Australian Government
Digital Transformation Agency

COVIDSafe

Cryptography Specification

dta

Table of contents

Introduction

About COVIDSafe

COVIDSafe is the Australian system for using mobile technology to supplement manual contact tracing during the COVID-19 pandemic. For general information about COVIDSafe, see the official COVIDSafe website: https://www.covidsafe.gov.au/

The COVIDSafe system consists of three major components:

1. The COVIDSafe applications for Android and iOS mobile devices (the “apps”), available from the Google Play Store and Apple App Store respectively.
2. A server API for managing registrations of, issuing identifiers to, and receiving uploaded exposure information from the mobile apps.
3. A web application (the ”portal”) used by authorised officials of Australian state and territory public health authorities to access uploaded exposure information.

The server-based components — the API and the portal — are hosted on an Amazon Web Services (AWS) tenancy, which is administered by an Australian Government agency authorised by the *Privacy Act 1988*.

The apps are supported on:

- Android 6+ (versions 1–2)
- Android 5.1+ (versions 3+)
- iOS 10+ (all versions).

Release history

Version numbers used in this document are the cryptography protocol version, numbered sequentially. Each protocol version aligns with a COVIDSafe release. Table 1.1 gives the protocol versions and corresponding Android and iOS app versions.

Crypto. ver.	Android ver.	iOS ver.	Release date
1	1.0.11	1.0	26 April 2020
2	1.0.18	1.4	26 May 2020
3	1.0.21	1.5	4 June 2020
4	1.0.33	1.7	2 July 2020
5	1.0.39	1.8	20 July 2020
6	1.0.48	1.9	2 August 2020
7	1.0.56	1.10	13 August 2020

Table 1.1: COVIDSafe cryptography version history

About this document

Scope

This document gives the high-level specification for cryptography in the Australian Government COVIDSafe system. It is concerned with how cryptography secures communication between, and data stored by, the major components of the COVIDSafe system.

This document only describes the internal cryptography of the COVIDSafe system. It does not detail the security of the underlying platforms or protocols, such as AWS, Android, iOS, or Bluetooth Low Energy (BLE), even though COVIDSafe may rely on security protections of those platforms and protocols. Refer to the official vendor documentation for more information on those systems.

Cryptographic security is often used in conjunction with, or is complemented by, non-cryptographic security. This document does not describe non-cryptographic security in the COVIDSafe system, unless a cryptographic mechanism directly depends on a non-cryptographic mechanism.

This is not a policy document. Information given on the security provided by cryptography in the COVIDSafe system is intended to assist the reader in assessing the purpose and degree of security provided. It is not intended as a statement or justification of Australian Government policy on contact tracing, privacy, or any other issue.

This document should be read in conjunction with the Privacy Impact Assessment (PIA) and the Australian Government Department of Health's response to the PIA.

Intended audience

This document assumes the reader has all of the following:

- familiarity with COVIDSafe, especially its purpose and high-level design
- a working understanding of fundamental cyber security concepts
- familiarity with common cryptographic algorithms
- an understanding of how cryptographic primitives are used to construct a cryptographic protocol.

Glossary of terms and abbreviations

Named actors

This document uses the following terms to describe the various actors in the COVIDSafe system:

user A person running COVIDSafe on one or more mobile devices.

health worker An officer of an Australian state or territory health authority, authorised to access the COVIDSafe portal.

data store administrator An officer of an agency defined as an administrator by the *Privacy Act 1988*, authorised to modify registration information with the consent of users.

app An installation of one of the COVIDSafe apps running on a mobile device.

National Data Store A COVIDSafe API endpoint, including any associated data processing and storage.

portal The COVIDSafe portal application used by health workers to access encounter data stored in the National Data Store.

server Either or both the National Data Store and the portal.

browser Used by a health worker to access the COVIDSafe portal.

Standards organisations

This document refers to standards by the following organisations:

IETF The Internet Engineering Task Force, an international standards community.

ISO The International Organization for Standardization, an international standards organisation.

NIST The National Institute of Standards and Technology, an agency of the United States Department of Commerce.

SECG The Standards for Efficient Cryptography Group, an international industry consortium.

Cryptographic primitives and other functions

Throughout this document, the following terms have the meanings given here:

AES The advanced encryption standard symmetric cipher, as specified by NIST in FIPS PUB 197, keyed by a 128-bit key generated by a CSRBG unless otherwise specified.

AES-CBC AES used in CBC mode, with PKCS #7 padding unless otherwise specified.

CBC The cipher block chaining block cipher mode, as specified by NIST in SP 800-38A, initialised with a 128-bit initialisation vector generated by a CSRGB unless otherwise specified.

