rule is signed by the country it applies to.

3.19 F-023 — Public key offered unsecured

Vulnerability ID: F-023

Vulnerability type: Integrity protection

Threat level: Low

Description:

`dgca-businessrule-service` offers its public key for signature function over an untrusted connection.

Technical description:

The `dgca-businessrule-service` contains the option to sign data that it made available via its endpoints. There is no documentation to say what this feature is for, but it makes sense in case the data needs to be transmitted over an unsecured connection.

There is an endpoint `/publickey` available that can be used to download the public key that signs the data.

This is a bit odd assuming an unprotected connection, then it is also not safe to transfer the public key that way. It could used securely e.g. by hard-coding the public key into the apps instead. We mark this as a finding given there is no documentation how this mechanism is supposed to be used and it implies it is intended to be used insecurely.

The `dgca-wallet-app-android` and `dgca-verifier-app-android` reference implementation do not use this mechanism and it is also not required because they use HTTPS with certificate pinning instead in the example configuration.

Impact:

The trust chain used to verify rules given by the `dgca-businessrule-service` is broken.

Recommendation:

- Provide documentation about the signing feature of the `dgca-businessrule-service` so it can be used correctly.

- Make the public key available in a hard-coded or verifiable way instead of via an unprotected endpoint.

4 Non-Findings

In this section we list non-findings that do not directly result in a vulnerability.

4.1 NF-006 — DoS protection

The provided national service reference implementations do not offer DoS protection. They are intended to offer most services without authentication. The users of these reference implementations are required to provide adequate DoS protection themselves.

This is not marked as a finding because this is to be expected of such a source-code-only reference implementation, but noting this in documentation would help.

4.2 NF-009 — dgca-issuance-service does not restrict key and signature algorithms

The wallet app submits a claim on a DCC using the `/claim` endpoint of the `dgca-issuance-service`. The `claimRequest` contains both the public key algorithm and signature algorithm used. The function `verifySignature` imposes no restrictions. Only the later function `asJwk` restricts it for the key algorithm only. The services should restrict choices to the agreed-upon algorithms from the reference documents.

We did not find any way for this to have a negative impact on others, should an attacker or legitimate wallet user willingly select an odd algorithm for their own key and signature.

4.3 NF-011 — dgca-issuance-service unused DCC properties

The `DgciEntity` class in the `dgca-issuance-service` project contains the following fields:

```
@Column(name = "revoked")
    private boolean revoked;

   ...

    @Column(name = "locked")
    private boolean locked;
```

The `locked` name implies that it might be set once all 3 attempts to claim a DCC have been used. Similarly, `revoked` implies it would be set when a DCC has been revoked.

However, the state of `revoked` is never be set or read. The state of `locked` is checked by the `HEAD /dgci/{dgciHash}` endpoint, but is never set.

While not a vulnerability these states should not exist if they do not accurately describe the real state of the DCC.

5 Future Work

- **Continuous retests**
  The references are in active development. Future changes might introduce vulnerabilities too, so we recommend retesting regularly.
- **Extended scope**
  Not all reference repositories were covered by this time-boxed evaluation. Another evaluation could focus on the remainder, although we tried to select the most relevant repositories, covering all flows once. For example, we did not evaluate the iOS apps, which to a large degree likely offer similar functionality, but might also have specific potential vulnerabilities.
- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed or documentation has been written, a repeat evaluation could be performed to ensure their quality.

6 Conclusion

We evaluated the reference implementations intended to be used as the basis for implementations of countries' digital COVID certificate infrastructure.

There is a general lack of documentation and guidance. The services have to be deployed in a secure manner to work as intended. Implementers have to dig deep into the source code to be able to develop properly. The general design is given by the reference documents but plenty of choices and some security features were added by the developers without documenting them.

We found issues in several different categories:

- Limitations that implementers have to be aware of such as limited documentation, hidden features, need to change default passwords, DoS protection, and so on.
- Security features not being implemented according to best practices, or insecurely
- Lack of verification of data coming from other countries, or various other components in the system. However, if it's assumed that member states do not act maliciously and the official DGC gateway is being used, many issues don't apply.
- A way to reclaim DCC certificates that is a deviation from the reference documents, but in most cases would require a local attacker to have control of the wallet app as a prerequisite.
- An authentication bypass, allowing lookup of anybody's revocation status. The impact is limited given that this information could also be obtained quite freely using the verifier app.
- An unimplemented "Lookup Proof" feature, that is required in rare cases to prove that one has a valid DCC that was recognized as invalid e.g. at borders or airports.

We discovered 7 Moderate, 11 Low and 1 Unknown-severity issues during this penetration test. Overall it is hard to say what the actual impact of these findings really is as it all depends on how these reference implementations are used in final products.

We recommend addressing all of these findings, provide complete developer documentation, implement missing checks, follow best practices in order to increase the overall likelihood of producing secure end-products. A minor issue here could become a severe issue in a derived implementation.

Finally, we want to emphasize that security is a process – this evaluation is just a one-time snapshot. Security posture must be continuously evaluated and improved.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

7 Bibliography

[1] EU Digital COVID Certificates Github repositories. https://github.com/eu-digital-green-certificates.

[2] Revocation B2B Centralized Version 0.7 2021-12-08.

[3] Revocation B2A Proposal Version 1.0 2022-02-10.

[4] Radically Open Security B.V.: EU Digital COVID Certificates Gateway - Evaluation Report v1.0 2022-03-01.

[5] EC eHealth and COVID-19 documentation. https://ec.europa.eu/health/ehealth-digital-health-and-care/ehealth-and-covid-19_en.

Appendix 1 Testing team

Tim Hummel	Tim Hummel is a senior IT-security analyst, consultant, developer and trainer. His speciality is hardware, crypto, and related software security. In his work he tests everything from apps, car components, payment solutions, white-box crypto, pay TV, mobile devices, IoT, TPMs, TEEs, bootloaders, entertainment systems to transport cards. He recently tested various Corona related apps and services for the EU and the Netherlands.
Robin Peraglie	Robin is a passionate bug hunter and security researcher. Since he was young he experimented with web security, cryptography and lockpicking. He received a M.Sc. degree in IT Security at the Ruhr-University Bochum, and has since gained industrial experience across many penetration tests and professional code audits.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.