Choo-choo: A Trustworthy Contact Tracing System

Chendi Wu
Stanford University

Jiani Wang
Stanford University

Angela Montemayor
Stanford University

Abstract

The COVID-19 pandemic reinforced the need for publicly available, robust, and secure contact tracing systems. These systems provide vital information to both users and public health officials, which helps slow down the spread of infectious diseases. However, the rise in prominence of these systems has also spurred necessary conversation surrounding the privacy of users, verifiability of results, and the danger of false sources of panic. We present Choo-choo, a trustworthy contact tracing system that aims to mitigate many shortcomings of modern contact tracing systems.

1 Introduction

COVID-19 continues to be the largest public health crisis in modern history. At the start of this pandemic, it became evident that the world was vastly underprepared for a global phenomenon of this magnitude. As scientists and medical experts explored methods of mitigation before vaccines became widely available, contact tracing technology came to the forefront as a powerful tool for users to be informed of potential exposures and for public health officials to detect large outbreaks. That said, this technology was met with substantial skepticism and concern amongst individuals aiming to safeguard their privacy and avoid false sources of panic [2]. Technologists were met with the challenge of balancing personal privacy and accuracy so individuals would be incentivized to use the contact tracing system for the greater good.

Several contact tracing systems came to fruition during this time, each with its own strengths and weaknesses. We outline some of the most notable protocols of these systems in Section 2. As we explored these systems and what they offered the public, we felt as though there was opportunity for improvement in both areas of privacy and accuracy. Specifically, we focused our efforts in improving the verification of test results and studying the potential implications of negative test results. We therefore present Choo-choo, a trustworthy contact tracing system. Choo-choo focuses on giving users a risk level, rather than a binary offering of "you have/have not been exposed." We do so by factoring in negative test results collected by others who were exposed in the same event the user was. Additionally, Choo-choo further verifies the validity of test results using a Public Key Infrastructure, so a valid test result is required to report a positive or negative result to the system. Lastly, Choo-choo aims to protect against some scenarios of collusion amongst malicious users by employing cryptography techniques.

This paper provides a survey of current contact-tracing protocols in Section 2, which also details aspects of the Apple-Google Protocol from which we took significant inspiration. In Section 3 we outline our security goals and threat model. Section 4 details the system design decisions we made, and Section 5 provides some basic experimentation results that verify the validity of the approach. Lastly, Section 7 concludes.

2 Background

In this section, we detail prior work in the area of contact tracing systems.

2.1 Survey of contact tracing protocols

Contact tracing protocols take many factors into consideration. For instance, proximity measurement, access control, data storage, and encryption all come into play when designing a contact tracing protocol [2]. Furthermore, some protocols are designed to be peer-to-peer enabled, whereas others operate with the presence of a central server. The following are some notable contact tracing protocols paired with some of their pros and cons [2]:

QUEST - This centralized protocol uses WiFi. It only accepts reports from authorities, and has relatively low battery consumption. Data is stored both on devices and server.

BlueTrace - This centralized protocol uses Bluetooth. It only accepts reports from authorities, and has relatively low battery consumption. Data is stored only on devices.

Recover - This centralized protocol uses GPS and Bluetooth. It only accepts reports from authorities, and has relatively low battery consumption. Data is stored only on server.

Each of these protocols has tradeoffs in the areas of privacy, efficiency, and security. They have varying levels of popularity and usage.

2.2 Apple-Google Protocol

We modeled much of Choo-choo's design after the Apple-Google Protocol [1]. We specifically chose this protocol due to its robustness and privacy-centric tactics.

