BlindSignedID: Mitigating Denial-of-Service Attacks on Digital Contact Tracing

Bo-Rong Chen and Yih-Chun Hu
University of Illinois at Urbana-Champaign
{borongc2, yihchun}@illinois.edu

Abstract

Due to the recent outbreak of COVID-19, many governments suspended outdoor activities and imposed social distancing policies to prevent the transmission of SARS-CoV-2. These measures have had severe impact on the economy and peoples’ daily lives. An alternative to widespread lockdowns is effective contact tracing during an outbreak’s early stage. However, mathematical models (e.g., [24]) suggest that epidemic control for SARS-CoV-2 transmission with manual contact tracing is implausible. To reduce the effort of contact tracing, many digital contact tracing projects (e.g., PEPP-PT [4], DP-3T [33], TCN [8], BlueTrace [13], Google/Apple Exposure Notification [4], and East/West Coast PACT [6, 17]) are being developed to supplement manual contact tracing. However, digital contact tracing has drawn scrutiny from privacy advocates, since governments or other parties may attempt to use contact tracing protocols for mass surveillance. As a result, many digital contact tracing projects build privacy-preserving mechanisms to limit the amount of privacy-sensitive information leaked by the protocol. In this paper, we examine how these architectures resist certain classes of attacks, specifically DoS attacks, and present BlindSignedIDs, a privacy-preserving digital contact tracing mechanism, which are verifiable ephemeral identifiers to limit the effectiveness of MAC-compliant DoS attacks. In our evaluations, we showed BlindSignedID can effectively deny bogus EphIDs, mitigating DoS attacks on the local storage beyond 90% of stored EphIDs. Our example DoS attacks showed that using 4 attackers can cause the gigabyte level DoS attacks within normal working hours and days.

Keywords: Denial-of-Service Attacks, Privacy, Digital Contact Tracing, COVID-19

1 Introduction

In light of the success of several countries’ SARS-CoV-2 containment strategies based on aggressive contact tracing, several researchers have developed approaches to digital contact tracing [4, 6, 7, 8, 13, 17, 34]. Because of the privacy concerns inherent with disclosing user location traces, several proposals include mechanisms for privacy-preserving proximity tracing. These mechanisms make use of a limited-time identity to reduce an adversary’s ability to track that device from broadcast to broadcast; these identities are variously called ephemeral ID [34], Temporary ID [13], and Rolling Proximity ID [4]; in this paper, we will use the term ephemeral ID or EphID. To reduce the amount of bandwidth needed to disseminate positive identities, these ephemeral IDs are generated from a master secret in a one-way manner, such that the master secret is sufficient to derive all relevant ephemeral IDs, but that several ephemeral IDs leak no information about other associated ephemeral IDs.

In existing designs, a user receiving an ephemeral ID can neither associate that ephemeral ID with other ephemeral IDs from the same contact, nor can the user verify the validity of an ephemeral ID that it receives; as a result, a user must store all of the ephemeral IDs that it encounters across the maximum incubation period (widely set at 14 days for SARS-CoV-2). Furthermore, storing these ephemeral IDs in the cloud faces significant privacy challenges; if a large fraction of users trust a single cloud storage provider to store unencrypted ephemeral
this paper, we propose a design for BlindSignedIDs that cloud storage provider would learn extensive privacy-compromising information about a user’s social structure. As a result, existing schemes [4, 6, 8, 17, 34] largely assume that ephemeral IDs are stored on the user’s own device until the incubation period has expired. However, because of the limited storage available on commonly-used mobile devices, an attacker can send a large number of bogus ephemeral IDs, overwhelming the storage of all but high-end devices. We call this attack the DoS attacks. In fact, the existing EphID-based designs [4, 6, 8, 17, 34] are widely adapted by many countries [5, 7, 10, 13], which occupy approximately 12.5% of the world population, thus making these countries [3, 5, 7, 10, 13].

