Deniability in Automated Contact Tracing: Impossibilities and Possibilities

Christoph U. Günther
Institute of Science and Technology Austria
cguenthe@ista.ac.at

Krzysztof Pietrzak
Institute of Science and Technology Austria
pietrzak@ista.ac.at

ABSTRACT

Automated contact tracing (ACT) emerged as a promising measure to curb the spread of Covid-19. Users enable ACT on their smartphones to automatically record contacts with other users. If a user tests positive for the disease, they report their diagnosis to alert their contacts.

Designing effective ACT protocols is challenging since they need to be efficient and secure while also ensuring users' privacy. As ACT protocols necessarily leak some information by design, defining privacy is difficult. For example, a user cannot deny having met another user. Ideally, however, the user can plausibly deny everything else, in particular, when they met. We call this privacy property *contact-time deniability*.

While some early works discussed contact-time deniability informally, it has received little attention since then. We investigate deniability from a rigorous, theoretical point of view and arrive at the following impossibility result:

A decentralized protocol with unidirectional communication cannot be contact-time deniable and replay-secure. This holds even if malicious users treat smartphones as black-boxes.

Unidirectional protocols are usually very efficient and many proposals are unidirectional, e.g., the widely-deployed Google-Apple Exposure Notifications. So the impossibility result considerably constrains the design space of efficient, secure, and private ACT protocols. However, it can also be used as a guide; we discuss several possibilities to achieve contact-time deniability in practice.

KEYWORDS

automated contact tracing, deniability, replay security, impossibility result

1 INTRODUCTION

Automated contact tracing (ACT) emerged as one measure to fight the Covid-19 pandemic. ACT automatically identifies contacts of an infected person, enabling preventative measures to stop the disease's spread. While there are many conceivable ways to automate contact tracing (e.g., correlation of credit card transactions [22]), most approaches utilize smartphones interacting via Bluetooth Low Energy (BLE).

People install an ACT app to participate; the app continuously sends and receives messages over BLE to automatically record contacts with other app users. When a user is diagnosed, they report their diagnosis to the app and their past contacts promptly receive a digital alert.

The efficacy of ACT solutions crucially relies on high adoption. The fraction of detected contacts grows roughly quadratically with the fraction of app users amongst the population [14]. Therefore, prospective user must be confident in the app and, importantly, the underlying ACT protocol.

In this work, we examine the design space of ACT protocols from a theoretical point of view. We formalize security and privacy properties to show that a wide class of protocols cannot be secure (i.e., accurately alert users even under adversarial influence) and offer a strong form of privacy, contact-time deniability, at the same time. In other words, there is a tension between security and privacy properties, so the *ideal* ACT protocol does not exist. This explains some of the trade-offs that prior ACT protocol proposals had to strike.

1.1 ACT Protocols

Prior work proposes numerous protocol designs, e.g., [5, 6, 8, 11, 12, 16, 17, 21, 24, 25, 28–30, 32]. The most widely used ACT implementation is the Google-Apple Exposure Notification API (GAEN) [4] based on the Decentralized Privacy-Preserving Proximity Tracing (DP3T) [31] protocol. GAEN is a cross-platform API that implements the bulk of an ACT protocol allowing public authorities to easily publish an ACT app.

DP3T's design is simple and efficient. The phone broadcasts changing pseudorandom messages every couple of minutes; at the same time it listens for broadcast from other phones and stores them. When a user is diagnosed, they upload the keys used to generate the pseudorandom messages to a server. Phones regularly fetch newly added keys from the database, recompute the pseudorandom messages, and check whether any stored message matches. Since this check happens locally, the protocol is *decentralized*, hopefully increasing the privacy of users.

DP3T and also other protocol proposals received a lot of scrutiny. Multiple works analyzed ACT protocols in terms of efficiency, security[1], and/or privacy (for surveys, see e.g., [2, 3, 14, 27]). While efficiency (e.g., in terms of energy or mobile data usage) can be estimated from the protocol description and benchmarks, security and privacy require careful examination. Unlike most modern, provable cryptography, the majority of the aforementioned works informally define properties and evaluate protocols based on their resistance against specific attacks.

Defining privacy for ACT protocols is tricky because ACT necessarily leaks some information to correctly fulfill its purpose.[2] In general, the two most salient issues are preserving the privacy

Proceedings on Privacy Enhancing Technologies 2024(4), 636–648

https://doi.org/10.56553/popets-2024-0134

[1] Sometimes called "integrity".
[2] For example, a comic strip [10] depicts is how privacy might be violated during a job interview. The interviewer switches on a different phone for every candidate. A

of users' movements and their social interactions. However, even these are not set in stone and the public might prefer trading off privacy for, e.g., higher-fidelity epidemiological data.

Early on, this lead to a debate on whether decentralized protocols (contact check performed on the device, e.g., DP3T) or centralized protocols (contact check performed on the server) are more private [33]. The former might leak the movement patterns of reported users to the public but ensures privacy for non-reported users. The latter protects the privacy of all users against the public but places trust in the authority administrating the server.

1.2 Deniability

Deniability is a privacy property complementary to the ones mentioned so far. As before, the mere correctness of ACT protocols implies that the smartphones of users hold evidence of their recent encounters. For example, after users A and B have met, they cannot plausibly deny having been in close proximity to each other. Indeed, given access to both of their smartphones, a judge[3] uses A's phone to report A as sick and checks whether B's phone reports a contact.

Ideally, however, the data stored on users' smartphones should not reveal anything apart from whether they met or not. In particular, if there was a contact, it should be impossible to pin down *when* this contact happened. We call this form of deniability *contact-time deniability*.

It turns out that many protocols are not contact-time deniable. Essentially, a malicious user who deviates from the protocol can construct digital evidence that they encountered someone [6, 24, 33]. So it is an "edge of the social graph [coming] together with a proof" [33]. To illustrate how this might be problematic, consider the following hypothetical scenario.

Example 1. Consider a country that is composed of federal states, and that, say, a medical procedure P is a felony in state X but legal in another state Y. Furthermore, state X not only criminalizes this procedure within its own borders, but also prosecutes its citizens when they have this procedure performed in other states. Now, for a thought experiment, assume that an ACT protocol is used country-wide across all states and that the protocol does not offer contact-time deniability.

Someone might claim to have seen a user A, a citizen of state X, at the medical facility in state Y. Furthermore, they accuse A of having procedure P performed there at a certain time and substantiate their claim with, say, recorded ACT protocol BLE messages from user A's phone. So a judge confiscates A's phone and forces the phone to report A as sick (possibly by getting a fake test result from a regional testing facility). We emphasize that the judge does not have any privileged access to the ACT protocol and the servers running it. They only uses their powers *regionally* in state X. If the protocol were deniable, the accused user A could claim being framed and that they were simply in the area, say, on a weekend when the medical facility was closed.

couple of days later, they check if any phone registered a contact and deduce that the corresponding candidate tested positive.

[3] An entity to be convinced of whether a contact happened based on the given evidence.

1.3 Our Contributions

Surprisingly, we could only identify three papers [6, 24, 33] informally discussing deniability. Apart from that, no attention has been paid to deniability, especially not in a more formal manner. This includes informal analyses and surveys [2, 3, 6, 14, 18, 19, 27, 32, 33] as well as the works rigorously modeling security and privacy properties [8, 12, 15, 17, 20]. Our contribution is to remedy this state of affairs.

First, we give a rigorous and fine-grained deniability definition: Δ-contact-time deniability.

Definition (Informal)**.** An ACT protocol is *not* Δ-contact-time deniable if a malicious user M that meets an honest user A at time t can provide digital evidence to a judge that they have been in A's proximity at some point during the timespan $t - \Delta$ to $t + \Delta$.

Second, our main result shows a tension between Δ-contact-time deniability and Δ'-replay security, a fundamental security property. Ideally, a protocol fulfills both properties with Δ large (say, two weeks) and Δ' small (say, 2 seconds). Unfortunately, this is often not possible as the following impossibility theorem shows.[4]

Main Theorem (Informal; cf. Theorem 1)**.** *A correct, decentralized ACT protocol with unidirectional communication (like DP3T/GAEN) cannot be Δ-contact-time deniable and Δ'-replay-secure with $\Delta \geq \Delta'$ when devices are treated as black-boxes.*

While the theorem only holds for decentralized, unidirectional protocols, this covers a large class of efficient protocols and thus also proposals (e.g., [12, 17, 24, 25, 28, 29, 31]). Nevertheless, we modify the theorem to hold for *any* decentralized protocol *if* the judge gets slightly more capabilities. Furthermore, we argue that these impossibility results apply to centralized designs in practice. We just cannot capture centralized protocols entirely because their design space is too large in theory.

1.4 Technical Overview

Let us briefly state a high-level proof of the main theorem. In addition to Δ-contact-time deniability, we will need the following definitions.

Correctness ensures that an honest user gets alerted if they were in contact with another honest user that reported as infected later on.