Particularly, we were drawn to the fact that the Apple-Google Protocol does not store private user data in the clear but rather requires an anonymous identifier to determine exposures [1]. To illustrate the importance of this fact, consider if we were to naively construct a contact-tracing system. Perhaps the easiest way of doing this would be to have a high-capacity central server. Users would consistently ping this server at regular intervals, say every 5 minutes or so, with the user's id and location. When a user reports a positive case to the server, the server would perform some cross-section calculation to determine which users were exposed by the positive-case individual and inform those users accordingly. Ignoring the absurd amount of data the server would have to keep track of in this approach, the idea that a central single server would contain such highly sensitive data is very concerning. If an attacker managed to hack into this server, they would be able to track individuals and infer even further personal information like home locations. The Apple-Google Protocol addresses this problem head-on and uses a methodology that does not track location in this manner nor tie a user's identity to the data it generates [1]. This approach is further explained in Section 4.

Additionally, we chose the Apple-Google protocol as our main model since the contact tracing process occurs locally on a user's phone [1]. This allows for a more even distribution of power in the system; if the server did the contact tracing process, and was hacked, an attacker could determine a plethora of information. In addition, with the user performing this calculation locally on their phone, we avoid the infrastructure and risk of having to transfer this information over a network.

3 Goals and Problem Statement

In this section, we present the threat model and security goals of the Choo-choo system.

3.1 Threat Model

Clients in our system are semi-trusted: all security properties of our system holds in the face of an individual malicious client, but we acknowledge potential attack scenarios when malicious clients collude. We will address the impact of malicious collusion in Section 3.2.

The central server is trusted for availability. Failures will compromise the system's liveness, but the privacy of the clients will always be preserved. We consider fault-recovery an orthogonal problem to our system goals. To protect against benign failures, the server could implement a state-replication protocol on top of our central server.

The testing authority server is required to be completely trusted. We require this server to always abide to our protocol to guarantee the integrity of the system.

3.2 Security Goals

We describe Choo-choo as a *trustworthy* contact tracing system. We define trustworthy to be a state when the following properties of *correctness*, *privacy* and *integrity* are satisfied.

Correctness. The scheme is correct when all servers execute the protocol faithfully, and if both of the following conditions are satisfied:

- all verifiable test results and the correct corresponding client devices are binding, and
- the positive and negative cases based on which a client's risk level is computed match the positive and negative reports from the client's contacts.

Since we rely on all servers for availability, correctness of the system need only hold when all servers are running the protocol correctly.

Privacy. The system preserves user privacy in the following aspects:

- The servers cannot infer the users' geographical locations or the users' contacts from the information the server receives.
- The users cannot discover the identities of the reporting users.

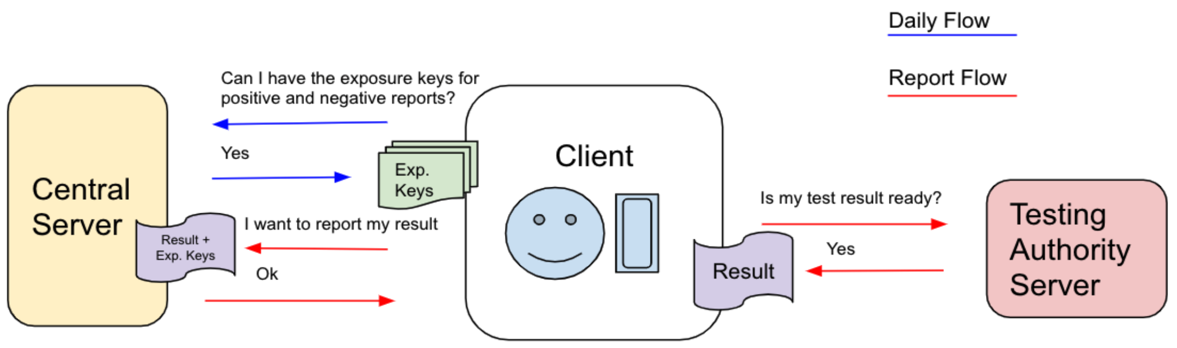

Figure 1: Choo-choo daily and report flows.

- The users can discover where they are in contact with the reported cases, but not the complete trajectory of the reporting users.

