Tracking Keys in COVID Tracking Apps

Liam Ó Lionáird

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Computer Science

Supervisor: Stephen Farrell

April 2024

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Liam Ó Lionáird

April 15, 2024

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Liam Ó Lionáird

April 15, 2024

Tracking Keys in COVID Tracking Apps

Liam Ó Lionáird, Master of Computer Science

University of Dublin, Trinity College, 2024

Supervisor: Stephen Farrell

Digital contact tracing apps were a common tool for combating the spread of COVID-19 on a national scale, but their true efficacy in the wake of the pandemic is still unclear. The most prominent framework used, Google/Apple Exposure Notification (GAEN), employed a decentralised solution involving the exchange of cryptographic identifiers (TEKs) between users' devices to identify infectors and notify infectees; but despite mass adoption in dozens of apps, along with international backend networks for sharing TEKs across countries (as in Europe), there is still a lack of research data to support in-depth large-scale evaluations of this system. This project adapts a previous multi-year survey of TEKs from 32 regions into a concise yet comprehensive database, by tracking and processing all unique TEK instances throughout the survey. The final database is easily queriable for insights into TEK metadata differences between regions, as well as for tracking individual TEKs across space and time—revealing many intriguing possible avenues of analysing GAEN's ultimate effectiveness.

Acknowledgments

This dissertation owes its entire genesis to the work of Prof. Doug Leith and Dr. Stephen Farrell in their Testing Apps for COVID-19 Tracing (TACT) project at TCD, where the data used in this project was thoughtfully gathered for future research—I thank them for both their significant contribution and the opportunity to build on it. Dr. Farrell gave invaluable further guidance, knowledge, and encouragement as my dissertation supervisor, and has been as engaging and inspiring as a teacher could hope to be; I must also acknowledge his kind supervision of April Sheeran, who herself conducted important Masters research with the same data and who shared fruitfully in our discussions.

Throughout this work, I was assisted generously in life by my long-time Dublin benefactors Michael and Mary Telford; by my dearest friends Lydia MacBride and Owen Gallagher, and all who share their friendship; and by my wonderful family, who (though less literally post-pandemic) are always with me, and whom I thank for everything I am.

Liam Ó Lionáird

University of Dublin, Trinity College
April 2024

Contents

Chapter 1

Introduction

They say the worst is done
And it's time to find out what we've all become...

—Weyes Blood

While the emergency phase of the COVID-19 pandemic has formally concluded, and academic analysis of our global response has shifted from reactive to retrospective, the importance of such analysis remains as acute as ever. We now possess the data and hindsight to conduct bigger-picture studies of international scale (as has already been done for general COVID policy [1]), enabling stronger evaluations and conclusions for the global community to heed in future. However, studies of digital contact tracing apps—once a linchpin of many COVID policies—have remained sparse in variety, small in scope, and lacking in synthesis[1]. Although a vast amount of relevant data was generated by these apps, and used by health services in their active response, there is still a need to collate and publish this data in a form more useful to researchers going forward. This dissertation project takes a practical step in that direction.

Contact tracing[2] was recommended by the World Health Organisation throughout the COVID-19 emergency as a "key strategy" for containing the spread of infection, but the organisation also acknowledged many challenges to manual tracing, such as delays in identification and notification of close contacts [2]. Digital contact tracing solutions were therefore a popular choice for public health authorities—taking advantage of the ubiquity of smartphones and the easy automation of tracking and notification—with several frameworks for app-based COVID tracking systems being designed and implemented worldwide in the first year of the pandemic.

[1] As the literature review in the 'Related Work' section will later show.

[2] The practice of identifying close contacts of an infected individual and tracking/quarantining them to prevent further infections.

This dissertation project focuses on Google/Apple Exposure Notification (GAEN), the most prominent of these frameworks. Developed by two smartphone tech giants and integrated within their operating systems, it presented itself upon debut in April 2020 as a "comprehensive solution" for building digital contact tracing systems [3]. Its automatic implementation in millions of devices worldwide enabled vast decentralised contact-tracing networks, whereby GAEN apps would handle their own detection of nearby devices using Bluetooth, instead of leaving these duties to a centralised server. With these apps being used to report infections to public health authorities, GAEN would also identify an infectee's device within its records and notify users of possible exposure—enabling easy and effective digital contact tracing en masse.

However, questions and concerns over the national-scale implementation of digital contact tracing, and its efficacy against COVID, were raised by researchers even in these early stages. The GAEN framework faced particular criticism for its closed implementation, its questionable protections of privacy [4], and its decentralised contact tracing model being potentially less effective and ethically sound than the centralised solutions it deterred [5]. As more and more health authorities adopted GAEN for use in their contact tracing apps, these concerns could soon be evaluated with data gathered in practice.

One unique investigation into the GAEN framework was the Testing Apps for COVID-19 Tracing (TACT) project [6], headed by Prof. Doug Leith and Dr. Stephen Farrell of Trinity College Dublin. Initiated in April 2020, it primarily worked to determine the accuracy of GAEN's Bluetooth proximity detection [7]. As a secondary activity, it conducted a survey of GAEN-based apps and their public-facing servers, scraping data published to the servers of 32 apps on an hourly basis from June 2020 to September 2023. This data contained the Temporary Exposure Keys (TEKs) and associated metadata sent by devices after reporting infection, which were used to notify other users of their exposure—the literal keys to GAEN's contact tracing operation. With the dataset now complete, containing over 20,000 hourly snapshots and totalling 2.3 terabytes in size, it represents a sizable potential contribution to GAEN research; however, no study outside TACT to date has made use of this dataset.

1.1 Research Question

Several trends identified over the course of the TACT survey added both intrigue and complexity to the resulting dataset—making it both more valuable and more difficult to extract value from. Early in the survey, it was noted that regions with shared GAEN backends were sharing uploaded data among themselves. One example from October 2020 [8] involved Ireland and Northern Ireland, whose health authorities, though gov-

erned separately, shared the same developer of their apps and backends—hence TEK data uploaded to one region would later be uploaded to the other. (The survey detected a two-hour delay on average.) Similar arrangements employed elsewhere became apparent later—most notably, the European Federation Gateway Service (EFGS), established in October 2020 to enable GAEN backend data-sharing on a continental scale, linked 19 participating countries together in its network [9].

Additionally, the TACT paper cited above presented evidence of 'fake' TEKs being published in certain regions. Where public health authorities did not report these practices directly, they could still be inferred through examining trends in the hourly server snapshots: some TEKs may be published in abnormally large numbers, only to disappear from the server after a fraction of their standard lifetimes. The report assumes that these measures aimed to pad out legitimate TEKs for "privacy reasons" [8], but the dubious success of these measures (and the wildly varying ratios/approaches between regions) warrants further investigation.

Despite its potential value, the dataset as it stands is not perfect in structure or integrity (as will be detailed further), nor does its hour-by-hour redundancy and obfuscated data format make for research-friendly utility. The latter issues can be solved through any number of data-processing approaches, but the best possible approach requires tailoring for the best possible purposes. In seeking such an ideal form and context for the TACT dataset, this project began with the following research question:

Research question: Can a database of TEKs be created which allows easy querying and tracking of TEKs?

