DP3T API

Documentation

October 27, 2020

Contents

Technical Description

Introduction

This document outlines the backend as it is. The models and requests are automatically generated. Hence, they should reflect the current live situation. We try to provide examples and description to clarify the use of the fields and responses returned.

Verification of Data

To handle heavy workload, requests are routed via a content-delivery-network (CDN). This means that we need to provide proof that the data was not modified by the CDN. We propose a Elliptic-Curve Digital Signature Algorithm using the P256 elliptic curve with a SHA-256 hashing algorithm. The P256 elliptic curve has good native support for the Apple and Android platforms to verify signatures. The public key should be available on the discovery platform and is as well included and distributed with the applications for iOS and Android.

The ensure the possibility of signature verification, the signed endpoints return an object with a signature and a data field, of which the data field contains a base64 representation of the list. In the current implementation the representation is a json of a list of keys. To improve performance of possible large decodings, we plan to switch to protobuf or something similar, which should speed up the parsing.

Since we only want to ensure that the data we are processing was indeed the data sent from the specified backend, it is sufficient to generate the signature of the content which will be processed.

Too further improve operability, the algorithm used to generate the signature should as well be encoded within the json object, similiar to a JWK (Json web key).

Google/Apple Privacy-Preserving Contact Tracing Similarities

Web Service

Introduction

A test implementation is hosted on: https://demo.dpppt.org.

This part of the documentation deals with the different API-Endpoints. Examples to fields are put into the models section to increase readability. Every request lists all possible status codes and the reason for the status code.

/v1/gaen/

```
get /v1/gaen/
```

Hello return

Responses

200 Success

server live

Type
string

/v1/gaen/exposed

```
post /v1/gaen/exposed
```

Send exposed keys to serverincludes a fix for the fact that GAEN doesn39;t give access to the current day39;s exposed key

Request Headers

Field	Type	Description
User-Agent *	string	App Identifier (PackageName/BundleIdentifier) + App-Version + OS (Android/iOS) + OS-Version

Request Body

Field	Type	Description
*	GaenRequest	The GaenRequest contains the SecretKey from the guessed infection date, the infection date itself, and some authentication data to verify the test result

Responses

200 Success

The exposed keys have been stored in the database

Type
string

400 Bad Request

Invalid base64 encoding in GaenRequest

403 Forbidden

Authentication failed

/v1/gaen/exposednextday

- `post /v1/gaen/exposednextday`

Allows the client to send the last exposed key of the infection to the backend server. The JWT must come from a previous call to /exposed

Request Headers

Field	Type	Description
User-Agent *	string	App Identifier (PackageName/BundleIdentifier) + App-Version + OS (Android/iOS) + OS-Version

Request Body

Field	Type	Description
*	GaenSecondDay	The last exposed key of the user

Responses

200 Success

The exposed key has been stored in the backend

Type
string

400 Bad Request

- Ivnalid base64 encoded Temporary Exposure Key- TEK-date does not match delayedKeyDAte claim in Jwt

403 Forbidden

No delayedKeyDate claim in authentication

/v1/gaen/exposed/keyDate

```
get /v1/gaen/exposed/{keyDate}
```

Request the exposed key from a given date

Query Parameters

Field	Type	Description
publishedafter	integer	Restrict returned Exposed Keys to dates after this parameter. Given in milliseconds since Unix epoch (1970-01-01).

Path Parameters

Field	Type	Description
keyDate *	integer	Requested date for Exposed Keys retrieval, in milliseconds since Unix epoch (1970-01-01). It must indicate the beginning of a TEKRollingPeriod, currently midnight UTC.

Responses

200 Success

zipped export.bin and export.sig of all keys in that interval

This request returns **application/zip** . This represents zipped export.bin and export.sig of all keys in that interval .