Admittedly, the user will be able to discover the identity of a reporting user if the reporting user is the only person that the user has been in contact with at a specific time and location. We don't consider this a privacy concern. If the reporting user wants to hide their test information from the other user, they can simply don't report; otherwise if the reporting user wants people they have been in contact with to be notified, even without our system they would want to notify the single person they've been in contact with.

Integrity. The system has integrity even in the face of attackers that:

- report dishonest results (including faking non-existent results and modifying their own results) or
- intercept other users' testing results from a testing authority server in order to impersonate an honest user.

Liveness is naturally preserved in the face of these attack scenarios as the system detects this malicious behavior and discards the confounding reports.

It should be noted that in order to preserve privacy, no information related to a person's real identity, such as contact information, personal identification number etc., is stored or used anywhere in the system. In other words, the current system identifies a user by the users' exposure keys (explained in more details in Section 4.2) which are non-binding to the user's real identity.

This results in a caveat: our system does not protect against all collusion among malicious users. The following attack may undermine the correctness of our system: If malicious user Alice tests positive and malicious user Bob tests negative, they can exchange both their exposure keys and the signed testing result they receive independently from a testing authority server, while still passing our verification scheme. The impact of this is no worse than the impact of a false positive report in the current Apple-Google contact tracing protocol, and we consider it an acceptable trade-off with the privacy guarantees of our system. Future work may consider using zero knowledge proofs to provably bind a user's registered real identity information with exposure keys while not revealing the real identity information.

4 System Design

In this section, we describe the Choo-choo system design and discuss implementation details.

4.1 Architecture

Choo-choo has three main parties: the client, the central server, and the authentication server. Figure 1 shows the basic channels of communication between these entities. Note that we use *client* and *app* interchangeably in the sections below.

On a daily basis, the client generates an *exposure key* that is unique to them. The user maintains a running set of their own exposure keys from the last 5 days. From this exposure key, we generate a *token* every 10 minutes. This is done using cryptographic primitives, which we discuss in greater detail in the next subsection. As users go about their daily lives, tokens are exchanged via Bluetooth when users come into close proximity. These are stored locally on user devices. We maintain a running set of tokens collected in the last 5 days, sorted by day. With this in mind, we present two flows: the **report flow** and the **daily flow**.

The **report flow** begins with user action outside of the system. That is, a user needs to register to take a test at a Choo-choo compatible testing authority. On the day the user sends the test specimen, the Choo-choo app would send the user's exposure keys in the past three days (including the day of the test) to the testing authority server. After the user has completed the test, the user's Choo-choo app client polls daily for the result from the testing authority server and sends the exposure key of that day. For testing purposes, we have developed a simple and lightweight testing authority server. In a real-world setting, the testing authority server would be a hospital or clinic's own server, which would be set up to communicate with the Choo-choo client. Once the result is available, the Choo-choo app receives the result, and the user determines whether they have tested positive or negative. The user has the option to report **either** result. If the user does so, the test result, alongside the user's exposure keys from the last 5 days, are sent to the central server.

Once the keys and result have been received, the server performs a cryptographic validity scheme to ensure the validity of the digital signature on the result. It does so by using the testing authority's publicly available public key. More detail on this cryptography is detailed in the next subsection. After the result has been verified, the server stores each of the user's exposure keys in files corresponding to the day the exposure key was generated. Every day has two associated files: one for positive reports and one for negative reports.

The **daily flow** consists of the client polling the central server for exposure keys corresponding to positive and negative reports filed for the day before. This flow is how a user determines if they have been exposed, so that the risk level can be adjusted accordingly. Once the client receives the batch of exposure keys, it again performs a set of cryptographic functions to convert those keys back into tokens (again, refer to the next subsection for more details). If any of the tokens corresponding to positive reports match any of the tokens the client collected for the corresponding day, an exposure has been detected. The risk level is adjusted to convey to the user that their risk has increased. If any of the tokens corresponding to negative reports match any of the tokens the client collected for the corresponding day, and an exposure was detected during that same event, the risk level is adjusted to convey to the user that their risk is still significant, but not the highest it can be. In other words, negative reports corresponding to exposure events is meant to convey to the user that another user who was exposed at the same event has since tested negative, so perhaps the disease was not spread.