A satisfactory answer to this question may empower the answering of further questions relevant to future study of the TACT dataset, and of the GAEN ecosystem as a whole. Such a database would ideally reveal the scope of aforementioned data-sharing between regions and nations in fullest possible detail. From there, spatio-temporal data tracking would illuminate where each TEK came from, where it went, and with what delays or changes inbetween. Already new avenues of policy evaluation present themselves—e.g. if TEKs are shown to take several days on average to move from one region to another, the relevance and utility of TEK sharing in general may be called into question. The potential of a TEK database thus stands to extend the frontier of interesting large-scale studies concerning the implementation, usage, and efficacy of GAEN-based contact tracing initiatives on a national and international level—perhaps reinvigorating the slowly stagnating landscape of research on this topic in recent times.

This project presents a proof-of-concept solution in multiple stages. Firstly, we show that the TEKs stored in the TACT dataset can be tracked through initial extraction

and processing, resulting in an initial condensed dataset that stores all unique instances of each TEK using a fraction of the disk space. Secondly, we introduce a design and implementation of a relational database built on this new dataset. Finally, we demonstrate that this database can be easily used to track TEKs across regions chronologically, as well as enabling further insights on the use of TEK metadata—concluding afterwards with a discussion of these findings and of future work.

Chapter 2

Background

This chapter provides further details on the most relevant technical aspects of this dissertation project—namely the GAEN framework and the TACT dataset—to give further context and justification for the project's motivation, methodology, and results. It also includes a literature review in which the current state of GAEN analysis, and inspirations for the methodology of this project, are discussed.

2.1 Google/Apple Exposure Notification (GAEN)

The GAEN framework was jointly announced by Google and Apple in April 2020 [3], and the first national COVID tracking app using the framework (Switzerland's 'SwissCovid') was deployed two months later [10]. GAEN is notable among COVID-era digital contact tracing frameworks for its relatively wide adoption and its Big Tech lineage, as well as its operational details, which will be sketched below.

Being created by the developers of iOS and Android, GAEN was launched as a component of both operating systems—providing a built-in OS-level API for COVID tracking apps to access, rather than implement individually. This design decision, though controversial in locking down access, served to secure operational consistency across tracker apps, and it is also relevant in proving the integrity of the GAEN-derived data being used in this dissertation project—all data is generated and transmitted in the same manner, following the same data specification, and using the same GAEN implementation deployed on all iOS and Android phones. (There is still variance to be found between regions in the usage of metadata fields defined by the GAEN specification; this will be discussed in the 'Results' chapter.)

2.1.1 GAEN in Practice

In keeping with Google/Apple's stated commitment to preserve privacy, GAEN "is designed so that information about a positive diagnosis or potential exposure does not include identifying information." [11] Instead of sending names and locations, pseudo-random identifiers are cryptographically derived, changed regularly within the device, and exchanged with nearby devices using Bluetooth Low Energy (BLE). These identifiers can be traced back to the device that generated them, but only symbolically, and their use is intended only for other devices that have been in close contact. Such a decentralised detection system is the basis of GAEN's design and function.

The most important identifier within the GAEN ecosystem (and the chief unit of data within this project) is the Temporary Exposure Key (TEK). Each TEK is a 16-byte number, derived by a cryptographic random number generator function (left undefined in the GAEN spec [12]). Every 24 hours, a GAEN device generates a new TEK and stores it with a timestamp.[3] TEKs have a lifespan of 14 days before being deleted; thus a maximum of 14 TEKs are held at a time.

TEKs are not used directly in day-to-day proximity tracking—a more temporary identifier, the Rolling Proximity Identifier (RPI), is derived from the most recent TEK using a HMAC-based key derivation function, and is replaced each time the device's Bluetooth MAC address changes, "to prevent linkability and wireless tracking" [12]. It is these RPIs that are exchanged between devices via BLE when within an app-defined range of each other. Crucially, these RPIs can be used to re-derive the original TEK of the device that generated them.

Only when a device user reports a COVID infection do the device's TEKs come into public use. All TEKs stored on the infectee's device are transmitted to the public health authority's Diagnosis Server, each one bundled with app-specific metadata such as report type or risk level. The Diagnosis Server stores all such reported TEKs from all devices using the given GAEN app (usually for 14 days, keeping with the TEKs' lifespans). Each day's new TEKs are batch-exported into ZIP files, which are posted on the Diagnosis Server's public URL directory.[4] Each GAEN app client periodically fetches new TEK export ZIPs from this URL. These TEKs can be compared, through derivation of the RPIs exchanged with the client, to the TEKs of each device it has previously recorded. If a match is found, the client notifies the user of a potential close contact with an infected person; relevant COVID policies are asserted from here.

[3] Both the timestamp and the 24-hour rolling period are specified in 'rolling intervals' of 10 minutes—the former is converted to a UNIX timestamp for this project's purposes.

[4] These directories are maintained by the app's developers and often hardcoded into the apps—TACT's paper on GAEN privacy [4] lists raw HTTP GET requests used by several GAEN apps.

2.1.2 The TEK Export ZIP

Each TEK export ZIP contains two files: an ECDSA signature[5] verifying the payload (`export.sig`), and the payload itself (`export.bin`).

The data within each payload is not stored in plain text—it is serialised with the Google-developed Protocol Buffer ('Protobuf') data format. The structure of a Protobuf file is pre-defined (in this case by Google) in a '`.proto`' specification, which is used to unpack and access the data within. The Protobuf data itself is defined as a list of structs and fields containing strings, numbers, or enums, similar to variables in code, and the Protobuf specification can be 'compiled' in most programming languages to produce a class of functions for interfacing with this data in programs.

Viewing the original Protobuf specification[6] is the best way to understand how the TEK export ZIP is structured and interfaced with—however the most important elements and quirks will be detailed here for emphasis.

The specification begins with start/end timestamps marking the arrival window of TEKs contained in the ZIP's payload. It then defines a list of `TemporaryExposureKey` structs, allowing any number of TEKs to be stored in a given payload. Each struct contains a single TEK and all of the metadata associated with it. A summary of the `TemporaryExposureKey` struct's contents follows:

- `key_data` (`char[]`): The TEK itself, stored in raw bytes.

  (The byte array is not fixed-size, allowing malformed TEKs of different length than 16 bytes. See Appendix for such examples found in the TACT dataset.)

- `transmission_risk_level` (`int32`): Risk level associated with the TEK—defined in earlier specifications as an integer range from 1 to 8.

  (Interpretation and implementation of risk level was left to national health authorities, making it less useful as an empirical risk indicator, and the field was eventually deprecated.)

- `rolling_start_interval_number` (`int32`): Timestamp of the TEK's original generation, measured in 10-minute 'rolling intervals'.

[5]The public keys used by GAEN apps to verify these signatures have not been located. This project assumes that each TEK ZIP in the TACT dataset has a valid signature.

[6]Google has taken down most of their GAEN documentation, including the export Protobuf specification—however it remains publically viewable in the source code for their sample GAEN server at `https://github.com/google/exposure-notifications-server/blob/main/internal/pb/export/export.proto`.

- `rolling_period` (`int32`): The duration for which the TEK is valid (before a new one is generated), measured in 10-minute 'rolling intervals'. The default value is 144, i.e. 24 hours.

- `days_since_onset_of_symptoms` (`signed int32`): Number of days between the TEK's generation and the user's reported onset of symptoms.