Decentralized protocols perform all computation on the users' phones; the server only acts as a database.

Unidirectional communication means that messages from one user to another are sufficient to record a contact (e.g., DP3T). This property usually implies that a protocol is efficient since it precludes interactive protocols requiring session state.

Δ-replay-security ensures that a malicious user M cannot record an interaction with an honest user A, wait for Δ time, and then resend the interaction (or information derived from it) to an honest user B such that a contact between A and B is recorded.

Black-box access to devices means that malicious users and the judge may only use the normal functionalities exposed by

[4] Notions yet to be defined, e.g., correctness or black-box model, are somewhat self-explanatory; otherwise, Section 1.4 will cover them momentarily.

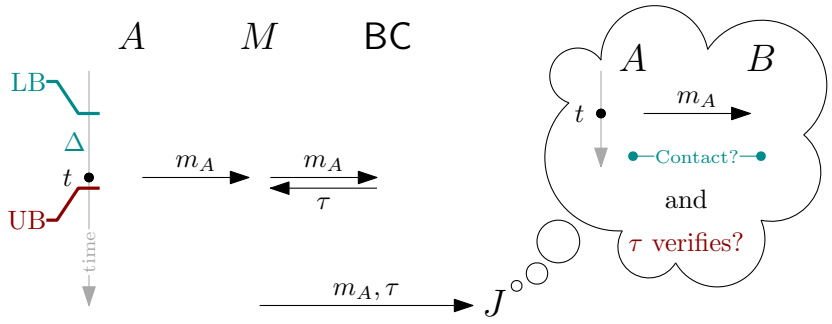


Figure 1: Intuitive proof sketch of the impossibility result.

devices, e.g., reporting as sick or setting the device's time. In particular, the cannot, e.g., inspect or modify devices' memory directly.

Timestamping services allow someone to submit a message and receive a certificate publicly proving that they submitted (and therefore knew) the message at the time of submission. This may be realized by, e.g., posting the hash of the message to a popular blockchain.

Without loss of generality, we ignore the distinction between Δ and Δ' in the theorem and prove the following: A decentralized, unidirectional, and correct protocol cannot be Δ-replay-secure and Δ-contact-time deniable for any Δ.

Proof (Sketch). The proof sketch is illustrated in Figure 1 and explained below.

Consider a decentralized, unidirectional ACT protocol that is correct and Δ-replay-secure. Let M be a malicious user that wants to give evidence to a judge J that they were in contact with an honest user A at a certain time t.

When M receives message m_A from A, they immediately timestamp it by posting the hash of m_A to the blockchain BC resulting in timestamp τ. At a later point in time, M goes to the judge J with this timestamped message (m_A, τ). Following that, judge J confiscates A's phone. Using A's phone, m_A and τ, J performs two tests (depicted in the thought bubble).

- J uses a dummy phone to register a dummy user B and replays the message m_A at time t to it. Note that J can turn the time back by just modifying the clock in the dummy phone's settings. Then, J forces A to report as infected and checks whether B got a notification (depicted as "Contact?").
- J verifies that τ is a timestamp for m_A at time t (depicted as "τ verifies?").

If B got a notification, J knows that A and M were in contact at some point. Since the timestamp verifies, J knows that they were in contact by time t at the latest (i.e., an upper bound UB on the contact time).

Now, for the crucial part of this impossibility result: Since the protocol is Δ-replay-secure, J also has a lower bound LB of $t - \Delta$ on the contact time—otherwise replaying m_A to the dummy user B would not have lead to a contact. Thus, J is certain that M was in contact with A in the timespan $t - \Delta, \ldots, t$, so the protocol is at most $(\Delta - 1)$-contact-time deniable. □

2 ACT PROTOCOLS

The design space for ACT protocols is quite large. As mentioned before, we restrict our attention to proximity-based ACT where devices exchange messages using, e.g., Bluetooth Low Energy (BLE). While this still allows for a multitude of designs, all protocols adhere to the following high-level blueprint [14, 33]:

Setup to participate in the protocol (possibly in concert with, e.g., health care authorities).

Interaction with other devices by sending and receiving BLE messages.

Reporting a positive test result (possibly controlled by, e.g., testing facilities).

Checking whether this device has been in contact with the device of a user who reported a positive test result.

In all these steps, the user's device may interact with *servers* over the internet. Usually, these are servers hosted by public authorities or are decentralized (e.g., a blockchain).

The role of the servers varies considerably between protocol, yielding three broad classes [2, 14, 33]

Centralized protocols require users to sign-up (e.g., solving a CAPTCHA or providing their ID) before participating and handles most things server-side. That is, devices only send server-prescribed messages and upload all received messages to the server, enabling server-side checking for contacts.

Decentralized protocols, in contrast, only use the server as a database or bulletin board. Users do not have to sign up, generate all messages on device, and report their diagnosis by uploading data to the database. Other users regularly download new data and use it to check for contacts with diagnosed users locally on-device.

Hybrid protocols capture everything that falls in between the preceding ones. So requiring user sign-ups is possible, but not strictly necessary. Similar to decentralized protocols, messages might also be generated on device. Checking for contacts may be performed server-side or using an interactive protocol between devices and the server (e.g., private set intersection).

Note that some specific implementations might diverge slightly from these classes, but we will disregard that in our theoretical analysis. For example, in real-world deployments, decentralized protocol only allow users to report a diagnosis if they have a valid

test. Otherwise, malicious users could cause notifications that are not related to an actual case of infection.

2.1 Existing Constructions

Throughout the paper we will refer to a selection of protocols that are instructive for our impossibility result. A simplified description of these protocols follows.

Note that some designs may seem needlessly complex at a first glance. These complexities are either the result of reducing the messages sizes or storing as little information in plain text on a device. The latter strives to improve privacy of users against white-box attacks where the adversary has direct access to the device's memory. For example, not storing the time and/or location of a contact in plaintext makes reconstructing the user's movements more difficult.

DP3T. DP3T [31] is a decentralized protocol and the basis for the most widely deployed ACT solution, Google/Apple exposure notifications (GAEN) [4]. While GAEN and DP3T differ in implementation details, their basic principle is the same.

Every day d, the device samples[5] a new key SK_d locally. Then, SK_d is expanded and divided into so-called ephemeral IDs using

$$\mathsf{PRG}(\mathsf{PRF}(\mathsf{SK}_d, \text{“broadcast key”})) = \mathsf{EphID}_0||\mathsf{EphID}_1||\cdots. \quad (1)$$

The ephemeral IDs are broadcast in a random order via BLE throughout the day in fixed intervals. At the same time, the device stores received ephemeral IDs together with the current day.

To report a positive test, the device publishes the keys of the preceding, say, week (or whatever period is deemed epidemiologically relevant). To this end, the device uploads $\mathsf{SK}_{d-7}, \ldots, \mathsf{SK}_d$ to the database.

Other users periodically download newly published keys from the database. To check for contacts with diagnosed users, the device uses the keys to recompute the ephemeral IDs for every day. Then, the recomputed ephemeral IDs are compared to the stored ones for that day. If there is a match, the device notifies the user of the potentially dangerous contact.

Challenge-Response. Vaudenay [32] noted that DP3T is vulnerable to replay attacks within the same day. They construct a replay-secure but interactive protocol.

It uses Ephemeral ID generated like in DP3T (cf. Equation (1)). In addition, ephemeral secret keys are generated similarly, i.e.,

$$\mathsf{PRG}(\mathsf{PRF}(\mathsf{SK}_d, \text{“secret key”})) = \mathsf{EphSK}_0||\mathsf{EphSK}_1||\cdots. \quad (2)$$

The interaction between devices works as follows. The sender advertises EphID_i, the receiver responds with a random challenge c, and the sender returns $\mathsf{MAC}(\mathsf{EphSK}_i, c) = \mathsf{tag}$ where MAC is a message authentication code. Consequently, the receiver stores $(\mathsf{EphID}_i, c, \mathsf{tag})$.

Like in DP3T, a user reports their diagnosis by uploading the keys SK_j. Given the keys, other users can recompute ephemeral IDs as before and also verify the MAC tag τ.

Note that the stored triple $(\mathsf{EphID}_i, c, \mathsf{tag})$ does not contain or depend on the time t. This is beneficial for user privacy because it is resistant against white-box analysis as described at the beginning of this section.

[5] Often, it derives it from a longer-term key.

Delayed Authentication. Pietrzak [24] proposed a non-interactive replay secure protocol which requires the devices to have reasonably synchronized clocks.

Ephemeral IDs and keys are generated as in Equations (1) and (2). At time t, the sender transmits the message

$$(\mathsf{EphID}_i, \rho, \mathsf{tag}, t)$$

where ρ is a random string and

$$\mathsf{tag} = \mathsf{MAC}(\mathsf{EphSK}_i, \mathsf{Hash}(t||\rho)) \quad (3)$$

with || denoting concatenation. Upon receipt, the receiver checks whether their own time t' is close enough to t (say ± one second). If it's not, they ignore the message as it could be a replay attack attempt. Otherwise they compute $\sigma_i = \mathsf{Hash}(t||\rho)$ and then store $(\mathsf{EphID}_i, \mathsf{tag}, \sigma_i)$. Again, when tested positive, a user publishes their keys and others check for contacts by recomputing ephemeral IDs and verifying tag.