4.2 Cryptography Flow

The system relies on cryptographic primitives to generate reproducible tokens that enable clients to "rediscover" their contacts with the reporting users and to provably bind a user's test result to the user's "live" exposure keys. We will analyze in this section how our composition of primitives work to satisfy our security goals.

Rediscover Contacts. The ability to rediscover contacts on clients' local devices is based on the reproducible token generation process (Algorithm 1).

We followed the Apple-Google protocol's token generation process, starting from generating a 16-byte *exposure key* for each client device using a cryptographic random number generator on a daily basis. A *token key* for the same day is derived using HKDF, a key derivation function based on HMAC message authentication code, in preparation for generating tokens to be exchanged via Bluetooth. We used HKDF on top of HMAC-SHA256 which is robust against dictionary and brute force attacks. Then, a *token* is the ciphertext from encrypting time interval metadata using the token key and is generated per 10 minutes and constantly exchanged with nearby client devices through Bluetooth payload.

The series of cryptographic primitives incorporates sufficient entropy such that the tokens cannot be correlated to the exposure keys. The tokens are also collision resistant, meaning that it is proven hard to generate two identical tokens from different exposure keys. Hence, a token can be considered as a unique identifier for one party during a 10-min contact.

Each day, the client polls the central server and gets back the "live" exposure keys of reporting users. We define "live" exposure keys to be exposure keys over the last 5 days, determined by the active transmission period of COVID-19. After running HKDF to reproduce the token key corresponding to each exposure key, the client can further regenerate the tokens related to the exposure key (and thus the reporting user) by calling the AES function on all time intervals over the past 5 days. A set intersection is then performed to count how much overlap there is between the client's locally stored tokens and the regenerated tokens. The intersection set identifies "rediscovers") the client's contact with the reporting user and we deduce more contextual information about the contact events from locally stored information and compute the user's risk level (explained in Section 4.3).

Generate Verifiable Test Results. We engage the testing authority to generate test results that are verifiable, tamper resistant and binding to the test-takers' exposure keys, which are the only identifiers of users in our system. This approach allows our system to be more trustworthy than the Apple-Google protocol in that our scheme is able to prevent the majority of false positive reports.

The scheme (Algorithm 2) takes advantage of existing digital signature schemes so that only the testing authority can generate signatures that are verifiable and will be accepted by the system. Each testing authority holds a keypair (sk_{ta}, pk_{ta}).

Suppose the user takes the test on day i, sends exposure keys between day $i-2$ and day i when they start the test and continuously sends exposure keys daily until they receive the test result on day $i+2$. To generate the verifiable test result, signing is performed twice. The first time, the testing authority uses sk_{ta} to sign a concatenation of all exposure keys it received from the user. The result signature is used as a sequence number (seq_k) for this test event. While digital signature schemes are not pseudorandom and leak distribution information about the signer, the signer (i.e. testing authority) identity and reporting exposure keys are all public in our system so we are not leaking additional information to a potential attacker. The second time, the testing authority appends its id, used later in verification to retrieve the testing authority's public key, seq_k, and test result, signs the appended message and appends the signature to make the complete verifiable test result.

Verify Test Results. When a user reports their test results, the user sends all live exposure keys along with the verifiable test result (VTR) they receive from the testing authority server.

The central server just follows the existing digital signature verification scheme to verify test results and filter maliciously-crafted ones. Similar to the signing process, verification is performed twice: the first time it verifies the sequence number in the VTR is a valid signature of the sequence of exposure keys; the second time it verifies the testing authority's signature on the entire test result so that the test result body cannot be mutated. All user reports that don't pass the verification tests are discarded.

This scheme is also naturally robust against replay attacks. Only "live" exposure keys can generate effective tokens that users receive and store, and old test results, though with a valid second signature, doesn't contain a valid sequence number that is binding to the "live" exposure keys.