- `report_type` (`ReportType enum`): Custom data type representing the type of infection report made by the user. Values can be 'Unknown', 'Confirmed test', 'Confirmed clinical diagnosis', 'Self report', 'Recursive' (reserved for future use), or 'Revoked'.

  ('Unknown' and 'Revoked' types are not returned by the client API; they are designated by the Diagnosis Server as needed.)

- `variant_of_concern` (`VariantOfConcern enum`): Custom data type representing levels of concern associated with the reported COVID-19 variant. Values can be 'Unknown' (default), 'Vaccine is effective', 'Highly transmissive', 'High severity', or 'Vaccine breakthrough'.

  (These definitions are only stored as comments in Google's Protobuf specification, leaving their final interpretations unclear. In the end, this field went unused by every region in the TACT survey, as the 'Results' chapter will discuss.)

A second list of TEK structs is defined in the specification for storing 'revised keys' that have been revoked or otherwise changed. This was rarely used by apps, but still processed along with the main list of TEKs for this project.

The Protobuf specification was updated a handful of times in GAEN's first year of implementation, adding new metadata types for use by health authorities. (These fields are prefixed with '`optional`', thus early TEK zips without later metadata are still backwards-compatible with the latest specification.) According to Google's 'Exposure Notification Server' GitHub release logs [13], the `variant_of_concern` field was the latest addition, in August 2021. However, each health authority would need to update its own Diagnosis Server to replace old Protobuf specifications with new ones, and it is unknown how common this practice was.

2.2 The TACT Dataset

The TACT dataset comprises 27,850 hourly snapshots of content from the Diagnosis Servers of 32 regions with GAEN-deployed apps—calculated by this project to contain

143,521,117 unique TEKs. These were downloaded through the Diagnosis Servers' official public URLs, which were reverse-engineered by TACT "based on examination of the open-source client implementations" and additional network traffic analysis [8]. Each snapshot is stored in its own timestamped folder (titled in '`YYYYMMDD-HHMMSS`' format), which contains TEK export ZIP files, JSON configuration files from Diagnosis Servers, and HTTP headers (in `.txt` form) from the GET requests made to Diagnosis Servers. ZIP files are prefixed with a code representing their region of origin, usually the region's internet TLD (e.g. '`at`' for Austria, '`za`' for South Africa)—a complete table of regions and codes can be found in the Appendix.

Because new Diagnosis Servers were only added to the survey after their deployment, with many being operational for weeks beforehand, the first 14 days of data from each region may not show accurate initial upload dates for their ZIPs—in these cases the timestamps included in the ZIPs themselves are a more accurate measurement.

Accompanying the hourly snapshots is the `/all-zips` folder. Throughout the survey, the largest instance of each ZIP recorded in the hourly snapshots was copied into this folder, producing a condensed version of the complete TEK dataset. In an additional de-duplication effort, many early ZIP files were replaced with symlinks pointing to their corresponding (identical) files in `/all-zips`. Though the folder is not used directly for this project, it has been kept in the dataset for the purpose of preserving these symlinks.[7]

It should also be noted that counting unique TEKs from the `/all-zips` folder gives a slightly smaller result than counting from the full hourly snapshot collection—a discrepancy of around 6 million TEKs. It is possible that TEKs were removed from ZIPs across updates to the Diagnosis Servers, mostly likely test TEKs.

A summary of the TACT dataset's total coverage and data integrity can be found in the Appendix. Here it suffices to conclude that the TACT dataset in its raw form is not ideal for direct application in research. With ZIP files meant to remain on Diagnosis Servers for up to 14 days, there are up to hundreds of copies of each ZIP file stored in the dataset's hourly snapshots, with the de-duplication effort mentioned above left incomplete (files after April 2022 remain duplicated) and impractical due to the potential of ZIP files to change in contents over time. The ZIP files themselves, being doubly obfuscated by both compression and serialisation, require far more hassle to work with compared to plain-text files or database tables. It will take much extraction and processing to bring the valuable data contained within to the light it deserves.

[7] Originally the symlinks pointed to absolute filepaths—these were converted with a Bash script into relative links for use by the author. This was the only direct manipulation of the TACT dataset made as part of this project.

2.3 Related Work

The literature review conducted for this dissertation project focused primarily on the current landscape of GAEN-based COVID tracker app research, with an eye towards large-scale studies and usage of TEK data similar to TACT's.

The scope of studies utilising the TACT dataset is restricted to papers from the TACT project itself—chiefly the 'October 2020 Survey of GAEN App Key Uploads' [8] in which an overview of TACT's survey effort is described along with preliminary statistics. An earlier paper from June 2020 [14] introduces the survey's tooling as a possible model for GAEN app transparency, with initial statistics from the first month of GAEN deployment. Given that the survey continued for three more years, these papers obviously no longer represent the total scope of the data gathered. A third TACT paper focusing on TEK statistics [15] only uses Irish data from October 2020 to April 2021. This is the complete extent of research using the TACT dataset as of this project.

There remains a gap—of both scale and synthesis—in COVID tracker app research literature that a comprehensive TEK database may help fill; this can be shown through a brief survey of digital contact tracing app studies since 2020. The European Commission published a notably intensive study in November 2022 [9] examining the 27 contact tracing apps deployed by European countries during the pandemic. It remarks that health authorities' own evaluations of their apps "are rather few and lack standardised assessment methods"—the cited evaluations generally use no more than a year of data.

A review of reviews of COVID tracker apps [16] was published in January 2024, and remains the only one to date. It examined two dozen reviews, finding the majority to be of low quality, with "methodological deficits" leading to a general weakness of evidence for their conclusions. Notably, the frequency of reviews drops off sharply past mid-2021, reflecting both the lower sense of urgency in the late phase of the COVID emergency and perhaps a dwindling supply of data to draw from. Another general comparison of 73 COVID tracker apps by Kesarev and Korochkin [17] restricts itself to a high-level privacy and policy evaluation, explaining that a "lack of data" makes it a far harder task to "assess and compare the effectiveness of these applications."

Other smaller-scale studies, however, have showcased the usefulness of such data in evaluating the performance of individual apps. Vaudenay and Vuagnoux's review of Switzerland's SwissCovid app [10] is a vigorous, detailed critique of both the GAEN framework and the app's implementation, driven in part by GAEN usage statistics published by the public health authority maintaining the app (such as real/fake infection reports). The aforementioned TACT paper on the Irish Covidtracker app [15] uses daily TEK upload statistics effectively to draw conclusions on the Irish GAEN app's low over-

all usage relative to reported infections. There is clear potential for similar studies to be made with apps in other regions—if data is available.

An extra investigation was made into previous databases for cryptographic keys similar to TEKs. The most prominent examples were for HTTPS certificates—first collected at the IPv4 address space scale in 2013 by Durumeric et al. [18], producing a comprehensive dataset maintained by their company Censys [19] and accessible to researchers studying the HTTPS certificate ecosystem. Durumeric et al. used a PostgreSQL database to store historical certificate data of scanned hosts, inspiring the use of PostgreSQL in this project to store historical TEK data.