If coarse location data, like GPS coordinates, is available to the protocol, it can be augmented to also protect against *relay* attacks. Relay attacks are the location analogue of replay attacks, they prevent an attacker from replaying messages at different location (even if this happens in almost real time, and thus replay security would not prevent this).

Note that like in Challenge-Response the users don't store the contact time t (or location data); while σ_i was computed using t, by using a sufficiently large ρ, the σ_i will be statistically independent of t. In cryptographic terms, the receiver stores a statistically hiding commitment of the contact time and a message authentication tag of this commitment. The tag can be verified (and a contact is detected) if the sender reports sick and the receiver learns their MAC key. This ensures that a user's movement patterns stay secret even if their phone is forensically analyzed at a later point in time. Let us stress that this is an orthogonal issue to contact-time deniability, which considers users actively deviating from the protocol with the goal to create digital evidence of contact time.

CleverParrot. Canetti et al. [12] propose CleverParrot, a decentralized protocol that follows a different design philosophy. In the protocols described so far, a diagnosed user uploads their keys, a concise description of all their sent messages. CleverParrot, instead, requires diagnosed users to upload all received messages of the past, say, week. Senders download these messages and check whether they have sent them.

In more detail, at time t, user A broadcasts $m = \mathsf{Hash}(t)^{\mathsf{SK}_A}$ and the receiver stores (m, t). Here, Hash now maps to a group of prime-order p and $\mathsf{SK}_A \in \mathbb{Z}_p$ is the sender's secret key. For security, the discrete logarithm problem must be hard in the chosen group.

If a user is tested positive, for every stored broadcast (m, t), they compute $(m^\rho, \mathsf{Hash}(t)^\rho)$ with random $\rho \in \mathbb{Z}_p$, shuffle the resulting list, and upload it. Then, any other user A learns of a contact if the uploaded list contains an element (u, v) such that $u = v^{\mathsf{SK}_A}$, i.e., one of A's broadcasts re-randomized.

The re-randomization ensures that other users can only recognize their own messages. This is boon to privacy since, e.g., other users cannot find out whether they and the diagnosed users had mutual contacts. In terms of security, the protocol hinders replay attacks by hashing the time.

Decentralized Diffie-Hellman. Performing a Diffie-Hellman (DH) key-exchange is a natural way to implement a decentralized ACT protocol (e.g., [6, 21]). User A has the secret key α and broadcasts g^α where g is the generator of a group where the DH problem is assumed to be hard. Upon receiving g^β from user B with key β, A stores the shared secret $(g^\beta)^\alpha = g^{\alpha\beta}$ which is identical to what B stores. To report a positive test, A uploads $g^{\alpha\beta}$ and B can easily check whether they were in contact.

While this protocol is non-interactive (unlike Challenge-Response), it still requires *bidirectional* communication in contrast to DP3T, Delayed Authentication, and CleverParrot. Interestingly, this slight relaxation from uni- to bidirectionality allows for a protocol that is replay secure and deniable at the same time.

NTK. NTK [23] is a centralized protocol implementing the PePP-PT[6] design. It is similar to ROBERT [13], also following PePP-PT, but slightly simpler to describe.

User i registers with the server and receives a unique identifier ID_i. The server can use ID_i to send a notification to the user's device.

For every, say, hour h, the central server has a secret key SK_h. The user's device regularly queries the server for new ephemeral IDs. In response, the server computes

$$\begin{aligned}\mathsf{ENC}(\mathsf{SK}_h, \mathsf{ID}_i) &= \mathsf{EphID}_0\\ \mathsf{ENC}(\mathsf{SK}_{h+1}, \mathsf{ID}_i) &= \mathsf{EphID}_1\\ &\vdots\end{aligned}$$

and returns the list $(\mathsf{EphID}_0, \mathsf{EphID}_1, \ldots)$ to the user. This list covers, e.g., a days worth of ephemeral IDs. Like in DP3T, the user's device broadcasts the ephemeral IDs throughout the day; when it receives an EphID from another device, it stores the tuple (EphID, h) where h is the current hour.

When a user is diagnosed, they upload all received tuples to the server. For every tuple (EphID, h), the server fetches the corresponding key SK_h and decrypts the ephemeral IDs to get the sender's identity. This allows the server to notify the sender of the contact with a diagnosed user.

Hybrid Diffie-Hellman. Apart from being a popular decentralized protocol, DH key-exchange can also be used for hybrid designs. DESIRE [11] is one such implementation that derives some of its privacy guarantees from the use of trusted hardware (e.g., Intel SGX). As we will see, however, trusted hardware is not necessary in theory.

Hybrid DH is very similar to Decentralized DH. Users sample their own private keys, perform a non-interactive key exchange with users in their proximity, and also upload the shared secrets upon diagnosis. The difference lies in how other users check for contacts. In DESIRE, other users periodically upload their stored shared secrets to the central server and the server performs the matching. Importantly, all server-side computation is performed in trusted Intel SGX enclaves to ensure privacy.

Instead of relying on trusted hardware such as Intel SGX, it is possible to perform this step using a private set intersection cardinality protocol [30] or, ideally, a private set intersection test [33]. This is essentially a protocol that returns a boolean indicating whether two sets (one held by the server, one by the user's device) intersect. In theory, this is an elegant solution, but presumably too inefficient for practice.

[6] https://www.pepp-pt.org/

3 PROPERTIES OF ACT PROTOCOLS

So far, we have seen multiple constructions of ACT protocols. Using these existing protocols as a guide, we will first formalize some basic terminology and then discuss three properties that protocols should fulfill.

3.1 Basic Terminology

Devices. All ACT protocols have in common that users interact with others automatically using their devices (e.g., smartphone or a small BLE-capable device). For user A, we denote their device by dev_A which communicates with other devices over BLE. Since BLE communication is not fully reliable, message delivery is not always guaranteed in practice.

Decentralized Protocols. As stated before, decentralized protocols use the server as a database or bulletin board. We denote this server by db and assume that it is solely used in a database manner. That is, devices can push data to db and query it for new data. We emphasize that db cannot generate key material or run any other general-purpose computation.

Adversarial Models. We assume that the BLE communication between honest devices cannot be blocked as the adversary could trivially violate correctness otherwise. The adversary can passively listen and also inject additional messages.

Additionally, we need to specify how the adversary can access its own devices and the devices of users it is given explicit access to. We distinguish between three different models.

Black-box access assumes that the adversary has the same access to devices as normal users. So they can interact with the protocol following its specification (i.e., whatever the phone and ACT app allow) and change the user-facing settings of the device. In particular, the adversary can set the local time of the device.[7]

White-box access give the adversary full access to the device. In particular, they can read and modify the entire memory. This allows accessing information or participating in the protocol in a way that honest users are not able to.

Gray-box access lies in-between black and white-box access. In this paper, we define it like black-box access with the only addition being that the adversary can fix the randomness used by the device and the ACT app. We will motivate this unconventional model towards the end of the paper in Section 5.2.

Contacts. The definition of contacts is central to ACT. It needs to be broad enough to capture all possible protocols. Cicala et al. [14] informally analyze how various protocols report positive test results. They define three types of protocols where the uploaded data depends on sent messages, received messages, or on sent and

[7] Iovino et al. [19] consider "time-travel" attacks where they even manage to set the time of victims remotely without interaction.

received messages. This roughly equals our following, slightly more formal definition.

Definition 1 (Contact)**.** Consider a protocol interaction between two devices dev_A and dev_B where dev_A sends the initial message. This interaction results in a *contact* if the joint state of the devices after the interaction is such that at least one of the following holds:

(1) If A reports sick, then B will be alerted *(AB-contact)*; or
(2) If B reports sick, then A will be alerted *(BA-contact)*.

Note that DP3T, Challenge-Response, and Delayed Authentication have *AB*-contacts while NTK and CleverParrot have *BA*-contacts. DH-based protocols have *AB* and *BA*-contacts because it does not matter who uploads the shared key.

Message Flow. A special class of protocols are *unidirectional* protocols such as DP3T, Delayed Authentication, CleverParrot, and NTK. This is because every interaction leading to a contact has a distinct sender and receiver where only the sender sends messages (usually only a single message). In contrast, Challenge-Response is clearly *bidirectional* and DH-based protocols as well, albeit for a subtler reason. Essentially, DH-based protocols require *both* devices to send a message whereas the receiver in a unidirectional protocol does not need to send any message. This leads us to the following definitions.