Algorithm 1 Generate Tokens

```
$ExpKey_i \leftarrow CRNG(16)$
$TokenKey_i \leftarrow HKDF(ExpKey_i, salt, \text{“}RPIINFO\text{”}, 16)$
$Metadata_j \leftarrow \text{“}RPIINFO\text{”}||Interval_j||Padding$
$Token_{i,j} \leftarrow AES_{128}(TokenKey_i, Metadata_j)$
```

Algorithm 2 Generate Signed Test Result

```
$ExpKeys \leftarrow ExpKey_{i-2}||...||ExpKey_i||...||ExpKey_{i+2}$
$Seq_i \leftarrow Sign(sk_{ta}, ExpKeys)$
$Msg \leftarrow ID_{ta}||Seq_i||Result_i$
$Sig \leftarrow Sign(sk_{ta}, Msg)$
$SignedTestResult = Msg||Sig$
```

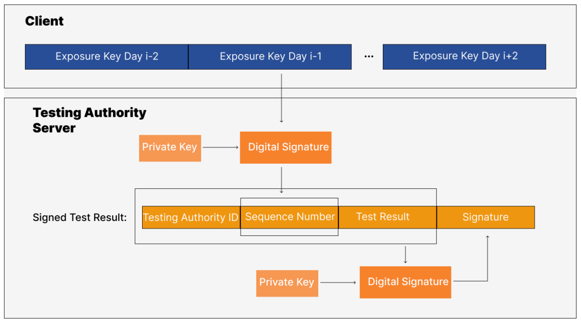

Figure 2: Signed Test Result

4.3 Risk Level Evaluation

Choo-choo provides risk level evaluation based on the exposures it determines. To evaluate the risk level of exposure, we consider 4 factors: number of positive tokens, number of negative tokens, GPS location of local device, Bluetooth Received Signal Strength Indicator (RSSI).

Through the daily flow, the client will receive exposure keys corresponding to both negative and positive reports from the central server. After regenerating the tokens and comparing with the locally stored tokens, the client can determine how many positive cases it has come into contact with. Our system also makes use of the exposure keys corresponding to negative reports. If we determine an exposure to a positive case, but other users report negative in the same exposure instance, we can have more confidence that our risk of having contracted COVID is lower.

For each client, when it exchanges cryptography tokens with nearby clients, it also collects its GPS location and peer device's RSSI information. When it receives a cryptography token from a peer, the client records its current GPS location. The GPS location is used to notify users the high risk locations that they have contact with another positive case. The user could recall whether they wear a mask in that location and the distance between themselves and other people, then test accordingly. The GPS information also helps decide the co-occurrence of positive and negative cases in same physical location. For Bluetooth RSSI, it is the strength of the beacon's signal as seen on the receiving device. The RSSI signal strength depends on distance and broadcasting power value. With this RSSI value, we can infer the distance between the current client and the peer. If we receive several positive reports with strong Bluetooth signal, we can infer that we have had close contact with those positive cases and the risk of getting COVID is relatively high. If there are interference between two clients, it will also reflected by the RSSI value.

For our Choo-choo implementation, we use a naive linear model to evaluate the risk score, and consider the effectiveness of distance inferred by RSSI. The algorithm is shown in “Algorithm 3”. The basic assumption of our risk score model is that 5 negative cases around a positive case is sufficient to indicate the virus is no longer contagious. Some parameters in our risk score model are derived from [3].