Chapter 3

Methodology

A two-phase methodology is used for this project. Firstly, the TACT dataset is processed into a condensed form, listing every unique instance of every unique TEK in the dataset. Secondly, this condensed dataset is used to populate a complete relational database, designed to be easily queriable for information and insights on the data within. The first phase produces actionable material on its own, perhaps for purposes beyond implementation of the second phase; it is hoped that such future use will be found in both phases of this project.

All tools and scripts developed for this project are included in the `scripts.zip` file submitted with this dissertation. The original GitHub repository containing the complete code for this dissertation may also be accessed on request.

3.1 TEK Extraction and Processing

The first development phase of this project concerns the extraction of TEKs from the bulk export ZIPs comprising the TACT dataset, and the further processing of these TEKs (and their occurrences/alterations) into a middle-stage condensed dataset, which is used later to populate the final TEK database.

3.1.1 Development

The tooling used throughout this project was developed from scratch by the author. The TACT project which produced the TEK dataset had previously developed a suite of small scripts to extract and process TEKs, written in Bash and Python, but the function of each was limited in scope, not relevant to the task of this project, or thought to be inefficient at processing the sheer amount of data at hand. A single comprehensive solution was sought—hence the original direction taken.

While Python development was considered in the project's early stages, this was eventually abandoned in favour of using Rust. Given the large amount of data to process, and the small individual sizes of the TEK ZIPs in question, native code speeds and thorough memory safety were sought—Rust provided both of these benefits, along with a streamlined build system (Cargo) enabling fast development time and easy iteration.

3.1.2 The `key_extract` Tool

`key_extract` is a Rust command-line tool that outputs a CSV-formatted list of unique TEKs from a specified region, tracking first/last appearances and encoding all unique instances of each TEK throughout the TACT dataset.

The tool assumes a locally-stored instance of the TACT dataset, and accepts two forms of input (alongside a region code):

1. `-d <FOLDER PATH>`: Either the full dataset folder or a single hourly snapshot folder within the dataset.

   (While useful in the latter case for quickly testing a small sample of the dataset, the need to traverse the full directory and verify each file's type before proceeding makes this option prohibitively slow for processing the full dataset, which contains over 22.5 million files in total.)

2. `-p <FILE PATH>`: A text file containing a sorted list of TEK ZIP filepaths.

   (This list can be generated from the dataset directory with the GNU `find` tool piped through `sort`; a Bash script `get-zip-paths.sh` was written to automate this task. This was found to be the far more efficient input solution of the two—it takes mere seconds to filter through the list for ZIPs from the specified region. When both '`-d`' and '`-p`' are invoked, the latter takes priority.)

A verbose flag (`-v`) can also be passed to display various error messages while processing TEK ZIPs.

The tool prints a dynamic progress bar (and any verbose info) to standard error, and prints CSV data to standard output—this can be redirected to save results as a text file. All combined, an example invocation '`key_extract ie -p paths.txt > ie.csv`' will process all Irish ZIPs listed in the path file at `paths.txt` and save the resulting CSV data to `ie.csv`.

3.1.3 TEK Extraction

The first step taken by `key_extract` is to confirm the validity of the arguments passed to it. A hardcoded list of valid region codes is kept in memory to check against the code passed as the first argument; the program will fail instantly if an invalid region is passed. It will also fail if neither '`-d`' nor '`-p`' are called, or if either argument points to an invalid filepath.

Past these initial checks, the program branches depending on the choice to use a directory or a path file. If a directory is used, the program will search using a glob pattern for relevant ZIPs in the given directory. The TACT survey scripts prepend each ZIP filename with its region of origin; hence we can assume, for example, every Irish ZIP filename in a directory will begin with '`ie-`', and will thus be found by a suitable search pattern. (One important exclusion is made to ignore files under the '`all-zips`' parent folder; the contents are redundant and mostly undated.) If using a path file, the same filtering technique is used to search through the path file for all ZIP files matching the given region code.

Matching ZIP filepaths are pushed to a list once found. If using a directory, this list is sorted alphabetically after searching completes. A path file is presumed to be already sorted, thus no further action is needed in that case.

The only minor procedural exception to the above steps will occur if either '`hr`' or '`hu`' are passed as the first argument. The regions represented by these codes, Croatia and Hungary, actually contain exclusively Croatian data, owing to an error made in the original TACT survey scripts[8]—hence ZIPs marked under either should be processed together. When either region is called, the program will therefore filter the given directory or path file for both Croatian and Hungarian ZIPs, and proceed using this combined list. (Because the final filtered ZIP paths are sorted by their parent directories, we assume the ZIPs are not processed achronologically, i.e. there is no risk of inaccurate start/end timestamps for each key instance.)

With the list of ZIPs finalised, the program iterates through each filepath on the list. A series of additional validity checks is made for each filepath. If the filepath points to an invalid file, it is skipped. If valid, the zip is opened and its contents are examined—if its structure does not match the standard TEK export format (two files, `export.sig` and `export.bin`) or is otherwise malformed, it is skipped.

From here, the `export.bin` file is opened and the program attempts to decode and process the Protobuf data within. The Protobuf specification itself is pre-compiled into a Rust class, with functions for creating structs from decoded data and accessing their

[8]See the 'Misclassified Data' section of the Appendix for more information.

contents. If the `export.bin` file fails to be decoded by this class, the file is skipped as malformed.

3.1.4 TEK Processing

If successful to this point, the `key_extract` program has decoded a given TEK export file into a struct with all of its data accessible within; now its list of TEKs can be processed.

The end goal decided upon for the final processed dataset was to track every unique instance of every unique TEK. Given these characteristics and the desire for optimal speed, a two-dimensional hash map[9] (`key_list`) was implemented for storing and updating processed TEK data within program memory. The key data of the TEKs, already being cryptographic random numbers, are used as the literal keys in the top-level hash map. Looking up a previously-stored TEK will therefore return its own hash map entry, enabling easy updating when that TEK is found again in the dataset.

(At this point caution may be raised about the possibility of a collision between TEKs. Some obvious test data, e.g. the key `00000000000000000000000000000000`, were detected from multiple regions at different times, but were not considered a compromise of the hash map's overall integrity. The probability p of at least one collision in n outputs from an ideal h-bit random number generator, approximated by the 'birthday problem' formula

$$p \approx \frac{n^2}{2(2^h)}$$

is calculated in our case[10] to be $p \approx 143521117^2/2(2^{128}) \approx 3.03 \times 10^{-23}$, i.e. negligible. Regardless, multiple disparate instances of each TEK are easily judged by comparing the key generation timestamps of each instance.)

The value tied to each key in the hash map is a `Key` struct containing the following:

- `first_appear`, `last_appear` (`int32`): Timestamps marking the earliest and latest detected appearances of the given TEK within the TACT dataset.

  (Thanks to the naming scheme of each hourly snapshot folder in the database, we can simply extract a timestamp from the ZIP's parent directory name itself; the program's `get_path_date()` function performs this. A newly-added TEK always initialises `first_appear` and `last_appear` as the same current folder time; upon finding the TEK again, `last_appear` will be re-assigned to the new current folder

[9] This program uses the `ahash` library for its hash map implementation instead of Rust's built-in `HashMap`—the former uses a faster non-cryptographic hashing function that was considered a worthy choice for processing TEKs with least delay.

[10] Using the number of unique TEKs in the TACT dataset from Section 2.2.

time. The prior chronological sorting of the program's ZIP path list ensures that `last_appear` will never be overwritten by an earlier timestamp.)