Definition 2 (Unidirectional Protocol)**.** If a contact (Definition 1) between dev_A and dev_B is established while dev_B does not send any message, a protocol is *unidirectional*

Definition 3 (Bidirectional Protocol)**.** A protocol is *bidirectional* if and only if it is not unidirectional (Definition 2).

At a glance, classifying protocols by message flow does not seem meaningful. First, however, it is a rough estimate for the energy efficiency of the protocol. Unidirectional protocols are non-interactive and thus usually amount to devices broadcasting a single message. This is energy-efficient in contrast to interactive protocols which are more complex. For example, Challenge-Response requires three messages and constantly keeping track of session state.

Second, unidirectional protocols are more robust than bidirectional ones. In a unidirectional protcol, a contact is established as soon as dev_B receives dev_A's broadcast. In contrast, bidirectional protocols (e.g., DH-based protocols) require that *both* devices receive broadcasts from the other one. So bidirectional protocols are not as robust as unidirectional ones against, e.g., BLE message transmission failures.

3.2 Correctness

Naturally, a protocol should be correct, i.e., whenever two users meet and one user reports as sick later on, the other user is informed.

Definition 4 (Correctness)**.** A protocol is correct if at least one of the following holds:

(1) Whenever two devices interact, an *AB*-contact is recorded; or
(2) Whenever two devices interact, a *BA*-contact is recorded.

Different Ways to Define Correctness. In principle, one might come up with a Frankensteinian protocol that alternates between executing a protocol fulfilling *either* Item 1 *or* Item 2 *exclusively*. For example, a protocol could flip a coin and execute DP3T or CleverParrot depending on the outcome. Such a protocol is not correct by Definition 4, but one might reasonably argue that it should be. A suitable alternative definition capturing Frankensteinian protocols would be the following:

> A protocol is correct if, whenever two devices interact, at least one of the following holds: Either an *AB*-contact or a *BA*-contact is recorded.

We chose Definition 4 for two reasons. First, we do not know of any ACT protocol that is only correct according to the alternative definition but not according to Definition 4. Second, in any case, our results could be adapted to work with the alternative definition at the cost of making our arguments harder to follow and verify.

3.3 Replay Security

Security of ACT protocols is concerned with integrity of contacts. It ensures that a user is notified of a contact with a diagnosed user only if they actually met the user. In other words, interacting with a malicious user cannot cause honest users to register additional, spurious contacts with other honest users.

There are multiple security properties one could consider (see e.g., [32, §4], [6, §4], [14, §4.3], and [12, §6.2]). We will focus on a well-known, baseline security property—replay security. Note that the following definition is parameterized with respect to time.

Definition 5 (Δ-Replay Security)**.** A protocol is Δ-replay secure if every malicious user M wins the Δ-replay security game (Figure 2) with at most negligible probability.

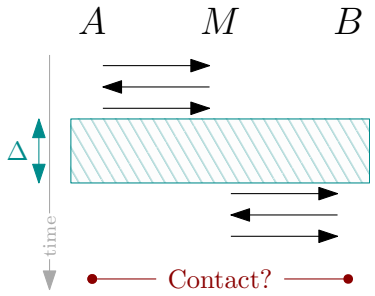


Figure 2: Replay security game.

In the replay security game (Figure 2), a malicious user M first interacts with an honest user A. Then, after Δ time has passed, M interacts with another honest user B. Here, M might, for example, re-send messages it previously received from A. A replay-attack is successful if M makes A and B establish a contact even though the never interacted with each other.

Interpreting Δ. The Δ parameter offers a parameterized analysis of replay security. This is important because time in ACT protocols is usually quite granular in practice. For example, the time t in Delayed Authentication (cf. Equation (3)) might only have a resolution of 1 second. Strictly speaking, this is not replay-secure because an adversary might perform a replay attack within, say, 10 seconds. However, according to Definition 5 it is (1 second)-replay secure. Ideally, a protocol is Δ-replay secure for some small Δ.

Replay Security of Existing Constructions. Challenge-response and DH-style protocols are interactive, so it is not hard to see that they are 0-replay-secure. Delayed Authentication and CleverParrot are Δ-replay-secure where Δ depends on assumptions on the synchrony of devices' clocks and the latency of BLE broadcasts. Replay-security of DP3T and NTK additionally depends on how the key schedule is implemented, i.e., how often fresh keys are sampled. With the (realistic) parameters stated in Section 2.1, DP3T is (1 day)-replay secure and NTK (1 hour)-replay secure.

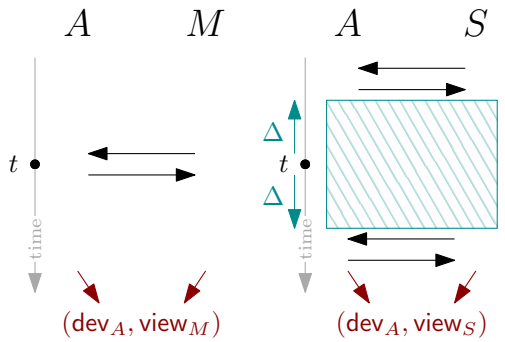
3.4 Deniability

Deniability of the contact time is a privacy property. Privacy properties ensure that honest users' behavior cannot be tracked. Besides deniability, there exist a many desirable privacy properties (see e.g.,[32, §5],[6, §3],[14, §4.1], and [12, §6.2]) covering the privacy of certain information (e.g., contact time or social interaction graphs) in different adversarial models (e.g., privacy for users who report sick or privacy from a malicious authority).

Intuitive Definition. Consider a malicious user M that claims to have interacted with honest user A at time t. If the protocol is Δ-contact-time deniable, then A can plausibly deny meeting M at time t. Instead, A can claim to have actually met M at a different time t' that is more than Δ time steps away from t, i.e, $|t - t'| > \Delta$. This holds even when A is forced to hand over their device (denoted by dev_A).

Formal Definition. The intuitive definition as stated is hard to capture formally. As is common in cryptographic literature, we define deniability using simulation.

Definition 6 (Δ-Contact-Time Deniability)**.** A protocol is Δ-contact-time deniable if, for every malicious user M and every judge J, there exists a simulator S such that J has negligible advantage in distinguishing $(\mathsf{dev}_A, \mathsf{view}_M)$ from $(\mathsf{dev}_A, \mathsf{view}_S)$ as in Figure 3.



(a) Claimed contact. **(b) Simulated contact.**

Figure 3: Deniability games.

In the real execution (Figure 3a), M only interacts with A at time t and produces some view view_M. The judge J then gets this view together with the claimed time of contact t and black-box access to dev_A.

Table 1: Properties of the existing constructions. ~ indicates a trade-off between replay security and contact-time deniability based on implementation parameter choices.

Protocol	Unidirectional	Replay-secure	Deniable
DP3T	✓	~	~
Challenge-Resp.	×	✓	✓
Delayed Auth.	✓	✓	×
CleverParrot	✓	✓	×
NTK	✓	~	~
DH-based	×	✓	✓

In the simulated execution (Figure 3b), S is allowed to interact with A at any time except close to time t. That is, S does not communicate with A in the interval $(t - \Delta, t + \Delta)$ and then outputs a view view_S. Similarly to before, J gets $(\mathsf{dev}_A, \mathsf{view}_S)$.

Interpreting Δ. Intuitively, Δ describes *how much* plausible deniability a user has. For example, consider a user who is a suspect in a break-in [33] and BLE messages have been recorded at the time of the break-in. If Δ equals a week, then the user can plausibly convince a judge that their encounter has nothing to do with the break-in. However, if Δ is 5 minutes, then a similar argument would fall flat. So protocols should strive to be Δ-contact-time deniable for large Δ.

Note that Δ larger than one to two weeks is not achievable in practice. This is because the concrete implementation of an ACT protocol usually only considers contacts that occurred within an epidemiologically relevant timespan (e.g., for Covid-19 this was oughly two weeks). So any evidence of contact reveals that the contact happened within that timespan.

Deniability of Existing Constructions. Bidirectional protocols like Challenge-Response and DH-based protocols are contact-time deniable. DP3T and NTK are Δ-deniable with $\Delta <$ 1 day and 1 hour, respectively. The timespan is complementary to replay security (which depends on how the key schedule is implemented). Essentially, the protocols are only deniable in the timespan where replay attacks are possible.

Intuitively, whatever these protocols transmit is independent of the time (or at least the precise time like in DP3T or NTK). It follows, that the interaction could have happened at any time (or only within the imprecise window). Conversely, the replay-secure protocols Delayed Authentication and CleverParrot are not deniable.

3.5 Summary of Existing Constructions

We have described which properties the existing constructions from Section 2.1 fulfill. Table 1 summarizes this and we note that the table reflects the impossibility result.

4 DECENTRALIZED UNIDIRECTIONAL PROTOCOLS

Recall that our aim is to prove an impossibility result for decentralized unidirectional protocols. Compared to ACT protocols in general, this class of protocols has a simple structure. This allows us

to give precise game-based definitions of the properties described in the previous section.