500 Internal Server Error

- invalid starting key date, doesn39;t point to midnight UTC- publishedAfter is not at the beginning of a batch release time, currently 2h

/v1/gaen/buckets/dayDateStr

`get /v1/gaen/buckets/{dayDateStr}`

Request the available release batch times for a given day

Path Parameters

Field	Type	Description
dayDateStr *	string	Starting date for exposed key retrieval, as ISO-8601 format

Responses

200 Success

zipped export.bin and export.sig of all keys in that interval

Type
DayBuckets

500 Internal Server Error

invalid starting key date, points outside of the retention range

/v2/gaen/

`get /v2/gaen/`

Hello return

Responses

200 Success

server live

Type
string

/v2/gaen/exposed

`post /v2/gaen/exposed`

`get /v2/gaen/exposed`

Requests keys published after lastKeyBundleTag. The response includes also international keys if includeAllInternationalKeys is set to true. (default is false)

Request Headers

Field	Type	Description
User-Agent *	string	App Identifier (PackageName/BundleIdentifier) + App-Version + OS (Android/iOS) + OS-Version

Query Parameters

Field	Type	Description
countries	string[]	List of origin countries of requested keys. (iso-3166-1 alpha-2).
lastKeyBundleTag	integer	Only retrieve keys published after the specified key-bundle tag. Optional, if no tag set, all keys for the retention period are returned

Request Body

Field	Type	Description
*	GaenV2UploadKeysRequest	JSON Object containing all keys.

Responses

200 Success

The exposed keys have been stored in the database

Type
string

400 Bad Request

- Invalid base64 encoding in GaenRequest- negative rolling period- fake claim with non-fake keys

403 Forbidden

Authentication failed

200 Success

zipped export.bin and export.sig of all keys in that interval

This request returns **application/octet-stream** . This represents zipped export.bin and export.sig of all keys in that interval .

500 Internal Server Error

Invalid lastKeyBundleTag

Models

All Models, which are used by the Endpoints are described here. For every field we give examples, to give an overview of what the backend expects.

DayBuckets

Field	Type	Description	Example
dayTimestamp	integer	The day of all buckets, as midnight in milliseconds since the Unix epoch (1970-01-01)	1593043200000
day	string	The day as given by the request in /v1/gaen/buckets/dayDateStr	2020-06-27
relativeUrls	string[]	Relative URLs for the available release buckets	

GaenKey

Field	Type	Description	Example
keyData *	string	Represents the 16-byte Temporary Exposure Key in base64	
rollingStartNumber *	integer	The ENIntervalNumber as number of 10-minute intervals since the Unix epoch (1970-01-01)	
rollingPeriod *	integer	The TEKRollingPeriod indicates for how many 10-minute intervals the Temporary Exposure Key is valid	
transmissionRiskLevel *	integer	According to the Google API description a value between 0 and 4096, with higher values indicating a higher risk	
fake	integer	If fake = 0, the key is a valid key. If fake = 1, the key will be discarded.	

GaenRequest

Field	Type	Description	Example
gaenKeys *	GaenKey[]	Between 14 and 30 Temporary Exposure Keyszero or more of them might be fake keys. Starting with EN 1.5 it is possible that clients send more than 14 keys.	
delayedKeyDate *	integer	Prior to version 1.5 Exposure Keys for the day of report weren't available (since they were still used throughout this day RPI=144), so the submission of the last key had to be delayed. This Unix timestamp in milliseconds specifies, which key date the last key (which will be submitted on the next day) will have. The backend then issues a JWT to allow the submission of this last key with specified key date. This should not be necessary after the Exposure Framework is able to send and handle keys with RollingPeriod < 144 (e.g. only valid until submission).	

GaenSecondDay

Field	Type	Description	Example
delayedKey *	GaenKey		

GaenV2UploadKeysRequest

Field	Type	Description	Example
countries	string[]	List of countries of interest.	
gaenKeys *	GaenKey[]	30 Temporary Exposure Keyszero or more of them might be fake keys.	