- `records`: Another hash map storing unique instances of the given TEK, in `KeyRecord` structs.

This second layer of hash maps is what stores the TEK metadata. Its keys are hashed using the `KeyRecord`s themselves, meaning any change in metadata will register as a new instance.

The `KeyRecord` struct's fields closely mirror the TEK ZIP's specified metadata fields as outlined in Section 2.1.2, with some exceptions and additions. It contains the following:

- `record_first_appear`, `record_last_appear` (`int32`): The earliest and latest ZIP file timestamps containing the given TEK instance.

  (Contrast with the `first_appear` and `last_appear` fields above, though the `record_last_appear` field is updated similarly to the latter. It was decided to include both ZIP timestamps and snapshot timestamps in the database, as only both together can provide full context for when the TEK was uploaded and when it was downloadable, respectively. The snapshot timestamps were chosen for the top-layer hash map as a broad overview of availability; the ZIP timestamps, being less chronologically flexible but allowing easier tracing of given TEK instances to their original ZIPs, were chosen for this bottom-layer hash map.)

- `start` (`int32`): UNIX timestamp of the TEK's generation.

  (Derived by multiplying the TEK's `rolling_start_interval_number` field by 600, the number of seconds in a 10-minute 'rolling period'.)

- `onset_of_symptoms` (`int32`): UNIX timestamp marking the user's onset of symptoms, as reported.

  (Derived by subtracting the TEK's `days_since_onset_of_symptoms` field—converted from days to seconds—from the `start` timestamp above. This conversion was decided upon for the sake of consistency and easy interoperability with other time-based fields; the original field's value can be easily derived in reverse as needed.)

- `risk_level` (`int32`) The TEK's `transmission_risk_level` field, unchanged.

- `rolling_period` (`int32`) The TEK's `rolling_period` field, unchanged.

(A mistake was made here in not converting the rolling period into seconds, which would fit better with the other UNIX timestamp fields. Doing this conversion on-the-fly is trivial, however.)

- `report_type`, `variant_of_concern` (`pointer size int`): Codes representing values of the `ReportType` and `VariantOfConcern` enums stored with the TEK.

  (These enums are hardcoded into the program as string arrays, for later CSV printing; hence these fields point to indices of either array. The pointer size int requirement is a related quirk of Rust's array accessing.)

This process is repeated for each TEK ZIP in the program's list, with data being added and updated to the same `key_list` hash map. By the end of the loop, this hash map contains each TEK posted by the given region, along with its date range and its various unique instances. This data must now be re-serialised into text form.

3.1.5 CSV Output

The decision to use the CSV format for `key_extract`'s output was driven primarily by its simplicity and compactness. The CSV structure defines its headers in the first row, with the remaining rows containing only raw comma-separated data; compared to JSON, where fields must be named every time, it is certainly more space-efficient.

However, the nature of the two-dimensional hash map defined above appears unsuitable for the inherently one-dimensional row format of CSV. The solution found was to store the `records` hash map as a two-dimensional list within a single column. This formatting is less readable in plain text, but successfully preserves structure for importing into another application, which remains the primary purpose of this CSV dataset. (The `KeyRecord` field names are still listed within the CSV header.)

Rust's built-in CSV library allows data to be written row-by-row to a given output (here the terminal's standard output) using the `Writer` class. The function `write_csv()` is defined to accept the given region code and `key_list` hash map as input, and print CSV rows by iterating through the hash map—most fields are printed as represented.

Here some formatting choices are made for the sake of PostgreSQL compatibility in the next phase of development—sacrificing further generic readability in order to save on time-consuming format conversion operations later as the database is populated. UNIX timestamps are converted into a string format using the `csv_timestamp()` function, in order to match PostgreSQL's default timestamp representation (based on ISO 8601). The `report_type` and `variant_of_concern` integer types are passed as indices to their respective hardcoded arrays, returning the original enum values as strings. All brackets,

commas and quote marks within the `records` column are escaped with a backslash—otherwise PostgreSQL will fail to process them correctly.

The 16-byte TEK itself is encoded as 32 hexadecimal characters, following the example set by the TACT project's scripting suite.

Through passing every region code covered by the TACT dataset into `key_extract`, we are left with a collection of 32 CSV files comprising a complete list of unique TEKs from every region in the dataset. These CSV files, though a good starting point, are not sufficient on their own to allow for rich querying of the dataset as specified in this project's research question. The next phase will fulfil this goal.

3.2 Database Design and Implementation

The final phase of development involves designing and building a database suitable for performing queries on the CSV data obtained through processing the TACT dataset with `key_extract`. Many database paradigms were considered for implementation, such as the document-based NoSQL platform MongoDB—the JSON-based database syntax of which would allow easy integration into web projects. Given the tabular structure of CSV, however, it was decided the simplest solution for a proof-of-concept would be a relational database management system. PostgreSQL was chosen for its advanced feature set and performance, as well as the author's previous experience working with it. Durumeric et al.'s use of PostgreSQL for their own cryptographic key database [18] (as mentioned in the 'Related Work' section) was a further contributing factor.

The process of initialising the database simply involved running the command `createdb tek` on a fresh PostgreSQL install. From there, SQL scripts were developed to create the `tek` database's tables, define each table's fields and special types, and populate their rows by importing the CSV files using the built-in `\copy` command.

To make database population and usage more manageable, separate tables are used for each of the 32 regions covered in the TACT dataset, matching the CSV files generated by the `key_extract` program; each table shares the same schema, preserving interoperability. (These tables were generated automatically using the `build-database.sh` file included with the dissertation materials.)

Because SQL cannot directly select data from multiple tables, a hardcoded list of all region table names is stored in a separate 1-column '`countries`' table, and is used in for-loops to iterate over the database (in part or in full). There is also another '`uniques`' table which simply stores a list of every unique 16-byte TEK in the database, again for easy iteration purposes.

The region table schema follows the structure and naming of the `Key` and `KeyRecord`

structs as formatted in the CSV files. Timestamp fields are stored as SQL's `timestamp` data type. The `KeyRecord` list is stored as a SQL 'composite type' array, preserving its two-dimensional form within a single SQL cell—its elements and attributes can be accessed by index as with any array. `report_type` and `variant_of_concern` are defined again here as enum types, with their string-based values easily matched to the CSV's values. The full schema can be found in the `build-1.sql` script included with the dissertation materials.

Chapter 4

Results

This project was conducted on a desktop PC running Manjaro Linux, with a Ryzen 7 5700X CPU, 32 GB of RAM (plus a further 32 GB of virtual swap space)[11], and a solid-state hard drive. All performance statistics have been recorded on this system.

4.1 New Datasets

When compressed into a `.tar.gz` file, the complete collection of CSV datasets measures only 30 gigabytes on disk—1.3% the size of the original TACT dataset (2.3 terabytes).

Despite this immense size reduction, much of the TACT dataset's pertinent content has been preserved—in the case of new UNIX-format timestamp fields, even enhanced. The primary aim of cutting all redundancy across the hourly snapshots has been achieved by storing only unique TEK records per country. This comes with some tradeoffs elsewhere, however.