4.1 Model and Syntax

We consider decentralized unidirectional protocols (DU protocols) where all users and adversaries are probabilistic polynomial-time (PPT) algorithms. We restrict ourselves to modeling time and ignore other measurements such as location. So a user can either be active or inactive at a certain time.

Time. Time proceeds in discrete steps starting at 0, i.e., time $\in \mathbb{N}$, and we assume that all honest devices have synchronized clocks (e.g., using NTP). In the following, $T \in \mathbb{N}$ is some appropriately large upper bound on the time span that we consider (think of it as, say, 2 weeks in the case of Covid-19).

In practice, blockchains exist and, amongst other things, offer a timestamping functionality. We use the following simple syntax to capture this.

Definition 7 (Timestamping Oracle)**.** A timestamping oracle provides two functions:

- $\mathsf{TS}(x, t) \to \tau$: On input x at time t produces a timestamp τ.
- $\mathsf{VerifyTS}(\tau, x, t) \to b$: Returns a boolean $b \in \{\top, \bot\}$ stating whether τ is a timestamp for x at time t.

The timestamping oracle fulfills correctness and unforgeability which are defined in the natural manner.

We assume that all (malicious) users have implicit access to VerifyTS. Looking ahead, queries to TS will be mediated by the security- and privacy games.

Protocol Syntax. A decentralized unidirectional protocol offers the following functions to a user A. Here, we assume that the protocol consists of a *single* unidirectional message without loss of generality.

- $\mathsf{Register}() \to \mathsf{dev}_A$: Registers A and returns an initial device state dev_A.
- $\mathsf{Broadcast}(\mathsf{dev}_A, t) \to m$: Broadcasts A's message m at time t.
- $\mathsf{Receive}(\mathsf{dev}_A, t, \boldsymbol{m})$: Receives all messages $\boldsymbol{m}$ sent at time t.[8]
- $\mathsf{Report}^{\mathsf{db}}(\mathsf{dev}_A)$: Reports A a positive test by uploading data to db.
- $\mathsf{Check}^{\mathsf{db}}(\mathsf{dev}_A) \to n$: Queries db for new data and returns the number of contacts $n \in \mathbb{N}$.

Note that some functions expose the time t as a parameter. As mentioned, honest users have synchronized clocks and always set t correctly. However, t can easily be set incorrectly by modifying the phones settings. Therefore, we consider this black-box access, so malicious users may set t arbitrarily.

4.2 Correctness

We formalize Definition 4 using a game in which the adversary M schedules two honest users A and B.

Definition 8 (DU Correctness)**.** A DU protocol is correct if at least one of the following holds:

(1) For every PPT algorithm M, $\Pr[\mathsf{Correct}(A, B) \to \top] = 1$ (AB-correct); or
(2) For every PPT algorithm M, $\Pr[\mathsf{Correct}(B, A) \to \top] = 1$ (BA-correct).

Here, $\mathsf{Correct}(A, B)$ is as in Figure 4 and the probability is taken over the randomness of M, A, and B. $\mathsf{Correct}(B, A)$ is defined analogously, with dev_A and dev_B swapped in Lines 7 to 8.

$\mathsf{Correct}(A, B)$**:**

```
01 Register() → dev_A and Register() → dev_B
02 M → a, b ∈ {⊤, ⊥}^T
03 For i = 0, ..., T:
04     If a_i = ⊤ and b_i = ⊤:
05         Broadcast(dev_A, i) → m_A
06         Receive(dev_B, i, (m_A))
07 Report^db(dev_A)
08 Output ⊤ if and only if Check^db(dev_B) = |{i: a_i = b_i = ⊤}|
```

Figure 4: DU protocol correctness game.

In Figure 4, M schedules two honest users A and B (Line 2), but cannot interact with them in any other way. Then, in the timespan 0 to T, A only broadcasts and B only receives (Lines 3 to 6), so the number of registered contacts is known. Last, depending on Items 1 to 2, B (resp. A) reports sick (Line 7), and the game checks whether A (resp. B) is notified of the actual number of interactions (Line 8).

4.3 Replay Security

We translate Figure 2 to a game. Similar to the correctness definition (cf. Definition 4), the game is parameterized to account for AB and BA-correct protocols.

Definition 9 (DU Δ-Replay Security)**.** Let $\Delta \in \mathbb{N}$. A DU protocol is replay-secure if, for every PPT algorithm M, the Δ-replay security game (Figure 5) outputs $\top$ (i.e., replay-secure) except for negligible probability.[9] That is,

$$\Pr\left[\mathsf{Replay}_\Delta(A, B) \to \top\right] \geq 1 - \mathrm{negl}$$

and

$$\Pr\left[\mathsf{Replay}_\Delta(B, A) \to \top\right] \geq 1 - \mathrm{negl}$$

where $\mathsf{Replay}(A, B)$ is as in Figure 5 and the probability is taken over the randomness of M, A, and B. $\mathsf{Replay}(B, A)$ is defined analogously with dev_A and dev_B swapped in Lines 15 to 17.

In Figure 5, M picks a time t to perform a replay attack at and schedules the two honest users (Lines 2 to 3). Then, M interacts with A in the timespan 0 to $t - \Delta - 1$ (Lines 5 to 9). Afterward, it waits for Δ steps of time before interacting with B (Lines 10 to 14). At the end of the game, M wins (so the output is $\bot$) if and only if it created spurious contacts (Lines 15 to 17).

We will later use the following claim that directly follows from the definition.

Claim 1. For any $\Delta, \Delta' \in \mathbb{N}$ with $\Delta' > \Delta$, if a DU protocol is Δ-replay-secure, then it is also Δ'-replay-secure.

[8] Vectors are denoted by boldface, e.g., $\boldsymbol{m}$.

[9] In the interest of readability, we omit the security parameter and just write negl.

$\mathsf{Replay}_\Delta(A, B)$:

01 $\mathsf{Register}() \to \mathsf{dev}_A$ and $\mathsf{Register}() \to \mathsf{dev}_B$
02 $M \to t \in \mathbb{N}$ with $\Delta < t \leq T$
03 $M \to \boldsymbol{a}, \boldsymbol{b} \in \{\top, \bot\}^T$
04 For $i := 0, \ldots, T$:
05 If $i < t - \Delta$ and $a_i = \top$:
06 $\mathsf{Broadcast}(\mathsf{dev}_A, i) \to m_A$
07 $m_A \to M$
08 $M \to \boldsymbol{m}_M$
09 $\mathsf{Receive}(\mathsf{dev}_A, i, \boldsymbol{m}_M)$
10 If $i \geq t$ and $b_i = \top$:
11 $\mathsf{Broadcast}(\mathsf{dev}_B, i) \to m_B$
12 $m_B \to M$
13 $M \to \boldsymbol{m}_M$
14 $\mathsf{Receive}(\mathsf{dev}_B, i, \boldsymbol{m}_M)$
15 $\mathsf{Check}^{\mathsf{db}}(\mathsf{dev}_A) \to n$
16 $\mathsf{Report}^{\mathsf{db}}(\mathsf{dev}_B)$
17 Output $\top$ if and only if $\mathsf{Check}^{\mathsf{db}}(\mathsf{dev}_A) \leq n$

Figure 5: DU protocol Δ-replay security game.

4.4 Deniability

We cast the claimed and simulated interaction depicted in Figure 3 as games. Note that we explicitly model the access to the timestamping oracle TS, and that this access is mediated by the game.

Definition 10 (Δ-Contact-Time Deniability)**.** Let $\Delta \in \mathbb{N}$. A DU protocol is Δ-contact-time deniable if, for every PPT algorithm M and every PPT algorithm J, there exists a PPT algorithm S such that

$$\left|\Pr\left[J\left(\mathsf{Deny}_\Delta^{\mathsf{Real}}\right)\right] - \Pr\left[J\left(\mathsf{Deny}_\Delta^{\mathsf{Sim}}\right)\right]\right| \leq \mathrm{negl}$$

where $\mathsf{Deny}^{\mathsf{Real}}$ and $\mathsf{Deny}^{\mathsf{Sim}}$ are as in Figure 6 and the probabilities are taken over the randomness of M, S, and A.

Initially, M (Figure 6a) and S (Figure 6b) output t and schedule A (Lines 2 to 3). Then, both M and S interact with A but at different times. M interacts with A exactly at time t and S in the time spans 0 to $t - \Delta - 1$ and $t + \Delta + 1$ to T (Lines 5 to 9). At every step in time, both M and S may perform some computation and query TS (Line 10), no matter whether they interacted with A or not. In the end, both output a view (Line 11). The judge J receives the output of the game, i.e., the view together with A's device (Line 12). We emphasize that J only has black-box access to dev_A.

5 IMPOSSIBILITY RESULTS

Equipped with the DU protocol definitions, we can now state the impossibility results. We will first state the impossibility result in the black-box model which only applies to decentralized, unidirectional protocols. Then, we generalize to all decentralized protocols by working in the gray-box model.