The attacker is able to modify the BLE protocol such that her device can advertise BLE beacon packets with secret-day seeds), an attacker with a high-power Bluetooth antenna can stream newly-generated ephemeral IDs to set the context for our attacks and our approaches. DP-3T generates ephemeral IDs using:

$$EphID_1 || \ldots || EphID_n = PRG(PR_F(SK_i, "broadcast key"))$$ (1)

where EphID_i is a 16-byte ephemeral ID, SK_i is secret day seed, PRF is a pseudo-random function, and PRG is a pseudo-random generator. To further reduce the amount of information disseminated for a positive report, DP-3T generates the next secret day seed SK_i by hashing the previous secret day seed SK_{i-1} with a one-way hash function, so that the dissemination of one secret day seed allows contact tracing for all subsequent days.

Each user generates 288 EphIDs per day, randomly reorders them, and uses Bluetooth Low Energy (BLE) to broadcast each EphID for one epoch (5 minutes). A device receiving such a beacon stores the EphID, signal strength information (such as RSSI), and the date on which the beacon was received, for a total of 36 bytes per EphID for the low-cost design and 52 bytes for the unlinkable design [34]. A receiving device must store these records for 14 days. When a user tests positive, her secret day seeds from the infectious period will be uploaded to the backend, so that other users can fetch seeds and compare with their local records to determine whether they have been in contact with this particular user.

3 Problem Environment and Statement

3.1 Existing Contact Tracing Design

In this section, we describe DP-3T [34] at a high level, to set the context for our attacks and our approaches. DP-3T generates ephemeral IDs through the use of blind signatures. The rest of this paper is organized as follows: in Section 2, we make system assumptions and describe the attacker model. In Section 3 we describe the security issues regarding current designs of digital contact tracing. Section 4 introduces our approach in detail. We provide an analysis of BlindSignedID in Section 5 and 6. Section 7 demonstrates our example DoS attacks and BlindSignedID can effectively reduce DoS attacks. Section 8 reviews current projects. We make conclusions in Section 9.

2 System Assumptions and Threat Model

2.1 System Assumptions

Each received EphID is kept in the user’s local storage for at least 14 days. The stored EphID includes additional information (e.g., encountering time and duration), which consumes fixed-byte spaces in the local storage. Moreover, the user stores every received EphID regardless of any encountering duration for recording and checking exposure. Finally, the mobile devices support Bluetooth Random Private Address [1], where each received EphID cannot be linked to the real Bluetooth device address.

2.2 Threat Model

The attacker is able to modify the BLE protocol such that her device can advertise BLE beacon packets with the minimum interval. Namely, the transmission interval is not bounded by the underlying BLE stack (e.g., maximum and minimum intervals in BlueZ [2]). She specifies the standard BLE flags with arbitrary content (i.e., spoofing EphID) such that other users will receive and keep the packets in the local storage. Moreover, the attacker uses a powerful BLE antenna for larger transmission ranges.
limited only by bandwidth. BLE has bandwidth of 1–2 Mbps, so any receiver within transmission range of an attacker will need to store (based on the overhead specified in the DP-3T paper) 1–2 GB per hour or 8–16 GB for an 8-hour day. Even when considering the possibility of compression or elimination of certain data, the incompressible EphID alone will amount to between 450 MB and 900 MB per hour, or 3.6 GB to 7.2 GB across an 8-hour working day. Since many phones do not have several gigabytes of available storage, such an attacker can cause the victims phone to run out of space. We believe that the lack of authentication of ephemeral IDs makes many privacy-preserving contact tracing approaches [4] vulnerable to DoS attacks.

3.3 Out-of-Scope Problems

Our approaches are designed for the attacks listed above, and not for other, similar attacks. In this section, we describe similar attacks that we do not attempt to address, and discuss the design choices that led us to draw these boundaries.

Our DoS attacks aims at MAC-and-PHY-compliant attackers. While previous work has examined jamming attacks against the physical layer [22, 23, 27, 33] and other attacks against the MAC layer [11, 13, 15, 19, 25, 28], the aim of digital contact tracing is to be immediately available on commodity hardware, which includes commodity wireless chipsets. As a result, mechanisms that require new approaches to the physical-layer or MAC-layer are non-starters for these protocols. Furthermore, the level of complexity required to build jamming devices or MAC-layer attacks makes these attacks less practical than ones that can be built on any Bluetooth compatible system. When future devices may include support for more robust physical- and MAC-layers, our work can integrate with these approaches for greater robustness.