CSRBG A cryptographically-secure random bit generator, specifically a RBG provided and defined as cryptographically secure by the platform, unless otherwise stated.

datetimestr The ISO 8601 string representation of a number of seconds past the Unix epoch, with both date and time components.

decstr The big-endian decimal string representation of a number.

ECDH The elliptic curve Diffie-Hellman key agreement scheme, as specified by SECG in SEC 1, using curve P-256 as specified by NIST in FIPS 186-4 unless otherwise specified.

ECIES The elliptic curve integrated encryption scheme, as specified by SECG in SEC 1.

ECkeypair The elliptic curve key pair generator, as specified by SECG in SEC 1, using curve P-256 as specified by NIST in FIPS 186-4 unless otherwise specified.

GCM The Galois/counter mode block cipher mode, as specified by NIST in SP 800-38D.

hexstr The big-endian hexadecimal string representation of a number. If the number is a UUID, it *may* include hyphens per IETF RFC 4122.

HMAC The keyed-hash message authentication code algorithm, as specified by NIST in FIPS 198-1, using SHA-256 unless otherwise specified.

IV An initialisation vector used with a block cipher mode, as specified in the relevant block cipher mode specification, generated by a CSRBG unless otherwise specified.

lpad Repeatedly prepend a character to a string until the string is at least a given length.

LSB The least-significant bit(s) of a bit-string.

MSB The most-significant bit(s) of a bit-string.

now The current system time, in seconds past the Unix epoch.

retkey Retrieve a named key or keypair from the platform’s key management system. Note that the names are only meaningful in this document, to distinguish between unique keys/keypairs in the COVIDSafe system. For example, α is a different key to β.

RSA The Rivest-Shamir-Adleman asymmetric cipher or digital signature, as specified by IETF in RFC 8017.

SHA The SHA-256 cryptographic hash function from the secure hash algorithm family, as specified in FIPS 180-4.

Notation

Algorithms in this document use the following notation:

- $x \leftarrow y$ assigns the value of y to x.
- $\|$ concatenates two strings.
- $\vdash$ partitions a bit-string into two or more substrings.
- String literals are typeset in monospace font and surrounded by quotes.

Trademarks

COVIDSafe is a trademark of the Crown in right of the Commonwealth of Australia, managed by the Australian Government Department of Health, registered in Australia.

Amazon Web Services is a trademark of Amazon Technologies Inc., registered in Australia and other countries.

Apple, App Store, Apple Store and CryptoKit are trademarks of Apple Inc., registered in Australia and other countries.

Google, Google Play, Play Store and Android are trademarks of Google LLC, registered in Australia and other countries.

iOS is a trademark of Cisco Technology Inc, registered in the U.S. and other countries, and used under licence by Apple Inc.

Security considerations

The COVIDSafe system aims to achieve as many of the following security properties as possible, without negatively impacting the functional requirements of the system or the end-user experience, and within the limitations of mobile platforms:

- **Anonymity**
    - *of users:* It should not be possible to identify any real person given the data exchanged by the app during an encounter or stored by the app following an encounter. except 1,2
- **Confidentiality**
    - *of user information:* It should not be possible to recover any personal information given by a user during registration or generated by the app during an encounter. except 1
    - *of user behaviour:* It should not be possible to infer any information about a user's behaviour as a result of installing and running the COVIDSafe app. except 1
- **Authentication**
    - *of users:* It should not be possible to impersonate any other user, either when communicating with the National Data Store or during an encounter. except 3
    - *of health workers:* It should not be possible to impersonate any other health worker when accessing the portal, or to do so anonymously.
    - *of the server:* It should not be possible for an app to communicate with any server other than the National Data Store.
- **Authorisation**
    - *of users:* It should not be possible for a user to exchange valid information in an encounter without registering.
    - *of health workers:* It should not be possible for a state or territory health worker to access encounter information from another jurisdiction without approval.
    - *by users and health workers:* It should not be possible to upload encounter information without the consent of *both* the user and a health worker.
- **Integrity:** It should not be possible to undetectably modify:
    - registration data after it has been provided except 4
    - information generated as part of an encounter
    - encounter histories uploaded to the National Data Store.

Cryptography is used to enforce many of these properties, as described in this document.

Exceptions

1. Unless you are a health worker viewing encounter information voluntarily uploaded by a user.
2. Unless you directly observe the encounter.
3. Unless you have access to the device.
4. Unless you are a data store administrator and have the user's consent to modify their registration information.