The most glaring weakness of the new CSV datasets are their approximation of TEK appearances using date ranges. Simple timestamps marking first/last appearances fail to illuminate gaps in ZIP appearances recorded over time. An effort to fill this data gap was attempted by including ZIP appearance data ranges as well (where first/last appearances are usually equal) to help users pinpoint the exact ZIP containing the given TEK more easily. Obviously including a full list of ZIP appearances would be both unwieldy and inefficient, not to mention the additional memory it would take to track them. A tradeoff of some kind needed to be made, and timestamps kept the most interoperability with the other time-based TEK metadata fields.

[11] Without adequate swap space, 48GB or more of RAM is recommended for running `key_extract` on regions with large numbers of TEKs, such as Germany and Spain; otherwise the program's hash maps will grow too large to fit in RAM and the program will crash.

4.2 Database Performance

The PostgreSQL TEK database takes up 132 gigabytes on disk, approximately 6% the size of the original TACT dataset.

Measuring the speed of queries depends on the size of the tables being queried, which vary in size from 13 rows (`ec`) to 72 million rows (`es`); in the case of the average-sized `nl` table (21 million rows), retrieving a single TEK takes 1.85 seconds. Performing queries on every region table in the dataset is significantly more resource-intensive—the `track()` function described in section 4.3.2 takes on average 2.5 minutes to complete.[12]

4.3 Query Examples

Queries spanning multiple regions/tables must use a for-loop to iterate smaller queries over each region; results can be printed to console per-country, or added as rows to a new temporary table for later use. Accessing individual records in a row can be done using array access (`records[1]` returns the first record); alternatively the built-in `unnest()` function can be used to separate the array into distinct rows for each record.

4.3.1 TEK records and metadata

The script `types.sql` outputs a tally of report types for TEK records by region. The full tally is shown in figure 4.1. (A similar count was made for the 'variant of concern' field, which showed that all TEKs in the database had this field set to 'Unknown'.)

4.3.2 Tracking TEKs

The script `tektracker.sql` defines a SQL function, `track()`, which takes a TEK string as input and produces a chronological table of all instances of the TEK across the database—allowing the user to track where/when the TEK first appeared, and where it travelled over time. Table rows are sorted by `first_appear` and `record_first_appear`, in that order.

A sample of queries and results using the `track()` function are shown in the figures below. Examples were chosen from regions known to be sharing keys (EFGS members, Ireland/Northern Ireland, US states), across the timespan of the TACT survey; only the most relevant columns have been included for clarity.

[12]These times were captured within PostgreSQL using its `\timing` command.

Region code	Unknown	Confirmed test	Confirmed clinical diagnosis	Self report	Recursive	Revoked
at	32674121					
be	58834	14409409	13816105	7015317	25198051	
br	18702					
ca	175343	384258				
ch	944287					
cz		4933195	532029			
de	731840	23483875	20594361	10961626	4537974	
dk	565886	4586265				
ec	13					
ee	61162					
es	72007699					
fi	2042381	28165048				
gu		2370958	3289776			944893
hr	268	64806905	682623			
ie	1126295	41024706				
it	14690292					
lv	61935396					
mt	58433641					
nl	2120935	18897936				
pl	26841870	182	224			28
pt	40960					
si	63578	7504716	7671856	3727230	13065078	
ukenw	12654772					
ukgi	739848	8244210				
ukni	799277	8387584				
uksc	590086	7801376				
usal	209000	15004106	6236	5845376		71
usde	98167	15217059	6236	5898025		71
usnv		6253				
usva	55503	14334923	6233	5728627		70
uswy	98167	15217059	6236	5898025		71
za	205691	1340981				

Figure 4.1: A chart tallying regional TEK records grouped by report type, according to queries made to the TEK database using `types.sql`.

Figure 4.2: `SELECT * FROM track('ffffffd2246680656101ac86f0f3a8e2');`

Region	1st server appearance	1st ZIP appearance	Risk level	Report type
ie	2022-03-13 21:00:01	2022-03-13 19:30:54	8	Confirmed test
de	2022-03-13 21:00:01	2022-03-13 20:00:00	8	Confirmed test
nl	2022-03-13 21:00:01	2022-03-13 21:03:07	3	Confirmed test
hr	2022-03-14 00:00:01	2022-03-13 23:59:59	8	Confirmed test
fi	2022-03-14 03:00:01	2022-03-14 00:00:00	6	Confirmed test
es	2022-03-14 06:00:01	2022-03-11 02:00:00	8	Unknown
mt	2022-03-14 14:00:01	2022-03-11 02:00:00	8	Unknown
lv	2022-03-14 18:00:01	2022-03-14 17:00:09	1	Unknown
be	2022-03-15 02:00:01	2022-03-15 00:00:00	8	Confirmed test
pl	2022-03-16 02:00:01	2022-03-16 00:00:00	8	Unknown

(TEK generated 2022-03-11 00:00:00)

Figure 4.3: `SELECT * FROM track('78a4243894366fece682766fd642e38e');`

Region	1st server appearance	1st ZIP appearance	Risk level	Report type
usde	2023-04-03 15:00:01	2023-04-03 12:00:00	6	Self report
uswy	2023-04-03 15:00:01	2023-04-03 12:00:00	6	Self report
usal	2023-04-03 20:00:01	2023-04-03 12:00:00	6	Self report
usva	2023-04-04 03:00:01	2023-04-04 00:00:00	6	Self report

(TEK generated 2023-03-29 00:00:00)

Figure 4.4: `SELECT * FROM track('e6e1d4635d6d2b52702268d91dc90bcf');`

Region	1st server appearance	1st ZIP appearance	Risk level	Report type
ukni	2020-10-14 09:00:01	2020-10-14 07:57:28	0	Unknown
ie	2020-10-14 11:00:01	2020-10-14 10:00:26	0	Unknown

(TEK generated 2020-10-07 00:00:00)

4.4 Discussion

The sample figures derived from the TEK database already allow interesting observations to be made about the GAEN app ecosystem:

1. The use of report types was found to be highly inconsistent between regions. Figure 4.1 shows that nearly a third of all regions had their TEKs' `report_type` permanently set to 'Unknown', while 22 out of 32 regions only utilised two `report_type` values at most. Only Belgium, Germany, and Slovenia showed consistent usage of five report types; none used all six. The usage of the 'Revoked' type, along with the extra 'revoked keys' TEK list in the Protobuf spec, was considerably rare, with only Guam making substantive use of both. Another interesting usage pattern is of the 'Recursive' type, which is only specified by Google as a placeholder type "reserved for future use"; its frequent usage by three regions in the survey is unexplained as of yet.

2. The 'variant of concern' field was found to be completely unused (i.e. set to the default 'Unknown' value) across the database. The late addition of this field to the TEK export specification, with unclear definition of its usage by Google, may have contributed towards its lack of implementation.

3. Regarding the EFGS member countries, the successful sharing of TEKs is evident, with the case of figure 4.2 showing 10 countries receiving the given TEK over time—however, risk levels and report types also vary across regions, here jumping erratically between 'Confirmed test' and 'Unknown'. This behaviour may be explained by the varying configurations of each region's GAEN backend, as previously discussed via figure 4.1, but is ultimately detrimental to most efforts of preserving and trusting TEK metadata, or using it for evaluating COVID policy.