4 Proposed Approach

In this section, we describe our approach to the attacks discussed earlier. First, we examine how ephemeral IDs can be verified. Previous work [12, 26, 29, 30, 33] have developed schemes that use attacker’s resource limitations (e.g., ability to compute computational puzzles or solve CAPTCHAs) to create fairness between normal users and automated attackers. Our approach to BlindSignedIDs builds on this work to limit the ability of adversaries to create unbounded numbers of valid ephemeral IDs. In particular, verifying a user’s real identity or phone number can serve as the basis for generating a set of valid identifiers in a privacy-preserving way.

4.1 Generating verifiable ephemeral identifiers

BlindSignedIDs. We build verifiable ephemeral identifiers on subliminal-free blind signatures [21], illustrating the approach using RSA blind signatures. The signer first generates a public-private key pair for each day; we call $(e_t, d_t)$ the verification and signing exponent used on day $t - 2$ to create credentials for day $t$. (The modulus $n$ should also change from day to day.) Both the verification exponent, the modulus, and a Prefix$_t$ (used to pad the EphID to a length suitable for signing) are published in advance of day $t - 2$. On day $t - 2$, the user creates a main secret day seed $sk_t$, and uses a pseudo-random generator on $sk_t$ to generate $M$ secondary secret day seeds:

$$ SK_{t_1} || \ldots || SK_{t_M} = \text{PRG} (\text{PRF}(sk_t)) $$

(2) (where $M$ is chosen to reduce the chance that an attacker generates EphIDs corresponding to a secret key; we suggest $M = 100$) and generates $M$ sets of EphIDs using

$$ \{ \text{EphID}_1 || \ldots || \text{EphID}_n \}_M = \text{PRG} (\text{PRF}(SK_{t_M}, "broadcast key M")) $$

(3)

The user then blinds each EphID. To blind EphIDs in a manner easily verifiable by the signer, the user chooses a main blinding seed $b_1$ and uses a pseudo-random generator on $b_1$ to generate $M$ secondary blinding seeds $b_{t_i}$, one for each secondary secret day seed:

$$ b_1 || \ldots || b_{t_M} = \text{PRG} (\text{PRF}(b_1)) $$

(4)

The user then uses a pseudo-random generator on each $b_{t_i}$ to compute a set of blinding values for each EphID, with

$$ \{ r_{t_1} || \ldots || r_{t_n} \}_{t_i} = \text{PRG} (\text{PRF}(b_{t_i})) $$

$$ r_{t_j} = r_{t_i}^{e_t} $$

(5)

where $e_t$ is the signer’s public exponent, and blinding the EphID

$$ \text{EphID'}_{t_j} := (\text{Prefix}_t || \text{EphID}_{t_j}) \cdot r_{t_j} $$

(6)

Prefix$_t$ is a value chosen for that day to pad the EphID to the size of the RSA group. For example, if we use 2048-bit RSA and 104-bit EphID, the Prefix$_t$ is 1944 bits. In this manner, the user generates $M$ sets of $n$ blinded EphIDs, each of which corresponds to a single secondary secret day seed. The user then contacts the backend server, proves
User Real IDs Server

\{\text{EphID}_i\}_1^{100}, \ldots , \{\text{EphID}_i\}_1^{100}

Pick \(s \in \{1, \ldots , 100\}\)

\{\text{SK}_{i, r_i}\}_1^{100}, \ldots , \{\text{SK}_{i, r_i}\}_1^{100}

where \(M \neq s\)

\(\text{SD}_s\)

\(\text{Figure 1: User Registration.}\)

her identity (for example, a phone number, social media account, or national identity), and sends the set of all \(M\) blinded EphIDs to the signer. The signer first verifies the identity and ensures that it has not previously signed certificates for this identity for this day. The signer then chooses one set of blinded EphIDs for this identity for this day. The signer then verifies the identity and ensures that it has not previously signed EphIDs for this identity for this day. The signer then chooses one set of blinded EphIDs and asks the user to reveal the corresponding day seeds of all remaining sets. The user then reveals the requested secret day seeds and their corresponding blinding seeds. If all revealed EphIDs generated correctly, the signer blindly signs all resulting values from the selected set and sends the signed values back to the user, as shown in Equation (7).