$\mathsf{Deny}_\Delta^{\mathsf{Real}}$:

01 $\mathsf{Register}() \to \mathsf{dev}_A$
02 $M \to t \in \mathbb{N}$ with $\Delta < t \leq T$
03 $M \to \boldsymbol{a} \in \{\top, \bot\}^T$
04 For $i := 0, \ldots, T$:
05 If $i = t$ and $a_i = \top$:
06 $\mathsf{Broadcast}(\mathsf{dev}_A, i) \to m_A$
07 $m_A \to M$
08 $M \to \boldsymbol{m}_M$
09 $\mathsf{Receive}(\mathsf{dev}_A, i, \boldsymbol{m}_M)$
10 $M^{\mathsf{TS}(\cdot, i)}$
11 $M \to \mathsf{view}_M$
12 Output $(\mathsf{view}_M, \mathsf{dev}_A)$

(a) Claimed contact.

$\mathsf{Deny}_\Delta^{\mathsf{Sim}}$:

01 $\mathsf{Register}() \to \mathsf{dev}_A$
02 $S \to t \in \mathbb{N}$
03 $S \to \boldsymbol{a} \in \{\top, \bot\}^T$
04 For $i := 0, \ldots, T$:
05 If $i < t - \Delta$ or $t + \Delta < i$ and if $a_i = \top$:
06 $\mathsf{Broadcast}(\mathsf{dev}_A, i) \to m_A$
07 $m_A \to S$
08 $S \to \boldsymbol{m}_S$
09 $\mathsf{Receive}(\mathsf{dev}_A, i, \boldsymbol{m}_S)$
10 $S^{\mathsf{TS}(\cdot, i)}$
11 $S \to \mathsf{view}_S$
12 Output $(\mathsf{view}_S, \mathsf{dev}_A)$

(b) Simulated contact.

Figure 6: DU protocol Δ-contact-time deniability games.

5.1 Black-box Model

Theorem 1. *If a timestamping oracle exists, a decentralized unidirectional automated contact-tracing protocol cannot be correct, Δ-contact-time deniable, and Δ′-replay-secure for any $\Delta, \Delta' \in \mathbb{N}$ with $\Delta \geq \Delta'$ in the black-box model.*

Proof. Assume that the DU protocol is Δ-contact-time deniable and correct. Since it is correct, it must either be an *AB*-correct or *BA*-correct protocol. In the following, we will only consider *AB*-correct protocols as the proof for the other case is symmetrical.

We will construct an adversary M' winning the Δ-replay-security game with non-negligible probability. To this end, consider the Δ-contact-time deniability game with M_t and J_t (t to be set later) as defined in Figure 7.

M_t schedules A to be online at time t, waits for A's message m_A, timestamps it, and outputs m_A together with the timestamp τ as its view. The judge J_t verifies m_A's timestamp, registers an honest user B, and simulates B receiving m_A. Then, because we assume an *AB*-correct protocol, J_t checks that m_A caused an additional contact

M_t:

02 Output t
03 Output $\boldsymbol{a} \coloneqq (\bot, \bot, \ldots)^T$ except for $a_t \coloneqq \top$
// Wait until iteration $i = t$ in Figure 6a
// …
// Iteration $i = t$:
07 Store m_A
08 Output $m_M \coloneqq ()$
10 Query m_A to TS receiving $\tau \coloneqq \mathsf{TS}(m_A, t)$
11 Output $\mathsf{view}_M \coloneqq (m_A, t, \tau)$

(a) M_t. Note that the line numbers correspond to Figure 6a.

$J_t((m_A, t', \tau), \mathsf{dev}_A)$:

01 If $t \neq t'$, output $\bot$
02 If $\mathsf{VerifyTS}(m_A, t, \tau) = \bot$, output $\bot$
03 $\mathsf{Register}() \rightarrow \mathsf{dev}_B$
04 $\mathsf{Receive}(\mathsf{dev}_B, t, m_A)$
05 $\mathsf{Check}^{\mathsf{db}}(\mathsf{dev}_B) \rightarrow n$
06 $\mathsf{Report}^{\mathsf{db}}(\mathsf{dev}_A)$
07 If $\mathsf{Check}^{\mathsf{db}}(\mathsf{dev}_B) > n$, output $\top$ (real) and $\bot$ (simulated) otherwise

(b) Judge J_t.

Figure 7: M_t and J_t in the proof of Theorem 1.

by forcing A to report sick and checking whether B registered an additional contact[10].

Since the protocol is assumed to be deniable, for every M_t with $\Delta < t \leq T$, there exists a simulator S_t outputting a view_{S_t} such that J_t cannot distinguish between $(\mathsf{view}_M, \mathsf{dev}_A)$ and $(\mathsf{view}_{S_t}, \mathsf{dev}_A)$ except with negligible probability. We now use S_t to construct M' breaking Δ-replay-security.

(1) M' picks t such that $\Delta < t \leq T$.
(2) M' starts S_t and waits until S_t outputs t' and $\boldsymbol{a}$. If $t' \neq t$, M' aborts.
(3) M' modifies $\boldsymbol{a}$ so that $a_t \coloneqq \top$, and then starts the Δ-replay-security game by outputting t and $\boldsymbol{a}$.
(4) For the time steps $0 \leq i < t - \Delta$, M' proxies messages and $\mathsf{TS}(\cdot, i)$ queries back-and-forth between the replay-security game and S_t.
(5) For $t - \Delta \leq i < t$, M' only proxies $\mathsf{TS}(\cdot, i)$ queries.
(6) When $i = t$, M' waits for the query $\mathsf{TS}(m_A, t)$.
(7) Finally, while $i = t$ still, M' interacts with B in the replay-security game by outputting the message $\boldsymbol{m}_M = (m_A)$.

M' breaks Δ-replay-security with non-negligible probability as it only fails when J_t distinguishes the output of S_t from the output of M_t or when S_t forges a timestamp.

Assuming that J_t cannot distinguish, by the checks performed by J_t (Lines 1 to 2, Figure 7b), S_t must output $t' = t$ and view_S must be a valid timestamp. Additionally assuming that S_t does not forge this timestamp, it must query $\mathsf{TS}(m_A, t)$. As a consequence, M' does not abort in Item 2, simulates the interaction with A and TS correctly during Items 4 to 6, and extracts m_A from the query $\mathsf{TS}(m_A, t)$ in Item 6. Since the final checks performed by J_t (Lines 5 to 7, Figure 7b) are equal to the ones at the end of the replay-security game $\mathsf{Replay}(A, B)$ (Lines 15 to 17, Figure 5) and the protocol is correct by assumption, M' wins the Δ-replay-security game. So the protocol is not Δ-replay-secure and, by applying the contrapositive of Claim 1, also not Δ'-replay-secure for any $\Delta' < \Delta$. This completes the proof. □

We note that the proof assumes that A did not report as sick. In practice, if A had already reported as sick, a slightly modified version of J_t can simply skip forcing A to report and check whether any contacts are reported.

5.2 Gray-box Model

Theorem 1 only applies to *unidirectional* protocols. To extend the impossibility result to other forms of decentralized protocols (i.e., bidirectional or even interactive), we need to use a stronger model. One idea would be to use the white-box model which assumes a strong adversary with a lot of capabilities. Ideally, however, we use a model that makes the adversary *just* strong enough to yield an impossibility result. This makes the statement of the result stronger and helps us better understand why the impossibility occurs. These considerations lead us to the gray-box model that is situated between the black and white-box model.

Recall that the gray-box model allows the adversary to set the randomness when using their devices. Essentially, this means that the functions defined in Section 4.1 additionally take the randomness as a parameter. For example, $\mathsf{Broadcast}(\mathsf{dev}_A, t; \rho)$ where ρ is the randomness to be used. Honest users set ρ to the output of the device's random number generator, but a malicious user might input anything (e.g., all 0s).

In the following proof, the malicious user M will set the randomness to the one produced by a randomness beacon [26]. A randomness beacon produces randomness for some time t such that it is publicly verifiable that this randomness could *not* have been known before time t. In practice, a blockchain is not only a timestamping oracle but also acts as randomness beacon [9].

Theorem 2. *If a timestamping oracle and a randomness beacon exist, a decentralized automated contact-tracing protocol (even an interactive one) cannot be correct, Δ-contact-time deniable, and Δ'-replay-secure for any $\Delta, \Delta' \in \mathbb{N}$ with $\Delta \geq \Delta'$ in the gray-box model.*

Proof (Sketch). The proof is similar to the one of Theorem 1.

At time t, M registers a device B using Register and interacts with A; for this, it uses the randomness ρ output by the randomness beacon at time t. Then, as before, it timestamps any message it receives from A before passing it on to B. view_M contains timestamped messages as well as the randomness used by B.

Again, as before, the judge J verifies the timestamps which gives a sharp upper bound on the time of the interaction. Then, it verifies that the randomness ρ was produced by the beacon at time t which gives a matching lower bound. By setting the time and randomness in the gray-box model, J can faithfully replay the interaction between A and B. Finally, J checks whether reporting A causes a notification for B.