4. The time it takes for TEKs to be shared across countries is sometimes significant; in figure 4.2 it takes three days for the TEK to propagate from Ireland to Poland—by which time the TEK is five days of age, a sizable portion of its 14-day lifespan. With delays of this length (or longer) now shown to be possible, the usefulness of TEK-sharing networks such as the EFGS may be called into question.

From these findings, the project's goal of using a dedicated database to facilitate new insights on GAEN in practice—as outlined in the 'Research Question' section—has been achieved. The tools have been established and demonstrated to go further in analysing individual TEKs than previous research has gone, and on the greatest scale yet by far.

Chapter 5

Conclusion

In October 2020, the TACT project concluded in their TEK survey paper that "the community needs to determine if COVID-19 tracing apps contribute to handling the pandemic, are a distraction, or are detrimental." [8] Four years later, there is still much to determine—and much global incentive to do so.

With much of GAEN's infrastructure and apps now defunct, the research landscape is primed for retrospective evaluations of both GAEN's varying implementations and its aspirant effectiveness against COVID-19. Conducting these evaluations, however, requires data—which is hard to gather now, in the mostly dormant state of the GAEN ecosystem. This is what gives so much significance to the TACT dataset: it is a sizable snapshot of GAEN activity from dozens of regions, covering the majority of its operational lifespan; though certainly not a tell-all statistical source, it throws new and substantive light on the usage and implementation of GAEN on an international scale, focusing on the raw TEKs that are so often overlooked in previous literature.

This dissertation project aimed to produce the most useful possible version of the TACT dataset by reducing away its most cumbersome traits of redundancy and format obfuscation. While this reduction came with tradeoffs in detail, the result preserves the dataset's potential for tracking TEKs through time, location, and metadata. It did so using a two-phase methodology that produced usable material at both stages: a condensed plain-text CSV dataset, and a powerful database implementation for querying it. As part of this effort, a single comprehensive command-line tool was developed for directly processing the TACT dataset, which may be adapted for other types of processing and output.

While only preliminary, the findings detailed in this report still provide a remarkable demonstration of the database's capabilities, including eye-opening insights into how both the Diagnosis Servers of public health authorities and the TEK-sharing backend systems

touted by the EU and US played out in practice. It is evident that this new database—or a similar one generated with these new tools—would support exactly the kind of extensive and comparative evaluations that digital contact tracing researchers might no longer have the means to conduct otherwise.

5.1 Further Work

The final database is designed to support further work in research concerning GAEN and its implementations. Many possible avenues for research have been discussed in this paper—chiefly the potential tracking of TEKs across key-sharing networks such as the EFGS. Whether delays in TEK propagation rendered them less useful is perhaps the most pressing question related to these networks, and a complete analysis would be both feasible to perform using this database and highly illuminating.

However, the database itself could also be improved in efficiency. Currently, the `track()` function takes over 2 minutes to run on a high-end PC, owing mostly to its `SELECT` statements iterating over the entire database—with more than 140 million TEKs in the database, tracking every single one may be impossible in a reasonable time. Since the TEK data column serves as a primary key (when the `records` array is not unnested), we should only expect to find one instance of a TEK per table, and can therefore continue to the next table as soon as it is found; this operation will require more advanced SQL than is currently used.

There is also no guarantee that the TEK database's optimal form is its current CSV/SQL-based relational paradigm. The two-dimensional array of TEK records is seemingly at odds with SQL's one-dimensional row format, though PostgreSQL enables it to work well through using composite types. There may still be worth in exploring NoSQL implementations of the TACT dataset: as mentioned in section 3.2, MongoDB was considered for its JSON-based document format, which may be a cleaner way of representing TEKs (being already defined as object-like structs of data). In the end only one database management system was tested for this project due to time constraints, but it may be fruitful to contrast it with implementations in other systems.

Bibliography

[1] Kwadwo Agyapon-Ntra and Patrick E McSharry. A global analysis of the effectiveness of policy responses to COVID-19. *Scientific Reports*, 13(5629), 2023. doi: 10.1038/s41598-023-31709-2.

[2] World Health Organisation. Contact tracing in the context of COVID-19: interim guidance, 1 February 2021. Technical document, 2021. URL `https://iris.who.int/handle/10665/339128`.

[3] Apple Inc. Apple and Google partner on COVID-19 contact tracing technology, April 2020. URL `https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/`.

[4] Douglas J Leith and Stephen Farrell. Contact tracing app privacy: What data is shared by Europe's GAEN contact tracing apps, July 2020. URL `https://www.scss.tcd.ie/Doug.Leith/pubs/contact_tracing_app_traffic.pdf`.

[5] Lucie White and Philippe van Basshuysen. Privacy versus public health? A reassessment of centralised and decentralised digital contact tracing. *Science and Engineering Ethics*, 27(23), 2021. doi: 10.1007/s11948-021-00301-0.

[6] Douglas J Leith and Stephen Farrell. Testing Apps for COVID-19 Tracing (TACT). URL `https://down.dsg.cs.tcd.ie/tact/`.

[7] Douglas J Leith and Stephen Farrell. Measurement-based evaluation of Google/Apple Exposure Notification API for proximity detection in a light-rail tram. *PLoS ONE*, 15(9), September 2020. doi: 10.1371/journal.pone.0239943.

[8] Stephen Farrell. October 2020 survey of GAEN app key uploads, October 2020. URL `https://down.dsg.cs.tcd.ie/tact/survey10.pdf`.

[9] European Commission. Digital contact tracing study: Study on lessons learned, best practices and epidemiological impact of the common European approach on digital

contact tracing to combat and exit the COVID-19 pandemic, November 2022. URL https://op.europa.eu/s/zizU.

[10] Serge Vaudenay and Martin Vuagnoux. SwissCovid in the perspective of its goals. *Digital Threats*, 3(3), February 2022. doi: 10.1145/3480465.

[11] Apple Inc. and Google Inc. Exposure Notification Privacy-preserving Analytics (ENPA). White paper, April 2021. URL https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf.

[12] Apple Inc. and Google Inc. Exposure Notification cryptography specification. Technical document, April 2020. URL https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf.

[13] GitHub. Releases · google/exposure-notifications-server. URL https://github.com/google/exposure-notifications-server/releases.

[14] Stephen Farrell and Douglas J Leith. Transparency in the deployment of coronavirus contact tracing apps, June 2020. URL https://down.dsg.cs.tcd.ie/tact/transp.pdf.

[15] Stephen Farrell and Douglas J Leith. Irish Covidtracker app key upload shortfalls, April 2021. URL https://down.dsg.cs.tcd.ie/tact/ie-stats.pdf.

[16] Felix Holl, Johannes Schobel, and Walter J Swoboda. Mobile apps for COVID-19: A systemic review of reviews. *Healthcare*, 12(139), 2024. doi: 10.3390/healthcare12020139.

[17] Nikita Kesarev and Andrey Korochkin. Tracing the tracing apps: A technical response to COVID in cultural comparison. *Technology and Language*, 4:97–115, 2023. doi: 10.48417/technolang.2023.02.10.

[18] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 291–304, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319539. doi: 10.1145/2504730.2504755.

[19] Censys Inc. Research access to Censys data. URL https://support.censys.io/hc/en-us/articles/360038761891-Research-Access-to-Censys-Data.