\[
\text{SD}_i := \text{EphID}_i^{d_i}
\]

(7)

where \(d_i\) is the private key for signing on day \(t - 2\) (and use on day \(t\)). If EphIDs are not correctly generated, the signer adds the user into a blocklist, such that the credentials she used are not valid at the signer for a certain period of time (the length of time to be chosen should depend on the possibility of reassignment; for example, longer-term blocking for national identities and social media accounts may be reasonable, but blocks for phone numbers should expire after a few months). By ensuring that EphIDs are generated correctly, we can ensure that a positive test can be represented by a single secondary secret-day seed, ensuring that anyone broadcasting verifiable ephemeral identifiers can correctly report a positive test. Figure 1 illustrates the registration process.

The user computes signatures \(\text{SD}_i\) from the signed values \(\text{SD}_i\) by inverting the \(r_i\) used to blind each EphID:

\[
\text{SD}_i := \text{SD}_i^{d_i} * r_i^{-1} \mod N
\]

\[
= ((\text{Prefix} || \text{EphID}_i) * r_i)^{d_i} * r_i^{-1} \mod N
\]

\[
= (\text{Prefix} || \text{EphID}_i)^{d_i} * r_i^{-1} \mod N
\]

\[
= (\text{Prefix} || \text{EphID}_i)^{d_i} \mod N
\]

(8)

All EphIDs generated and signed on day \(t - 2\) are used on day \(t\). We can choose an arbitrary period for signing and refreshing keys, but we choose a day in line with DP-3T [31].

4.2 Authenticating verifiable ephemeral identifiers in Beacons

Because BLE beacons only have 31 bytes of space, of which 3 bytes are used for BLE flags, including a blind signature for an EphID into the beacon would require the use of several beacons, which could lead to computational attacks on packet defragmentation. To avoid these possibilities, we propose to have the signer exchange signed EphIDs for EphIDs authenticated using TESLA [31]. TESLA uses authenticators based on symmetric cryptography to provide broadcast authentication by using the time at which information is disclosed as the mechanism for creating asymmetry. In particular, the entity wishing to authenticate messages (in this case, the signer) creates a sequence of keys using a one-way hash function:

\[
k_1, k_2 = H(k_1), \ldots , k_{t+1} = H(k_t), \ldots , k_{s+1} = H(k_s)
\]

and a schedule at which these keys will be released:

\[
t_1, t_2 = t_1 + T, \ldots , t_{s+1} = t_s + T, \ldots
\]

as illustrated in Figure 2. The signer will release key \(k_1\) at time \(t_1\), so if a verifier receives a message before the signer’s clock has reached \(t_1\), the signer knows that the message is authentic if it is authenticated using the key \(k_1\). TESLA relies on loose time-synchronization for security, meaning that each clock must be within a bounded margin of error \(\Delta\) from the signer’s clock (in our design, \(T\) will be approximately 5 minutes, and the maximum time-synchronization error will be around 10 seconds). Thus, if the message is received before time \(t_i - \Delta\) on the receiver’s clock, the receiver knows that it is before time \(t_i\) on the signer’s clock.

To exchange RSA blind-signed verifiable ephemeral identifiers for TESLA-authenticated MACs corresponding to each EphID, a user requests them through a MIX [20] on day \(t - 1\), providing a nonce, an encryption
4.3 Storing Received Beacons

When a device receives a beacon, the user stores them as $(\text{EphID}_i, \text{Auth}_i, \text{time}, \text{RSSI})$. When the user obtains the current released key $k_i$ at time $t_i$ from TESLA server, she firstly verifies $k_i$ with $k_{i-1}$ by a one-way hash function; she then verifies each EphIDs received in the previous period by checking its authenticator $\text{Auth}_i$. If the $\text{Auth}_i$ is valid, the device accepts the record and keeps that record in longer-term storage; authenticator verification can be performed in-place to mitigate DoS attacks. Optionally, our protocol can be combined with [32] to prevent relay and replay attacks. However, the details of preventing relay and replay attacks are beyond the scope of our protocol.