[10] For a BA-correct protocol, J_t would report B as sick and check whether A registered an additional contact.

Correctness and replay security of the protocol imply that Δ is the time between receiving ρ and the timestamp of m_A. □

6 DISCUSSION

The two impossibility results show that contact-time deniability cannot be achieved for a large class of protocols. The important question, however, is the impact on practical deployments of ACT protocols. Below we discuss some possibilities to overcome our no-go results.

6.1 Centralized and Hybrid Protocols

Both impossibility results only hold for decentralized protocols. Clearly, one might consider whether centralized and hybrid alternatives would be better. To this end, let us see why the proof only covers decentralized protocols.

In short, the design space of centralized/hybrid protocol is too large and intractable since the server can perform arbitrary computation and communication. For example, the ACT protocol could force the app to use a time mandated by the central server instead of the phone's settings. This violates the assumption in the proof that the judge can set the time arbitrarily. In practice, this specific countermeasure is probably too brittle (what happens if the phone, e.g., loses data connectivity?) and would negatively impact correctness.

In general, we believe that most natural centralized/hybrid protocol designs are still somewhat captured by our impossibility results. Furthermore, centralized systems only ensure sufficient privacy if one trusts the institutions running the servers [33]. If this is the case, we assume that deniability is less of an issue overall. Though this depends on the context and the precise nature of trust (e.g., device manufactures, federal government, etc.).

6.2 Diffie-Hellman-based Protocols

The DH-based protocols are not captured by Theorem 1 because they are bidirectional. In contrast to other bidirectional protocols (e.g., Challenge-Response), DH-based approach are still non-interactive. This makes them an attractive solution, however, they are not deniable in the gray-box model by Theorem 2. So, does the gray-box model capture realistic attacks?

Practicability of the Gray-box Model. The gray-box model requires control over the randomness used by the protocol. Compared to setting the time on a phone, setting the randomness is harder. On Apple's iOS, this seems particularly hard due to the locked-down nature of the platform. On Android, it might be possible but would require non-trivial modifications to the operating system. In practice, the easiest solution is probably re-implementing the protocol on open platforms such that time and randomness can be controlled.

One way to make the gray-box attacks impossible is utilizing the trusted platform modules and attestation techniques available on modern day smartphones. Then the ACT app can be reasonably sure that it is using properly generated randomness as re-implementing the protocol would be very challenging. This mitigation has already been suggested by Vaudenay [32] with respect to other attacks against privacy properties.

To summarize, gray-box attacks require more technical sophistication and premeditation than black-box attacks. So the black-box model is more relevant in practice.

Efficiency of Diffie-Hellman-based Protocols. Assuming that we are happy with black-box deniability, let us consider how efficient DH-based protocols are.

An important metric for users is battery life. Early on, BLE broadcasts were limited to 128 bits—too short for meaningful security for DH-based protocols. Nevertheless, Pronto-C2 [6] came up with a clever solution to the problem using a bulletin board. Luckily, this is not a problem anymore since newer smartphones support larger BLE broadcasts [34] that can easily fit group elements. Furthermore, sampling new public keys and computing the shared keys can be delayed until the phone is charging [21]. In summary, the DH-based protocols can be reasonably energy efficient.

Another metric is data usage which is important since many people have limited data plans. Protocols like DP3T use very little data because they only up/download keys. The keys constitute very concise descriptions of *all* messages sent by diagnosed users. In contrast, DH-based protocols up/download what essentially constitutes shared identifiers in the form of shared DH-keys, one for each contact. Since these depend on the randomness of two users, there is no hope of compressing them.

Decentralized DH-based protocols quickly become too expensive as they require users to download all newly reported shared keys. For example, if we conservatively assume 5 000 daily reports covering 14 days with 15 shared keys per report on average (prior work assumes 20 [21], 100 [11], or 200 [16]), then a user has to download roughly 1 million shared keys per day. The keys may be hashed to 128 bits to save space, but this still amounts to 16 MB daily and 0.5 GB monthly. Note that all these values are quite conservative estimates and the amount of data could be considerably larger in practice. So if this traffic is not zero-rated (i.e., free for users), users might be reluctant to install the ACT app.

In contrast, centralized DH-based protocols are more predictable because every user only needs to upload their own shared keys every day, using the numbers from above that would be just 15. In principle, DoS attacks are possible by artificially causing the victim's device to register a lot of contacts, i.e., shared keys. To summarize, data usage is an issue for DH-based protocols because it is not predictable.

6.3 Relaxing Correctness Requirements

The impossibility results assume perfect correctness. There have been proposals (e.g., DP3T Unlinkable [31]) that use a probabilistic reporting/checking process which will cause incorrect notifications (including false positives) to improve privacy. Adding noise reduces correctness, but increases privacy. In particular, it gives some deniability (which was not an explicit design goal of those proposals) as nothing definite can be deduced from observed behaviour.

We believe that adding noise is problematic. Intuitively, if the noise is large enough to matter in terms of privacy, then the protocol is too unreliable. Studies found that reliability of ACT protocols is important to prospective users [7], so this is not a good solution.

6.4 Fine-grained User Control

Instead of aiming for full contact-time deniability, it is also possible to offer users a choice in how much deniability they want. We discuss three solutions.

Partial Redactions. Hashomer [25] allows users to partially redact stored contacts. This is useful in cases such as Example 1 and might give users enough deniability in practice. Hashomer also allows users to exclude time-spans when reporting as sick, this is a nice privacy feature, but does not give any additional deniability.

For example, when considering DP3T as in Section 2, deleting a daily key SK_d redacts day d.[11] To achieve a finer granularity while still being reasonably efficient, a tree-style key-derivation may be used [25].

Mixed Protocols. The ACT app could run two protocols in parallel, e.g., a replay-insecure implementation of DP3T and Delayed Authentication. Users select whether they value replay-security (Delayed Authentication) or contact-time deniability (replay-insecure DP3T) more. Devices then only send *one* type of message (i.e., either Delayed Authentication or DP3T) while still receiving and processing *both* message types.

To increase security, the implementation could ensure that both types of messages are indistinguishable, i.e., making the message payloads of DP3T and Delayed Authentication look identical. So only when the sender reports as sick can receivers notice whether the sender used DP3T or Delayed Authentication.

By using Delayed Authentication as default setting, and only having a small subset of users who require strong privacy opting for DP3T, the protocol would remain mostly replay secure.

Passive Users. A recent work by Abtahi et al. [1] offers another protocol that gives users more control over their privacy. In short, their ACT protocol allows users to be either *active* or *passive*. Active users send and receive messages while passive users *only* receive messages. This ensures a high degree of privacy for passive users.

7 CONCLUSION AND FUTURE WORK

Our work shows that it is impossible for a decentralized ACT protocol to be correct, replay-secure and offer deniability with respect to contact time. The impossibility result holds for unidirectional protocols in the black-box model but can be extended to all decentralized protocols in a slightly stronger model. While these results are of theoretical nature, they also have practical implications.

First, the impossibility results explain some of the trade-offs taken by early ACT protocols designed at the height of the Covid-19 pandemic. Remarkably, existing proposals cover all sensible combinations (cf. Table 1)

Second, understanding *why* something is impossible in a theoretical, idealized model aids in finding practical circumventions. We have discussed several possibilities in Section 6 (e.g., allowing users to redact stored contacts) and hope to see more creative solutions that increase the privacy of users.

Lastly, our work is the first step in gaining a theoretically-founded understanding of the ACT protocol design space (but we are still far from fully understanding it). In the future, combining this with user studies (e.g., [7]) hopefully allows protocol designers to pinpoint which combination of properties (efficiency, security, and privacy) is most desired by users *and* practically realizable. Ideally, this leads to a protocol implementation with high rates of adoption.

[11] Instead of deleting the key, overwriting it with a random string makes the redaction undetectable—even against white-box adversaries.

ACKNOWLEDGMENTS

We thank Raluca-Georgia Diugan for her initial contributions and support afterward.

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/F85.