General

Key generation and management

Most long-term cryptographic keys used in COVIDSafe are generated and secured on the server.

Unless otherwise specified, AES keys are generated directly from a CSRBG, and long-term RSA and ECDH keypairs are generated using OpenSSL. Generation of ephemeral keys by the apps and browsers is platform dependent.

All long-term keys are stored in the relevant platform's key management system:

AWS	https://aws.amazon.com/kms/
Android	https://developer.android.com/training/articles/keystore
iOS	https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_keychain

Communication with the server

All communications with the server are secured under the Transport Security Layer (TLS) protocol, as configured to meet the requirements of the Australian Government Information Security Manual (ISM).

TLS implementations used in the COVIDSafe system are limited to:

- Apple's iOS platform implementation, used by the COVIDSafe app for iOS
- Google's Android platform implementation, used by the COVIDSafe app for Android
- browser or operating system implementations used to access the COVIDSafe portal
- the AWS Labs s2n implementation, used by all server components.

Server-side rules prevent non-compliant configurations of SSL/TLS being used to access the server.

Certificate pinning

Australian law prohibits COVIDSafe data and communications from being used or accessed for any purpose other than contact tracing.

Some networks implement TLS offloading or TLS inspection. TLS offloading occurs when traffic is secured between the server and a proxy on the network, and traffic between the proxy and the device is not secured. TLS inspection occurs when traffic is secured between the server and a gateway, using the server's certificate, and also between the gateway and the device, using the gateway's certificate. These network configurations are common in corporate networks, but are not allowed for COVIDSafe traffic.

To prevent TLS offloading/inspection and to mitigate malicious person-in-the-middle attacks, apps will only communicate with servers that can produce a certificate that is both:

- issued to the COVIDSafe API domain (`**.aws.covidsafe.gov.au`)
- issued by one of the Amazon Root Certificate Authorities (1–4) or the Starfield Services Root Certificate Authority (G2).

Networks with TLS offloading/inspection must create an exception for COVIDSafe traffic if they connect mobile devices running the app.

Security properties

Use of properly-configured TLS provides the following properties:

- confidentiality and integrity of data in transit between
    - an app and the National Data Store
    - a browser and the portal
- authentication of
    - the National Data Store to apps
    - the portal to health workers.

TLS is a standard widely-used protocol providing strong security. COVIDSafe uses TLS as the primary mechanism to enforce the above security properties.

Health workers are still susceptible to attacks that cause them to connect to different servers (e.g. phishing or DNS poisoning). Such sites may still be secured by TLS and may appear legitimate to a typical end user, but are controlled by a malicious actor instead of the Australian Government. Health authorities should refer to guidance issued by the Australian Cyber Security Centre on how to mitigate these attacks.

Registration

Overview

Registration is a three-step process:

1. **Identification:** the user provides registration information which is stored in the National Data Store.
2. **Authentication:** the user proves they have access to the mobile phone number they provided.
3. **Authorisation:** the National Data Store issues the app with long-term credentials.

Identification

Detailed procedure:

1. The user downloads and runs the app.
2. The user completes the registration form within the app.
3. The app sends the user's registration information to the National Data Store.[1]
4. The National Data Store creates a new registration record, or matches the phone number to an existing registration record.

When the server receives a registration request, it creates a unique registration record. Each record has a unique long-term identifier — the "registration identifier". This is a UUID version 4, whose random bits are generated by a CSRBG.

Authentication

Detailed procedure:

1. The National Data Store generates a session identifier and an associated random six-digit one-time PIN.
2. The National Data Store sends the session identifier to the app.
3. The National Data Store sends the PIN to the given mobile phone number via SMS.
4. The user enters the PIN into the app.
5. The app sends the session identifier and the user-entered PIN to the National Data Store.
6. The National Data Store matches the session identifier and the PIN to a registration record to authenticate the user.

[1] The app does not persist any registration information entered by the user.

Session identifier

Session identifiers serve as an authentication credential, allowing an app to prove in later registration steps that it initiated a specific registration.

The session identifier is the mobile phone number encrypted by AES-CBC, using a long-term key and random IV. The same key is used for all session identifiers, and is only used for this purpose. Random IVs ensure each session identifier is unique to a specific registration, even if the same phone number is used to register multiple devices or re-register the same device. IVs are stored as temporary data associated with the registration record and are not transmitted with the session identifier.

Session identifiers are only used during the authentication stage of registration. They are not exchanged during encounters. They are only stored by the app for as long as necessary to complete registration.