4.4 Checking Exposure

When a health authority determines a user is a positive case, all her secondary secret-day seeds and the selected numbers $s$ during the infectious period (typically 2 days before symptoms happen) are published in a blockchain, together with a sequential case number for the day of publication. Each user fetches the list of these positive cases, and generates EphIDs with the secondary secret-day seeds and the selected numbers $s$, and compares the EphIDs with the EphIDs in their local storage. If there is a match, the device can suspect contact with a positive case.

**FinalTrial.** We propose to use the blockchain to provide an additional defense to the identifier-spoofing attacks. When a user generates BlindSignedID, she also requests blind-signatures for a set of codes of the form $i \parallel n_i$, where $n_i$ is a nonce and $i$ represents a sequential case number; these codes are signed with a daily signing key different than the signing key used for BlindSignedID. Specifically, the user generates sufficient codes to respond to the maximum number of positive tests that might reasonable affect her jurisdiction for a day. Each such test is a nonce $n_i$. She then builds a Merkle tree over these nonces, by first hashing each one, then hashing adjacent pairs to reach a tree root. She blinds the tree root $R$ by calculating $R^{n_i}$, has it signed as $s = (R^{n_i})^{d_i}$, and unblinds the signature by calculating $SP = r_i^{-1}s_i = R^{d_i}$. When a device matches positive case number $i$, instead of immediately notifying the user, the device posts $n_i$, $SP$, and the Merkle path required for verification to the blockchain, allowing each user to determine the number of matches corresponding to each positive case. When a single positive case experiences an excessive number of matches, the device may choose to warn the user that the
match seems unlikely, or may entirely ignore the match.

5 Privacy Analysis

In our design, real IDs (such as phone numbers) are revealed to a signing server before the server provides EphIDs. Since EphIDs are blind-signed with a subliminal-channel-free signature, each EphID is indistinguishable, and could be associated with any real ID that was presented on the day on which it was signed.

Lemma 1. Given two signed EphIDs, EphID\(_1\) and EphID\(_2\), and two possible identities, A and B, the signer can gain no advantage on which EphID is assigned to which identity; that is, any attempt to assign EphID\(_i\) to A succeeds with probability 1/2.

Proof. The signer generates a RSA key pair \((e, d)\) and a public modulo \(N\), and publishes the public exponent and modulo \((e, N)\). The user generates a set of ephemeral identifiers, \(\{\text{EphID}\_1, \ldots, \text{EphID}\_n\}\), and a set of random numbers raised by the public exponent \(e \mod N\) to create blindedephID factors, \(\{r\_1^\text{ed} \mod N, \ldots, r\_n^\text{ed} \mod N\}\). She multiplies the ephemeral identifiers with the blinding factors to obtain the resulting value set \(\mathcal{R}\).

\[\mathcal{R} := \{\text{EphID}\_1 \ast r\_1^\text{ed} \mod N, \ldots, \text{EphID}\_n \ast r\_n^\text{ed} \mod N\}\] (9)

The set \(\mathcal{R}\) is sent to the signer, and the signer signs each EphID in \(\mathcal{R}\) using the private key \(d\). Since \(r\_i^{ed} = r\_i \mod N\), the signer generates a signature set \(\mathcal{SD}\)

\[\mathcal{SD} := \{\text{EphID}\_1 \ast r\_1 \mod N, \ldots, \text{EphID}\_n \ast r\_n \mod N\}\] (10)

The signer learns the signature set \(\mathcal{SD}\) is associated with the user’s real IDs. However, for any pair of values \((x, y)\) in the signature set \(\mathcal{SD}\)

\[(\text{EphID}\_x \ast r\_x \mod N, \text{EphID}\_y \ast r\_y \mod N) \leftrightarrow \mathcal{SD}\] (11)

Since the blinding factors are randomly generated, they could take on any value. For example, if we consider two EphIDs, EphID\(_1\) and EphID\(_2\), blinded as follows:

\[(\text{EphID}\_1 \ast r\_1^\text{ed}, \text{EphID}\_2 \ast r\_2^\text{ed})\]

these are indistinguishable from

\[(\text{EphID}\_2 \ast (\text{EphID}\_1 \ast \text{EphID}\_2 \ast r\_1^\text{ed} \ast r\_2^\text{ed}), (\text{EphID}\_1 \ast (\text{EphID}\_2 \ast \text{EphID}\_1 \ast r\_1^\text{ed}))\].