Appendix
Summary of TACT Dataset Integrity

For a broader overview of the TACT dataset, please consult the relevant section of this paper and the original TACT web page [6]. This appendix references information originally logged on the TACT web page, but with further detail provided by queries to the TEK database.

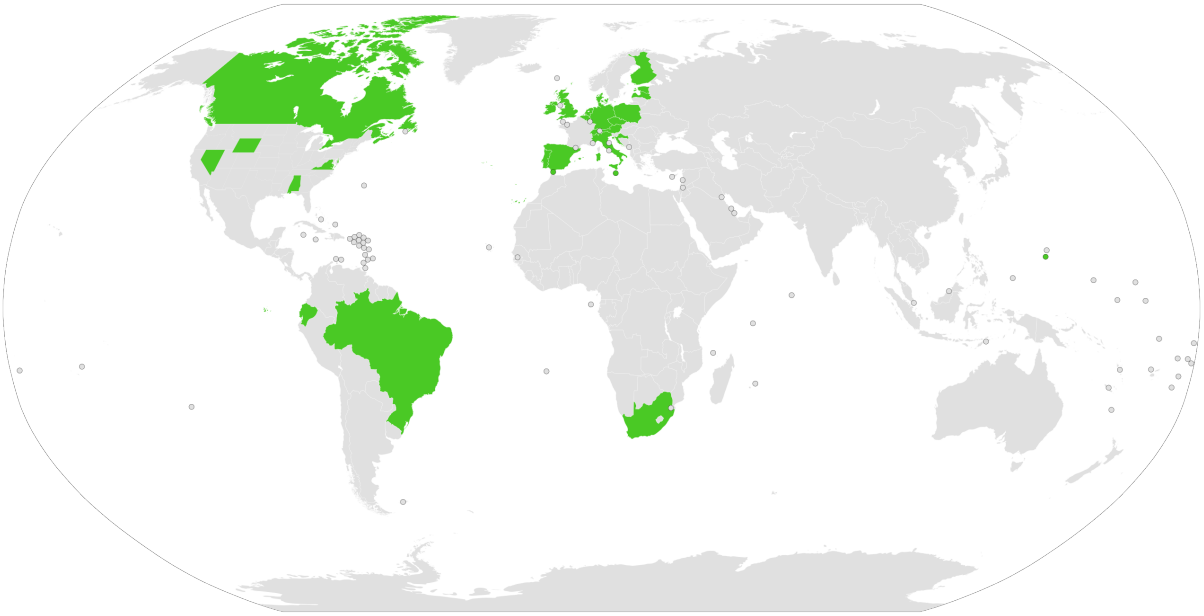

Figure 1: A world map diagram[13] with each region covered by the TACT dataset highlighted in green.

[13]Adapted from the Wikimedia Commons image 'Blank Map with US subdivisions.svg', created by user EmmaCoop and licensed under CC-BY-SA 4.0.

Data Coverage

The following table lists date ranges for the first and last detection of TEKs downloaded from each Diagnosis Server (as queried in the final TEK database). The actual monitoring period of each server may extend further in either direction, but with no TEKs detected in that time. This table also does not take gaps in data into account; information on the script outages and certificate expirations during the survey are chronicled on the TACT web page.

Region	Code	Data start	Data end
Austria	`at`	2020-07-09 01:23	2022-02-28 21:00
Belgium	`be`	2020-10-02 15:00	2022-12-03 15:00
Brazil	`br`	2020-10-02 15:00	2021-09-28 01:00
Canada	`ca`	2020-08-20 13:00	2022-06-17 17:00
Croatia	`hr`	2020-10-22 23:00	2023-02-21 21:00
Czechia	`cz`	2020-10-02 22:00	2021-11-22 08:00
Germany	`de`	2020-06-25 22:37	2023-05-26 09:00
Denmark	`dk`	2020-07-05 16:44	2022-04-01 09:00
Ecuador	`ec`	2020-10-02 15:00	2020-12-24 23:00
Estonia	`ee`	2020-09-10 19:00	2022-05-06 06:00
Finland	`fi`	2020-09-13 13:25	2022-06-15 00:00
Guam	`gu`	2020-10-03 23:00	2022-01-10 20:00
Ireland	`ie`	2020-07-10 19:15	2022-03-24 08:00
Italy	`it`	2022-06-25 22:37	2022-12-30 14:00
Latvia	`lv`	2022-07-09 02:38	2023-02-20 16:00
Malta	`mt`	2020-10-02 15:00	2022-07-11 11:00
Netherlands	`nl`	2020-10-03 00:00	2022-04-22 11:00
Poland	`pl`	2020-06-30 16:10	2022-03-31 08:00
Portugal	`pt`	2020-10-02 15:00	2022-06-10 06:00
Slovenia	`si`	2020-10-09 13:00	2022-04-04 08:00
South Africa	`za`	2020-10-02 22:00	2021-06-20 05:00
Spain	`es`	2020-07-13 23:00	2022-10-11 16:00
Switzerland	`ch`	2020-06-25 22:37	2022-03-31 22:00
UK: England & Wales	`ukenw`	2020-10-02 15:00	2023-05-12 15:00
UK: Gibraltar	`ukgi`	2020-10-08 18:00	2022-06-04 07:00
UK: Northern Ireland	`ukni`	2020-08-06 18:13	2022-09-21 07:00
UK: Scotland	`uksc`	2020-09-16 16:00	2022-04-29 13:00

US: Alabama	`usal`	2020-09-10 18:00	2023-05-12 15:00
US: Delaware	`usde`	2020-09-16 16:00	2023-05-12 15:00
US: Nevada	`usnv`	2020-09-16 22:25	2020-10-27 01:00
US: Virginia	`usva`	2020-08-14 15:10	2023-09-03 14:00
US: Wyoming	`uswy`	2020-09-16 23:00	2023-05-12 15:00

Misclassified Data

From October 2020, data was gathered from what were presumed to be Hungarian servers, and was classified as such ('`hu`'). Data from Croatia was later added in December 2020, but it appeared identical in content to Hungary. In reality, both sets of data were of Croatian origin. This was only discovered in April 2022, after which Hungary was dropped from TACT's survey and Croatian data gathering continued.

To account for both the overlap in data and the original misclassified Croatian data from Oct–Dec 2020, the `key_extract` tool includes a condition to extract ZIPs marked `hr` and `hu` simultaneously, and all such data is merged into the same CSV export. The final Croatia database table therefore contains all data from Croatia without gaps or conflicts.

Malformed Data

On several occasions, malformed UNIX timestamps were found attached to both TEKs and ZIP files—in each case set to zero (January 1, 1970).

Region	TEK timestamps	ZIP timestamps
`ca`	1	
`ie`	2	2360
`ukgi`	2	
`ukni`	2	2140

Figure 2: Table showing all regions with malformed UNIX timestamps.

There was also one instance of a malformed TEK found in the TACT dataset. The key first appeared in the Northern Ireland dataset on 2020-08-07 23:00 and appeared two hours later in the Ireland dataset. The key `0xd66cdcd462763af20dd34d34d34d34d34d34` exceeded the standard 16-byte length, and in each case the ZIP file's timestamp was set to zero. (It is assumed to have been half-corrupted during transmission.) These key instances were simply truncated to 16 bytes, then added to the database without further issues or conflicts.