One-time PIN

One-time PINs are generated by a CSRBG using rejection sampling, as follows:

```
procedure GenerateOTP
    repeat
        $n \leftarrow$ CSRBG$(24)$
    until $n < \lfloor 2^{24} \div 10^6 \rfloor \times 10^6$          ▷ Remove bias from base-2 → base-10 conversion
    $p \leftarrow$ DECSTR$(n)$
    return LPAD$(p, char = 0, width = 6)$
end procedure
```

One-time PINs are only valid for a fixed period of time. The expiry is not part of the PIN construction; it is stored as a temporary value associated with the registration record, and is checked during the final step of authentication. If the current system time is after the expiry time, the PIN is rejected and the user must request a new one.

If the server receives a PIN that is not valid for the given session identifier, it will reject the request and the app will prompt the user to enter the PIN again. If the server receives three invalid requests, it will permanently reject the registration. The user will need to start the registration process again.

Security properties

Use of a one-time PIN provides the following properties:

- authentication of users to the National Data Store.

Authentication of users is limited to proving the user has access to the mobile phone number they provided when registering. By entering a valid one-time PIN, the user demonstrates they are able to access SMS messages sent to that number. The Australian Telecommunication Network is responsible for enforcing confidentiality and integrity of the SMS message containing the one-time PIN.

If a user attempts to register with a phone number controlled by someone else, that person will be alerted to the registration attempt on receiving the (unsolicited) SMS containing the one-time PIN.

COVIDSafe does not verify that any information provided at registration is factually correct. Verification of that information is performed by human contact tracers, as necessary, in the event a user voluntarily uploads their encounter history.

Authorisation

Detailed procedure:

1. The National Data Store marks the registration as valid.
2. The National Data Store sends a unique long-term authentication credential to the app.
3. The National Data Store removes temporary information from the database, including the session identifier and PIN.
4. The app encrypts the credential and stores it on the mobile device.

Authentication tokens

Authentication credentials are implemented as a JSON Web Signature (JWS) with a JSON Web Token (JWT) payload.

The JWT contains:

- the registration identifier
- JWS-enabling metadata.

The JWS digital signature algorithm is RSA with a long-term 2048-bit signing key. The same key is used for all JWSs, and is only used for this purpose.

JWSs are valid for 12 months. They are only issued on registration, and cannot be reissued or replaced. On expiry of the JWS, the app will no longer be able to communicate with the server. If a JWS is compromised, a National Data Store administrator can invalidate the registration record to revoke the JWS.

Storage of the JWS on the mobile device varies between platforms:

- **Android 5.1:** The JWS is encrypted by AES-CBC with a dedicated key and stored in shared preferences. The AES key is encrypted by RSA with a dedicated key, and also stored in shared preferences. The RSA key is stored in the keystore.[2]
- **Android 6+:** The JWS is stored in encrypted shared preferences (backed by the keystore).
- **iOS 10+:** The JWS is stored in the iOS keychain.

[2] Android 5.1 does not support encrypted shared preferences. The Android 5.1 keystore does not support AES keys.

Security properties

Use of JWS credentials provides the following properties:

- authentication of apps to the National Data Store
- authorisation of apps to request temporary identifiers from the National Data Store
- authorisation of apps to upload encounter information to the National Data Store.

COVIDSafe leverages the platform-level guarantee provided by keychain/keystore to ensure the JWS is only retrievable by the app.

When requesting a temporary identifier or uploading encounter information, the app includes its JWS in the request to the National Data Store. The server validates the digital signature to confirm it issued the JWS, and confirms the registration identifier matches a valid and active registration. If not, the request is rejected. If an invalid JWS or no JWS is included in the request, it is rejected.

Authorisation to upload encounter information requires additional information; see Uploading encounters.

Temporary identifiers

Overview

Temporary identifiers are a way of partially assuring anonymity of individuals involved in encounters.

The temporary identifiers used in COVIDSafe:

- are unique
- appear to be random
- can be correlated to a user by the National Data Store
- are verifiable (i.e. can't be changed without detection)
- can only be validly shared in an encounter for a limited period after being issued.

Temporary identifiers are valid for two hours by default. Apps implementing protocol versions 6+ will retrieve a temporary identifier that is valid for seven days, by adding a query parameter to the endpoint request.[3]

Apps may not always be able to retrieve a new temporary identifier on expiry of the existing one, for example due to lack of network connectivity. To mitigate this risk, apps will request a new temporary identifier from the National Data Store during the first encounter that occurs after a temporary identifier reaches a given age. This age is one hour by default, or one day for apps implementing protocol versions 6+.

See Uploading encounters for more information on how temporary identifiers are de-anonymised when encounter information is uploaded to the National Data Store.