In other words, for each EphID \(\text{EphID}\_i\) and each blinded EphID EphID\(_i\), there exists a blinding factor for EphID\(_i\) such that the signing request for EphID\(_i\) would create a signature for EphID\(_i\). Naturally, the security of the blind-signature scheme requires that such blinding factors be computationally difficult to compute, but they exist and are equiprobable blinding factors: specifically, if a signer generates \(N\) signatures on one day, then for any given EphID, there are \(N\) blinding possible blinding factors, each of which corresponds to one blind-signature request. Each such blinding factor is indistinguishably probable to the signer.

When a user provides authenticators to the signer, she also includes the time interval and a nonce; the signer therefore learns only the which EphIDs are used in each time interval, and the correlation between nonces and EphIDs. First, if the signer is not compromised, there is no privacy loss, because only the signer has the correlation between EphID and nonce. Even when the signer is compromised, a user adopting the full download method reveals no information about her authenticators. However, the size of download lists might be multiple gigabytes if there are millions of participants. The partial download method saves user resources at the cost of some privacy in case of a compromised signer; however, revealing only a small part of the nonce \((e.g., 1\) byte of a 16 byte nonce), each user still retains a significant anonymity set.

Finally, the individual download method loses more privacy in case of a compromised signer, but the specific amount of privacy loss depends on the mechanism by which the values are downloaded (for example, when downloading over Tor\[9]\), the privacy properties are the same as Tor’s).

6 Security Analysis

Mitigate DoS attacks. BlindSignedIDs mitigate the DoS attacks by ensuring that each EphID stored by a user is an EphID generated for a valid user. BlindSignedID provides an in-place verification to mitigate the DoS attacks on the local storage in that the user will reject an arbitrarily generated EphID that is not associated with a real identity.

7 Evaluations

7.1 BlindSignedID on DoS attacks

Single attacker. In this section, we evaluated BlindSignedID via sending BLE beacons using mobile devices. Figure\[4\] shows our experimental setup. We implemented a release key UDP server as the TESLA server, and each key was released every 5 mins. Moreover, we built an Android application to advertise BlindSignedID and scan BLE beacons on two victim’s phones placed at a distance of around 20 cm. For the single attacker, we generated a set of 70,000 random EphIDs sent by a single Raspberry Pi 3 Model B with the distance 1.5 m. We used the first 13-byte of HMAC-SHA1 as authenticator.
Each random BLE beacon was broadcasted and the rotating time was 20 ms. The receiver stored each EphID with the time of first receipt, duration, RSSI and authenticator temporarily. Whenever a key release is scheduled, the receiver fetches the key over UDP, checks the key, and for each record, attempts to verify the record. When a record is verified, the receiver puts the time of first receipt, duration of contact, and RSSI in local storage, consuming 38 bytes per EphID. Figure 5 shows our results. Our BlindSignedID rejected all DoS attacks periodically created by the attacking set, and prevented nearly 30,000 fake EphIDs stored in 30 mins. Consequently, the original EphID designs are vulnerable to DoS attacks, and BlindSignedID can mitigate such attacks in-place. In an actual attack, the attacker can further modify BlueZ stack to transmit BLE beacons as frequently as possible, so DoS attacks are even more severe under such circumstances.

Multiple attackers. In addition, we evaluated BlindSignedID with multiple attackers to somewhat mitigate the long interval between attacker EphIDs. We generated 4 sets of 4,320,000 random EphIDs, which were used by 2 and 4 Raspberry Pis. Figure 6 shows that 2 and 4 attackers using off-the-shelf BlueZ broadcast intervals can cause approximately 1,227,000 and 1,891,000 stored EphIDs within 8 hrs, representing 645 MBytes and 1.076 GBytes across 14 8-hour days, respectively. Due to networking and processing delays, receivers do not filter all of the invalid EphIDs every 5 minutes, but the system still trends towards significant reductions in storage requirements.