Algorithm 3 Calculate Risk Score s

```
Require: $0 \leq s \leq 100$, RSSI, positive and negtive report
  score← 0
  for Positive Case Report do
    dist $\leftarrow \alpha * 10^{-RSSI/4}$
    if dist $< 1$ then
      score += $20 * (1 + 12.8\%)$
    else
      score += $20 * (1 + 2.6\%)$
    end if
  end for
  for Negative Case Report do
    if Same GPS location with any positive case then
      dist $\leftarrow \alpha * 10^{-RSSI/4}$
      if dist $< 1$ then
        score -= $4 * (1 + 12.8\%)$
      else
        score -= $4 * (1 + 2.6\%)$
      end if
    end if
  end for
```

Our current risk score model is simple and heuristic. However, we believe Choo-choo’s data can be used in a more powerful way when paired with medical domain knowledge. For example, an epidemiologist could propose a more professional model to predict the risk level based on this data.

5 Experiments

We conducted experiments using several devices to illustrate Choo-choo’s functionality. We built and ran our client on iOS devices, specifically iPhone and iPad. The testing authority server and central server ran on AWS instances. To test the whole system with multiple clients, we employed several iOS devices in the same location and ran the Choo-choo system on them.

We also conducted end-to-end experiments to show the effectiveness of our cryptography flows and risk score model. Table 1 illustrates the risk score trends between 3 clients, if one or two of them report positive test result. We keep the distance the same for these two clients and read the risk score reported by Choo-choo.

To better understand how negative report cases influence Choo-choo’s risk score, we mimic a typical scenario

Test Situation	RSSI	Positive Cases	Risk Score
In same room, distance 1m	-51	1	**26**
In same room, distance 1m	-51	2	**51**
Through glass wall, distance 1m	-65	1	20
Through hard wall, distance 1m	-72	1	20

Table 1: Testing different physical layouts

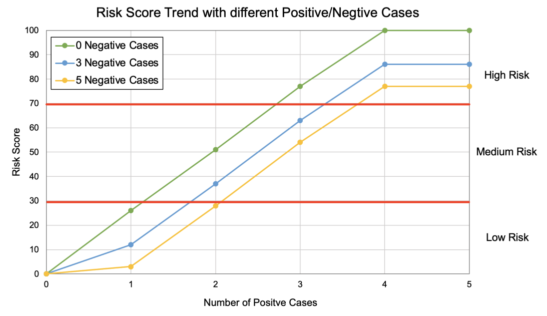

Figure 3: Risk Score trends

to show how the system works. We put 11 devices together (the distance between each devices being less than 1 meter) and read the risk score reported by Choo-choo. We tested 3 different scenarios and report the first device’s risk score: There are 0, 3, 5 clients reporting they are negative as seen in Figure 3. If more people report they tested negative, other clients might drop from “high risk” to “medium risk”. This change indicates that the client has relatively low risk to be infected by COVID.

6 Conclusions

In this paper, we proposed and implemented *Choo-choo*: a trustworthy and practical contact tracing system that factors in negative test results and exploits cryptographic primitives to prevent false positive reports.

We employ a reliable distributed protocol that involves three kinds of parties and supports multiple clients, modeled after the Apple-Google protocol. We defined our threat model and proposed our security goals to achieve trustworthiness: Correctness, Privacy, and Integrity. We further illustrate our cryptography flow that realizes our security goals, including the token generation and the server-side verification process of reported results. Finally, we evaluate the effectiveness and reliability of Choo-choo. Our experiments show that our system provides a reliable indicator of exposure risk and also supports a large number of clients.

References

[1] Apple-Google privacy-preserving contact tracing. https://covid19.apple.com/contacttracing.

[2] CHOWDHURY, M. J. M., FERDOUS, M. S., BISWAS, K., CHOWDHURY, N., AND MUTHUKKUMARASAMY, V. Covid-19 contact tracing: Challenges and future directions. *IEEE Access 8* (2020), 225703–225729.

[3] CHU, D. K., AKL, E. A., DUDA, S., SOLO, K., YAACOUB, S., SCHÜNEMANN, H. J., EL-HARAKEH, A., BOGNANNI, A., LOTFI, T., LOEB, M., ET AL. Physical distancing, face masks, and eye protection to prevent person-to-person transmission of sars-cov-2 and covid-19: a systematic review and meta-analysis. *The lancet 395*, 10242 (2020), 1973–1987.