Construction

Each time an app requests a new temporary identifier, the server generates it as follows:

```
procedure GetTempID
    $p \leftarrow$ HEXSTR($uuid$)                ▷ where $uuid$ is the unique registration identifier
    $e \leftarrow$ DATETIMESTR(NOW $+ \Delta$)
    $id \leftarrow p \parallel$ "/" $\parallel e$
    $K \leftarrow$ RETKEY($\alpha$)
    $iv \leftarrow$ CSRBG($128$)
    $c \leftarrow$ AES-CBC($K, iv, id$)
    $m \leftarrow$ HMAC($K, iv \parallel c$)
    return $iv \parallel c \parallel$ MSB($m, 128$)
end procedure
```

[3] Temporary identifiers are no longer the primary mechanism for mitigating replay attacks in versions 5+. This allows for the longer temporary identifier validity period in versions 6+.

Version 1

Prior to version 2, temporary identifiers were generated using AES-GCM:

```
$...$
$iv \leftarrow$ CSRBG$(96)$
$c, a \leftarrow$ AES-GCM$(K, iv, id)$        ▷ where $a$ is the GCM authentication tag
return $iv \parallel c \parallel a$
```

AES-GCM provides built-in integrity protection, but that protection is not guaranteed if the same IV is ever observed to be used twice with the same key. To ensure this did not occur in COVIDSafe, the API endpoint for requesting temporary identifiers was limited to 2^{32} invocations, in line with the NIST SP 800-38D recommend limit on random IVs with GCM.

Versions 2+

GCM was replaced by CBC+HMAC in version 2, to provide the same level of protection without the IV constraint. This affects the server only; all versions of the app now get the new construction.

Security properties

Use of temporary identifiers provides the following properties:

- anonymity of users
- confidentiality of personal information
- authentication of users during an encounter
- authorisation of users to participate in an encounter.

Anonymity

Temporary identifiers only have a role in enforcing anonymity in version 1. In versions 2+, the temporary identifier is under an additional layer of encryption (see Encounters, Versions 2+), where they continue to have a role in authentication protection.

As temporary identifiers appear to be random values, it is not possible to arbitrarily attribute them to specific users without the key. However, if a temporary identifier *is* attributed to a user, such as by direct observation of an encounter[4], then anonymity of that specific identifier is lost. This is partially mitigated by:

- rotating the identifier every 1–2 hours
- the limited range of BLE
- the complexity of attribution when there are multiple apps broadcasting.

[4] If you are able to directly observe an encounter, anonymity is not cryptographically enforceable.

Confidentiality

Temporary identifiers provide confidentiality by ensuring no uniquely identifying information is exchanged by the app during an encounter, other than the (encrypted) registration identifier.

In versions 2+, proximity data and device models receive additional protection. Temporary identifiers have no role in securing that information; see Encounters, Versions 2+ for more information.

Authentication

The server uses the registration identifier from the JWT to derive a temporary identifier on request, so the authenticity of temporary identifiers is derived from the authenticity of JWS.

However, because temporary identifiers are designed to be re-used, they are susceptible to replay attacks (including relay attacks), where an unauthenticated actor captures BLE messages and resends them to different devices. This allows the unauthenticated actor to masquerade as the user(s) of the captured traffic. Replay attacks are only successful while the temporary identifier remains valid, since the server will ignore any encounters that occurred after the temporary identifier expired.

Encounter records can also be deliberately duplicated in the local database by the user — for example, to convert a true casual contact to a false close contact by modifying the timestamps of the duplicated records.[5] As with replayed encounters, the server will ignore any duplicated encounters where the timestamp is later than the temporary identifier expiry. Duplicating records will have no impact unless the user subsequently tests positive for COVID-19 and is authorised by a health worker to upload their encounter history; see Uploading encounters for more information.

In versions 5+, there are additional protections that limit replay attacks to a few minutes. Temporary identifiers continue to provide backup protection against replay attacks; see Encounters, Versions 5+ for more information.

Authorisation

The message authentication code (or the GCM authentication tag in version 1) ensures a malicious user cannot create or modify a temporary identifier that will by accepted the server. With the exception of replay attacks, this means only registered users can sensibly participate in an exchange, since only registered users are able to retrieve temporary identifiers. Unauthorised users can still participate in exchanges by sending junk temporary identifiers, but these encounters will be ignored when uploaded to the National Data Store; see Uploading encounters for more information.

[5] The criteria for ”casual contact” and ”close contact” are defined by the relevant state or territory health authority.