Crowded environment. We performed 3 experiments where over 14 people were originizied in grid positions with a distance of 0.5 m, 1 m, and 1.5 m in a middle-size room, respectively. (These experiments were conducted in compliance with local social distancing requirements.) We used 6 Raspberry Pis as the attackers to send random EphIDs near the same power outlet. Each scenario was conducted for 1–2 mins. Table 1 reported the numbers of received EphIDs and real EphIDs verified by BlindSignedID. As distance increases, devices receive fewer EphIDs due to signal strength losses over longer distances. Also, our BlindSignedID continues to effectively reduce storage consumption, removing over 90% of the identities produced by DoS attacks.

8 Related Works

This section reviews several papers that inspire BlindSignedID.

Digital Contact Tracing. BlindSignedID mostly fol-
Table 1: DoS attacks on crowded indoor environment.

| Scenario Distance | Original EphIDs | BlindSignedID | Reduced Rate |
|-------------------|-----------------|--------------|--------------|
| 0.5 m             | 12,122          | 181          | 96.72%       |
| 1.0 m             | 6,526           | 76           | 97.87%       |
| 1.5 m             | 2,988           | 28           | 99.95%       |

DoS attacks however, current designs suffer from DoS attacks. Most current projects follow the similar designs. In particular, we propose BlindSignedIDs that are verifiable without compromising the user’s privacy to mitigate DoS attacks. Anonymous identifiers do cause the mass surveillance difficult to be conducted. However, using anonymous identifiers without verifications introduces security issues in the current designs. Finally, we evaluated our BlindSignedIDs with example DoS attacks and showed BlindSignedIDs can reduce DoS attacks.

9 Conclusion

In this paper, we present BlindSignedIDs, which can mitigate DoS attacks on the current digital contact tracing designs. In particular, we propose BlindSignedIDs that are verifiable without compromising the user’s privacy to mitigate DoS attacks. Anonymous identifiers do cause the mass surveillance difficult to be conducted. However, using anonymous identifiers without verifications introduces security issues in the current designs. Finally, we evaluated our BlindSignedIDs with example DoS attacks and showed BlindSignedIDs can reduce DoS attacks.

Acknowledgment

This work is supported by NSF under CNS-17313 and R.O.C. (Taiwan) Ministry of Education for MOE 108 Government Scholarship to Study Abroad. We thank Professor Hsu-Chun Hsiao for supporting experiments at National Taiwan University and offering valuable feedback.