REFERENCES

[1] Azra Abtahi, Mathias Payer, and Amir Aminifar. 2024. DP-ACT: Decentralized Privacy-Preserving Asymmetric Digital Contact Tracing. , 330–342 pages. https://doi.org/10.56553/popets-2024-0019

[2] Nadeem Ahmed, Regio A. Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S. Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K. Jha. 2020. A Survey of COVID-19 Contact Tracing Apps. *IEEE Access* 8 (2020), 134577–134601. https://doi.org/10.1109/ACCESS.2020.3010226

[3] Fraunhofer AISEC. 2020. Pandemic Contact Tracing Apps: DP-3T, PEPP-PT NTK, and ROBERT from a Privacy Perspective. Cryptology ePrint Archive, Paper 2020/489. https://eprint.iacr.org/2020/489

[4] Apple and Google. 2020. Exposure Notification. https://www.google.com/covid19/exposurenotifications/

[5] Benedikt Auerbach, Suvradip Chakraborty, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, Michael Walter, and Michelle Yeo. 2021. Inverse-Sybil Attacks in Automated Contact Tracing. In *CT-RSA 2021 (LNCS, Vol. 12704)*, Kenneth G. Paterson (Ed.). Springer, Heidelberg, 399–421. https://doi.org/10.1007/978-3-030-75539-3_17

[6] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. 2021. Towards Defeating Mass Surveillance and SARS-CoV-2: The Pronto-C2 Fully Decentralized Automatic Contact Tracing System. https://doi.org/10.14722/coronadef.2021.23013

[7] Oshrat Ayalon, Dana Turjeman, and Elissa M. Redmiles. 2023. Exploring privacy and incentives considerations in adoption of COVID-19 contact tracing apps. In *Proceedings of the 32nd USENIX Conference on Security Symposium* (Anaheim, CA, USA) *(SEC '23)*. USENIX Association, USA, Article 30, 18 pages.

[8] Wasilij Beskorovajnov, Felix Dörre, Gunnar Hartung, Alexander Koch, Jörn Müller-Quade, and Thorsten Strufe. 2021. ConTra Corona: Contact Tracing against the Coronavirus by Bridging the Centralized-Decentralized Divide for Stronger Privacy. In *ASIACRYPT 2021, Part II (LNCS, Vol. 13091)*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, Heidelberg, 665–695. https://doi.org/10.1007/978-3-030-92075-3_23

[9] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. 2015. On Bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015. https://eprint.iacr.org/2015/1015.

[10] Xavier Bonnetain, Anne Canteaut, Véronique Cortier, Pierrick Gaudry, Lucca Hirschi, Steve Kremer, Stéphanie Lacour, Matthieu Lequesne, Gaëtan Leurent, Léo Perrin, et al. 2021. Le traçage anonyme, dangereux oxymore. *Le droit face au coronavirus* (2021), 468–480. English translation: https://risques-tracage.fr/docs/tracing-risks.pdf.

[11] Antoine Boutet, Claude Castelluccia, Mathieu Cunche, Cédric Lauradou, Vincent Roca, Adrien Baud, and Pierre-Guillaume Raverdy. 2022. Desire: Leveraging the Best of Centralized and Decentralized Contact Tracing Systems. *Digital Threats* 3, 3, Article 28 (March 2022), 20 pages. https://doi.org/10.1145/3480467

[12] Ran Canetti, Yael Tauman Kalai, Anna Lysyanskaya, Ronald L. Rivest, Adi Shamir, Emily Shen, Ari Trachtenberg, Mayank Varia, and Daniel J. Weitzner. 2020. Privacy-Preserving Automated Exposure Notification. Cryptology ePrint Archive, Report 2020/863. https://eprint.iacr.org/2020/863.

[13] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. 2020. ROBERT: ROBust and privacy-presERving proximity Tracing. (May 2020). https://inria.hal.science/hal-02611265

[14] Fabrizio Cicala, Weicheng Wang, Tianhao Wang, Ninghui Li, Elisa Bertino, Faming Liang, and Yang Yang. 2021. PURE: A Framework for Analyzing Proximity-based Contact Tracing Protocols. *ACM Comput. Surv.* 55, 1, Article 3 (Nov. 2021), 36 pages. https://doi.org/10.1145/3485131

[15] Noel Danz, Oliver Derwisch, Anja Lehmann, Wenzel Puenter, Marvin Stolle, and Joshua Ziemann. 2020. Provable Security Analysis of Decentralized Cryptographic Contact Tracing. Cryptology ePrint Archive, Paper 2020/1309. https://eprint.iacr.org/2020/1309

[16] Giuseppe Garofalo, Tim Van hamme, Davy Preuveneers, Wouter Joosen, Aysajan Abidin, and Mustafa A. Mustafa. 2022. PIVOT: Private and Effective Contact Tracing. *IEEE Internet of Things Journal* 9, 22 (2022), 22466–22489. https://doi.org/10.1109/JIOT.2021.3138694

[17] Scott Griffy and Anna Lysyanskaya. 2023. PACIFIC: Privacy-preserving automated contact tracing scheme featuring integrity against cloning. Cryptology ePrint Archive, Report 2023/371. https://eprint.iacr.org/2023/371.

[18] Yaron Gvili. 2020. Security Analysis of the COVID-19 Contact Tracing Specifications by Apple Inc. and Google Inc. Cryptology ePrint Archive, Paper 2020/428. https://eprint.iacr.org/2020/428

[19] Vincenzo Iovino, Serge Vaudenay, and Martin Vuagnoux. 2021. On the Effectiveness of Time Travel to Inject COVID-19 Alerts. In *Topics in Cryptology – CT-RSA 2021*, Kenneth G. Paterson (Ed.). Springer International Publishing, Cham, 422–443.

[20] Kevin Morio, Ilkan Esiyok, Dennis Jackson, and Robert Künnemann. 2023. Automated Security Analysis of Exposure Notification Systems. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 6593–6610. https://www.usenix.org/conference/usenixsecurity23/presentation/morio

[21] Thien Duc Nguyen, Markus Miettinen, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Ivan Visconti. 2022. Digital Contact Tracing Solutions: Promises, Pitfalls and Challenges. *IEEE Transactions on Emerging Topics in Computing* (2022), 1–12. https://doi.org/10.1109/TETC.2022.3216473

[22] Sangchul Park, Gina J. Choi, and Haksoo Ko. 2021. Privacy in the Time of COVID-19: Divergent Paths for Contact Tracing and Route-Disclosure Mechanisms in South Korea. *IEEE Security & Privacy* 19, 3 (2021), 51–56. https://doi.org/10.1109/MSEC.2021.3066024

[23] PePP-PT. 2020. Data Protection and Information Security Architecture. https://github.com/pepp-pt/pepp-pt-documentation/blob/master/10-data-protection/PEPP-PT-data-protection-information-security-architecture-Germany.pdf

[24] Krzysztof Pietrzak. 2020. Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing. In *INDOCRYPT 2020 (LNCS, Vol. 12578)*, Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran (Eds.). Springer, Heidelberg, 3–15. https://doi.org/10.1007/978-3-030-65277-7_1

[25] Benny Pinkas and Eyal Ronen. 2021. Hashomer – Privacy-Preserving Bluetooth Based Contact Tracing Scheme for Hamagen. In *CoronaDef Workshop 2021*. https://doi.org/10.14722/coronadef.2021.23011

[26] Michael O. Rabin. 1983. Transaction protection by beacons. *J. Comput. System Sci.* 27, 2 (1983), 256–267. https://doi.org/10.1016/0022-0000(83)90042-9

[27] Leonie Reichert, Samuel Brack, and BjÖRN Scheuermann. 2021. A Survey of Automatic Contact Tracing Approaches Using Bluetooth Low Energy. *ACM Trans. Comput. Healthcare* 2, 2, Article 18 (March 2021), 33 pages. https://doi.org/10.1145/3444847

[28] Ronald Rivest, M. Curran Schiefelbein, Marc A. Zissman, Jason Bay, Edouard Bugnion, Jill Finnerty, Ilaria Liccardi, Brad Nelson, Adam S. Norige, Emily H. Shen, Jenny Wanger, Raphael Yahalom, Jesslyn D. Alekseyev, Chad Brubaker, Luca Ferretti, Charlie Ishikawa, Mariana Raykova, Brendan Schlaman, Robert X. Schwartz, Emma Sudduth, and Stefano Tessaro. 2023. Automated Exposure Notification for COVID-19. https://dspace.mit.edu/handle/1721.1/148149

[29] Pietro Tedeschi, Spiridon Bakiras, and Roberto Di Pietro. 2023. SpreadMeNot: A Provably Secure and Privacy-Preserving Contact Tracing Protocol. *IEEE Transactions on Dependable and Secure Computing* 20, 3 (2023), 2500–2515. https://doi.org/10.1109/TDSC.2022.3186153

[30] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. 2020. Epione: Lightweight Contact Tracing with Strong Privacy. *IEEE Data Eng. Bull.* 43, 2 (2020), 95–107. http://sites.computer.org/debull/A20june/p95.pdf

[31] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Čapkun, David Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel Smart, Aysajan Abidin, Seda Gürses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. 2020. Decentralized Privacy-Preserving Proximity Tracing. arXiv:2005.12273 [cs.CR]

[32] Serge Vaudenay. 2020. Analysis of DP3T. Cryptology ePrint Archive, Report 2020/399. https://eprint.iacr.org/2020/399.

[33] Serge Vaudenay. 2020. Centralized or Decentralized? The Contact Tracing Dilemma. Cryptology ePrint Archive, Report 2020/531. https://eprint.iacr.org/2020/531.

[34] Torbjørn Øvrebekk. 2022. Bluetooth 5 Advertising Extensions. https://blog.nordicsemi.com/getconnected/bluetooth-5-advertising-extensions