Encounters

Overview

An encounter occurs when two mobile devices running COVIDSafe come within Bluetooth Low Energy (BLE) range of each other. The app will periodically scan for BLE advertisements from other devices running COVIDSafe. For each device found, the “local” app will send the following information to the “remote” app:

- the user’s temporary identifier
- BLE technical data used for determining physical proximity
- the device model
- the protocol version number
- a constant string identifying the protocol.

The remote app will add this information to a database of encounter records stored on its device.

The local app will also request the remote app return the same information. It stores the returned information in its own database of encounter records.

Version 1

Encounter information is not encrypted in version 1 (noting temporary identifiers provide their own confidentiality and integrity protection).

Versions 2+

In versions 2+, encounter information is encrypted asymmetrically. Only the National Data Store can validate and decrypt encounter information generated or received by apps implementing protocol versions 2+.[6]

Encryption is implemented using an ECIES-like scheme. ECIES was selected for the following reasons:

- it is highly configurable
- it can be configured to be compliant with the Australian Government Information Security Manual (ISM)
- elliptic curve cryptography is widely implemented and highly optimised on mobile devices

[6] When versions 2+ receive encounter information from version 1, the received information will be asymmetrically encrypted before being stored, even though it was sent over BLE in cleartext. Version 1 can receive and store encrypted encounter information from versions 2+; i.e. versions 2+ are backward compatible.

- certain well-known curves have been supported since early versions of Android and iOS
- it adds minimal overhead to the encounter payload size compared with many other asymmetric encryption schemes.

Payload size is a significant concern for encounter data, since BLE messages have limited bandwidth, and to ensure encounter databases do not exceed the storage space available to the app on a device.

Standard ECIES implementation requires a key-agreement for every encounter. Due to platform limitations, it is not feasible to do this. To work around this limitation, keys are cached and reused for up to 7.5 minutes or 2^{16} encryption cycles (whichever occurs first). Semantic security is provided by a counter-based nonce used as the IV, instead of key rotation as in standard ECIES.

Encounter encryption is implemented as follows:

```
$PK \leftarrow$ RETKEY(public $\gamma$)        ▷ Long-term ECDH public key for the National Data Store
$t \leftarrow -\infty$
$ctr \leftarrow 0$
$\Delta \leftarrow 7.5 \times 60$
procedure ENCRYPT($p$)
    if $t + \Delta \leq$ NOW $\vee$ $ctr \geq 2^{16}$ then
        $priv, R \leftarrow$ ECKEYPAIR
        $S \leftarrow$ ECDH($priv, PK$)
        $K_{AES}, K_{MAC} \leftarrow$ MSB(128), LSB(128) $\vdash$ SHA($S$)
        $t \leftarrow$ NOW                                           thread-
        $ctr \leftarrow 0$                                           synchronised
    else
        $ctr \leftarrow ctr + 1$
    end if
    $k_{AES}, k_{MAC}, r, n \leftarrow K_{AES}, K_{MAC}, R, ctr$
    $iv \leftarrow$ AES($k_{AES}, n$)
    $c \leftarrow$ AES-CBC($k_{AES}, iv, p$)
    $m \leftarrow$ HMAC($k_{MAC}, r \parallel n \parallel c$)
    return $r \parallel n \parallel c \parallel$ MSB($m$, 128)
end procedure
```

Implementation note: The apps implement a slightly optimised version with fewer key expansions, given:

$$\text{AES-CBC}(k, iv = \text{AES}(k, n), pt = p) \equiv \text{AES-CBC}(k, iv = \varnothing, pt = \text{PKCS\#7}(n) \parallel p)$$

Security properties

Use of ECIES-like encryption of encounter information provides the following properties:

- anonymity of users
- confidentiality of information generated during an encounter

- integrity of information generated during an encounter.

Temporary identifiers, BLE technical data and mobile device models can no longer be recovered from COVIDSafe traffic or encounter histories, except by the server. This also limits other forms of masquerade attacks except replay attacks, since malicious users can't collect temporary identifiers directly.

BLE technical information and mobile device models can no longer be manipulated prior to replay or in an encounter history, for example to make a casual encounter appear to be a close encounter or vice versa. The server will ignore any uploaded encounter whose integrity check fails.

Temporary identifiers can no longer be used to track users, since they are no longer sent or stored as plaintext. However, the presence of the ephemeral public key in the BLE message has the same impact on anonymity: attribution of the key to a specific user breaks the anonymity guarantee. By ensuring keys are not used for more than 7.5 minutes, this risk is now limited to a similar level of risk inherent in the BLE MAC randomisation scheme.[7]

The IV nonce provides semantic security. It is sent in plaintext during the encounter so that the IV can be reconstructed by the server. However, this may allow limited information on the behaviour of the user to be inferred, given it confirms the number of encounters in which the user's mobile device has participated.