References

[1] Bluetooth technology protecting your privacy. https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/. Accessed: 2020-08-20.
[2] Bluez: le_set_advertising_parameters_cp. https://github.com/pauloborges/bluez/blob/master/lib/hci.h#L1496 Accessed: 2020-07-08.
[3] Cdc digital contact tracing tools for covid-19. https://www.cdc.gov/coronavirus/2019-ncov/downloads/digital-contact-tracing.pdf Accessed: 2020-06-26.
[4] Google/apple privacy-preserving contact tracing. https://www.apple.com/covid19/contacttracing Accessed: 2020-05-28.
[5] Japan releases contact-tracing app using apple and google tech. https://www.engadget.com/microsoft-built-japans-contacttracing-app-using-apple-and-google-tech-105556846.html Accessed: 2020-06-23.
[6] Pact: Private automated contact tracing. https://pact.mit.edu Accessed: 2020-05-28.
[7] Pan-european privacy-preserving proximity tracing. https://www.pepp-pt.org Accessed: 2020-06-06.
[8] Tcn protocol. https://github.com/TCNCoalition/TCN Accessed: 2020-06-06.
[9] Tor metrics. https://metrics.torproject.org Accessed: 2020-08-20.
[10] Uk abandons contact-tracing app for apple and google model. https://www.theguardian.com/world/2020/jun/18/uk-poised-to-abandon-coronavirus-app-in-favour-of-apple-and-google-models Accessed: 2020-06-23.
[11] Baruch Awerbuch, Andrea Richa, and Christian Scheideler. A jamming-resistant mac protocol for single-hop wireless networks. In Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing, pages 45–54, 2008.
[12] Vikas Bajaj. Spammers pay others to answer security tests. https://www.nytimes.com/2010/04/26/technology/26captcha.html, April 2010. Accessed: 2020-06-19.
[13] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongguan, Janice Tan, and Tung Anh Quy. Blue-trace: A privacy-preserving protocol for community-driven contact tracing across borders. Government Technology Agency-Singapore, Tech. Rep, 2020.
[14] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In USENIX security symposium, volume 12, pages 2-2. Washington DC, 2003.
[15] Alvaro A Cárdenas, Svetlana Radosavac, and John S Baras. Performance comparison of detection schemes for mac layer misbehavior. In IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications, pages 1496–1504. IEEE, 2007.
[16] Claude Castelluccia, Nataliia Biedova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Météayer, and Vincent Roca. Robert: Robust and privacy-preserving proximity tracing, 2020.
[17] Justin Chan, Dean Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Purneet Sharma, Sudheesh Singanamalla, Jacob Sunshine, and Stefano TESSA. pact: Privacy sensitive protocols and mechanisms for mobile contact tracing, 2020. arXiv:2004.03544 [cs.CR].
[18] Sang-Yoon Chang and Yih-Chun Hu. Securemac: Securing wireless medium access control against insider denial-of-service attacks. IEEE Transactions on Mobile Computing, 16(12):3527–3540, 2017.
[19] Sang-Yoon Chang, Yih-Chun Hu, and Nicola Laurenti. Simplemac: A simple wireless mac-layer countermeasure to intelligent and insider jammers. IEEE/ACM Transactions on Networking, 24(2):1095–1108, 2015.
[20] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM, 24(2):8490, 1981.
[21] David Chaum. Blind signatures for untraceable payments. In Advances in cryptography, pages 199–203. Springer, 1983.
[22] Jerry T Chiang and Yih-Chun Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. IEEE/ACM transactions on networking, 19(1):286–298, 2010.
[23] Yvo Desmedt, Rei Safavi-Naini, Huaxiong Wang, Lynn Batten, Chris Charmes, and Josef Pieprzyk. Broadcast anti-jamming systems. Computer Networks, 35(2-3):223–236, 2001.
[24] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtago, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. Science, 368(6491), 2020.
[25] Vikram Gupta, Srikanth Krishnamurthy, and Michalis Faloutsos. Denial of service attacks at the mac layer in wireless ad hoc networks. In MILCOM 2002. Proceedings, volume 2, pages 1118–1123. IEEE, 2002.
[26] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Secure information networks, pages 258–272. Springer, 1999.
[27] Markrey Hedy Kiesler and Antheil George. Secret communication system, August 11 1942. US Patent 2,292,387.
[28] Pradeep Kyasanur and Nitin H Vaidya. Selfish mac layer misbehavior in wireless networks. IEEE transactions on mobile computing, 4(5):502–516, 2005.
[29] Zhutoao Liu, Yushan Liu, Philipp Winter, Prateek Mittal, and Yih-Chun Hu. Torpolice: Towards enforcing service-defined access policies for anonymous
communication in the tor network. In 2017 IEEE 25th International Conference on Network Protocols (ICNP), pages 1–10. IEEE, 2017.

[30] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: Protecting connection setup from denial-of-capability attacks. ACM SIGCOMM Computer Communication Review, 37(4):289–300, 2007.

[31] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. The tesla broadcast authentication protocol. Rsa Cryptobytes, 5(2):2–13, 2002.

[32] Krzysztof Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. Technical report, Cryptology ePrint Archive, Report 2020/418, 2020. https://eprint.iacr.org/2020/418, 2020.

[33] Christina Popper, Mario Strasser, and Srdjan Ćapkun. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. ieee journal on selected areas in communications, 28(5):703–715, 2010.

[34] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing, 2020, arXiv:2005.12273 [cs.CR].

[35] Serge Vaudenay. Analysis of dp3t. Cryptology ePrint Archive, Report 2020/399, 2020. https://eprint.iacr.org/2020/399.

[36] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recapTCHA: Human-based character recognition via web security measures. Science, 321(5895):1465–1468, 2008.

[37] Wikipedia. List of countries and dependencies by population. https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population, 2020. Online; accessed: 2020-06-26.