Versions 5+

In versions 5+, a UTC timestamp is included in the encounter information under encryption. This timestamp is used to significantly limit replay attacks.

Security properties

Inclusion of an encounter timestamp under encryption provides the following property:

- authentication of users during an encounter.

Each encounter record now has two timestamps: one generated by the sender (under encryption), and one generated by the receiver. Since the timestamp under encryption can't be modified without detection, a mismatch between these two timestamps must indicate one of the following:

- the receiver's timestamp has been altered
- the encounter has been replayed at a later time
- there is a mismatch between the system clocks of the two devices.

Due to the widespread use of network time for synchronising system clocks of mobile devices, it is assumed to be unlikely that the clocks of two devices will differ by more

[7] Since platform limitations prevent the app from synchronising key rotation with MAC randomisation, the ephemeral public key and BLE MAC in combination allow an attacker to link a new public key or MAC to the same device. However, this is only possible while the device is under direct observation.

than a few minutes. The server therefore assumes that a larger mismatch must be the result of corruption or malicious action (see Uploading encounters for more information). This reduces the window for replay attacks from 2 hours ±5 minutes to only a few minutes.

Version 7

In version 7, the AES IV is generated from a 16-bit random nonce and keys are regenerated after no more than 2^8 encryption cycles:

```
$\ldots$
**if** $t + \Delta \leq \text{NOW} \vee ctr \geq 2^8$ **then**
    $\ldots$
**end if**
$k_{AES}, k_{MAC}, r, n \leftarrow K_{AES}, K_{MAC}, R, \text{CSRBG}(16)$
$\ldots$
```

Security properties

Use of a random nonce based IV provides the following property:

- anonymity of user behaviour.

In versions 2–6, the IV could be used to infer limited information about user behaviour. In version 7, this is no longer possible. However, the small nonce size (chosen to maintain backward compatibility) means there is a reasonable chance of an IV collision under high load, even with the reduced threshold for key rotation. This collision risk will negligibly reduce confidentiality due to the very low number of total collisions expected per key, the limited amount of ciphertext per key, and the block cipher mode of encryption.

Limitation on encounter encryption

Full ECIES encryption of each encounter would assure anonymity, since every encounter message would be unique. An asymmetric device-to-device handshake could also be added to mitigate replay attacks, except two-way relay attacks which cannot be mitigated by cryptography.

However, asymmetric cryptography is computationally expensive by design. Testing conducted by the Australian Government indicates full ECIES or device-to-device handshakes would significantly reduce battery life, impact performance, and push the limits placed on background services by Android and iOS, potentially resulting in encounters being dropped or users choosing not to use COVIDSafe. These results are consistent with the findings of the Singapore Government Technology Agency when developing BlueTrace.

Uploading encounters

Dual authorisation

Encounter histories should only be uploaded with the consent of *both* the user and a health worker. This is achieved using time-limited PINs.

When a user tests positive for COVID-19, a health worker contacts them and requests they upload their encounter history to the National Data Store. If the user consents, they provide their mobile phone number to the health worker. The health worker enters this into the portal to generate a time-limited PIN, which they provide to the user.

The user enters this PIN into the app, and the app sends it to the server for authentication. If successful, the server authorises the app to upload its local database of encounter records to the National Data Store.

The PIN is valid for all apps registered using the given mobile phone number.

Security properties

Use of a time-limited PIN issued to a health worker and entered by the user provides the following property:

- authorisation by users and health workers to upload an encounter history to the National Data Store.

Authorisation by the user ensures encounter histories can only be uploaded with the user's informed consent. Authorisation by a health worker ensures a malicious user cannot upload an encounter history if they have not tested positive for COVID-19, which would result in individuals self-isolating unnecessarily, and wasted effort by contact tracers and health authorities. Releasing the PIN to the health worker instead of directly to the user, and requiring the user to enter the PIN into the app, provides assurance that both a health worker and the user consent to the upload. See Health authority access for more information on how health workers access the portal to generate PINs.

The construction, server handling and security properties of the upload PIN are similar to registration PINs, but with the following differences:

- the PIN is checked against the JWS as well as the registration
- the PIN can only be used once per JWS
- the PIN can be used multiple times with different JWSs, so long as the associated registration uses the same phone number. Note that authorising a device with a PIN will prevent the JWS used by that device from using the PIN again — all other associated JWSs can still be used with the PIN for authorisation.

Processing and storage

When an app uploads its local database, the server performs the following actions:

1. An upload record is created and linked to the registration record for the app.
2. The encounter record is checked for uniqueness (versions 5+).
3. The integrity of each encounter record is checked.
4. The record is decrypted.
5. If the remote device was running an app with versions 5+ cryptography, the timestamp of the receiver is checked against the timestamp of the sender.
6. The integrity of the temporary identifier is checked.
7. The temporary identifier is deanonymised.
8. The timestamp of the receiver is checked against the temporary identifier expiry.
9. The (decrypted) record is linked to the upload record.

If any of the above checks fail for a given encounter record, no further processing of that record will occur. The encrypted encounter record is retained on the National Data Store, but is not linked to the upload record and is therefore not accessible through the portal.

A database uploaded by an app with version 1 cryptography may contain a mix of unencrypted and encrypted encounters. Steps 2 – 5 are skipped for any unencrypted encounters.

Integrity check and decryption

Encounter records may consist of one or two encryption blobs, depending on the combination of versions of the sender and receiver. Each blob is decrypted as follows:

```
procedure DECRYPT($msg$)
    $R, n, c, h \leftarrow \quad \vdash msg$
    $K \leftarrow$ RETKEY(private $\gamma$)        ▷ Long-term ECDH private key for the National Data Store
    $S \leftarrow$ ECDH$(K, R)$
    $K_{AES}, K_{MAC} \leftarrow$ MSB$(128)$, LSB$(128) \vdash$ SHA$(S)$
    $m \leftarrow$ HMAC$(k_{MAC}, R \parallel n \parallel c)$
    if $h \neq$ MSB$(m, 128)$ then
        abort
    end if
    $iv \leftarrow$ AES$(K_{AES}, n)$
    $p \leftarrow$ AES-CBC$(K_{AES}, iv, c)$
    return $p$
end procedure
```

Temporary identifier validity

Temporary identifiers are deanonymised and validated as follows:

Version 1

```
procedure DEANONYMISE($msg, t$)                    ▷ $t$ is the encounter timestamp
    $iv, c, a \leftarrow \ \vdash msg$
    $K \leftarrow$ RETKEY($\alpha$)
    $p \leftarrow$ AES-GCM($K, iv, a, c$)          ▷ AES-GCM will abort if integrity check fails
    $id, e \leftarrow \ \vdash p$
    if $e + 5$min $< t$ then
        abort
    end if
    return $id$
end procedure
```

Versions 2+

```
procedure DEANONYMISE($msg, t$)                    ▷ $t$ is the encounter timestamp
    $iv, c, h \leftarrow \ \vdash msg$
    $K \leftarrow$ RETKEY($\alpha$)
    $m \leftarrow$ HMAC($K, iv \parallel c$)
    if $h \neq$ MSB($m, 128$) then
        abort                                      ▷ See note below
    end if
    $p \leftarrow$ AES-CBC($K, iv, c$)
    $id, e \leftarrow \ \vdash p$
    if $e + 5$min $< t$ then
        abort
    end if
    return $id$
end procedure
```

In versions 2 – 3, instead of aborting if $h_0 \neq h_1$, the server would attempt the old deanonymisation procedure, for backward compatibility with earlier temporary identifiers. This fallback was removed in version 4, as all temporary identifiers generated by version 1 should have expired by that version.

Security properties

The mechanisms for securing encounter history uploads have already been described in:

- Temporary identifiers, Security properties
- Encounters, Security properties (versions 2+)
- Encounters, Security properties (versions 6+)

Health authority access

Authentication of health workers to the COVIDSafe portal is handled by AWS's identity management system Cognito.

Requests for health worker accounts are approved by the Australian Government Department of Health, in collaboration with state and territory health authorities. They are then created in Cognito by the data store administrator upon request from the Australian Government Department of Health.

Two-factor authentication is required for health workers to access the portal. On successful authentication, Cognito issues a JWS credential, stored by the browser as a cookie. The JWT includes the health worker's state or territory.

Health workers are able to view uploaded encounter histories by entering a registered phone number. The data store will only return encounter histories where *both* the following are true:

- the given phone number matches the registration record associated with the upload
- the state or territory calculated from the postcode in the registration record matches the state or territory in the health worker's JWT.

The ability to view encounter histories is therefore constrained by the JWS. For more information on JWS security, see Authentication tokens.

All health worker access and activity on the portal is audited, and can be reviewed by the Australian Government Department of Health. Audit records do not include personal information provided by users when registering or included in encounter records, other than the phone numbers entered by health workers.