Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Privacy-preserving COVID-19 contact tracing solution based on blockchain

Momeng Liu a, Zeyu Zhang a, Wenqiang Chai a, Baocang Wang b

a Shaanxi Key Laboratory of Clothing Intelligence, State and Local Joint Engineering Research Center for Advanced Networking and Intelligent Information Services, School of Computer Science, Xi’an Polytechnic University, Xi’an 710048, China
b State Key Laboratory of Integrated Service Networks, Xidian University, Xi’an 710071, China

ARTICLE INFO

Keywords:
Blockchain
Privacy-preserving contact tracing
COVID-19
Zero-knowledge proof
Aggregate Signature

ABSTRACT

The COVID-19 pandemic has severely affected daily life and caused a great loss to the global economy. Due to the very urgent need for identifying close contacts of confirmed patients in the current situation, the development of automated contact tracing app for smart devices has attracted more attention all over the world. Compared with expensive manual tracing approach, automated contact tracing apps can offer fast and precise tracing service, however, over-pursing high efficiency would lead to the privacy-leaking issue for app users.

By combing with the benign properties (e.g., anonymity, decentralization, and traceability) of blockchain, we propose an efficient privacy-preserving solution in automated tracing scenario. Our main technique is a combination of non-interactive zero-knowledge proof and multi-signature with public key aggregation. By means of aggregating multiple signatures from different contacts at the mutual commitment phase, we only need fewer zero-knowledge proofs to complete the task of identifying contacts. It inherently leads to the benefits of saving storage and consuming less time for running verification algorithm on blockchain. Furthermore, we perform an experimental comparison by timing the execution of signature verification with and without aggregate signature, respectively. It shows that our solution can actually preserve the full-fledged privacy protection property with a lower computational cost.

1. Introduction

Due to the explosion of Coronavirus Disease 2019 (COVID-19) has globally affected on every aspect in daily life, the World Health Organization (WHO) officially announces COVID-19 as a public health emergency [1]. In order to alleviate the spread of virus and avoid re-infection cases, national governments have advocated the counter-measure of self-isolation at home, and deployed various basic health facilities for large-scale population nucleic acid. However, these strategies seriously damage the global economy. Because of the highly contagious nature of COVID-19, the diverse and ambiguous transmission vectors, and the existence of plentiful asymptomatic infection cases, it is very urgent and demanded for an efficient contact tracing solution to identify close contacts of confirmed COVID-19 patients. Therefore, we can assess the risk of potential infection by the proximity and duration information collected from these contacts. However, most people are reluctant to share their personal information in public. It is difficult to complete the task for contact tracing and risk evaluation. Since inefficient and expensive manual tracing approach involves a mass of medical force, it may also cause further infections among those precious human resources.

Since most mobile devices (i.e., smartphone) are loaded with GPS and Bluetooth, they can communicate with nearby devices in a certain close distance and threshold time duration. Many researchers propose to use smartphone apps for automated tracing and detection. However, some automated contact tracing apps [2–4] claim that they can protect user’s privacy and not disclose private information without user’s consent. But those specific tracing processes always more or less expose user’s information. The privacy protection on automated tracing apps is worth to be actively explored.

Blockchain is a distributed ledger proposed by Satoshi Nakamoto for bitcoin [5] and widely used in the field of cryptocurrencies [6]. With the continuous improvement of blockchain-based infrastructures, blockchain technology can provide more reliable proof of existence for various digital assets [7,8]. It also has a huge potential in IoT security [9–12], data privacy-preserving [13–15], security audit [16,17], anonymous authentication [18], and anonymity protection in cryptocurrencies [19]. Moreover, blockchain can also provide significant privacy-preserving advantages for contact tracing apps.

In general, blockchain can be viewed as a distributed structure
tampered so that avoiding the case of false positive reports. Furthermore, blockchain can provide anonymity property for protecting user’s identity privacy. Therefore, when comparing to the traditional statistical techniques run by an inefficient centralized authority management, the above beneficial features of blockchain can jointly support automated contact tracing apps for a better privacy-preserving tracing solution.

Related work

The government of Singapore first releases a contact tracing app TraceTogether [20], which is based on Blue-Trace protocol [21]. Some similar apps are subsequently published, such as CovidSafe [22] and Stop Covid [23]. However, these apps cannot offer the privacy-preserving service. Later on, the work of De Carli et al. [24] proposes a solution to protect the privacy of confirmed patients using some cryptographic tools, and a location-based tracing scheme [25] relying on secure multi-party computation can also offer privacy protection. However, none of them is good enough to address all-sided privacy issues and prevent from the false positive reports case. Although the work of Troncoso et al. [26] proposes a decentralized solution “Decentralized Privacy-Preserving Proximity Tracing” to strengthen the privacy-preserving property, it is still vulnerable to the case of false positive reports. In addition, since most distributed solutions have to face the difficulty on interoperability, Google and Apple [27] jointly develop a contact tracing app compatible to different mobile phone operating systems.

Most of the above solutions do not consider the significant location information in automated contact tracing. Relying on the advantages of blockchain technology, Amoretti et al. [28] and Song et al. [29] both use Bluetooth and location data for contact tracing, but result in the overload problem on blockchain. The work of Xu et al. [30] proposes the BeepTrace with offering privacy protection for users, however, the risk of forging location data still exists. Therefore, Lv et al. [31] uses zero-knowledge proof to ensure the reliability of location information in their location-based blockchain tracing scheme. Unfortunately, this scheme does not fully cover all privacy concerns. Then the work of Avitabile et al. [32] presents a scheme called PRONTO-C2, which utilizes proximity tracing to protect the anonymity of the confirmed patient’s identity and avoid the false positive report case, but with the price of consuming large communication overhead. To the best of our knowledge, the work of Liu et al. [33] proposes a much better privacy-preserving contact tracing framework with very full-fledged privacy protection properties. They utilize zero-knowledge proof to protect the identity information of app user, and group signature to avoid the false positive report case. The blockchain used in their framework works as a bulletin board for storing and updating public daily information. However, their solution requires to generate an independent zero-knowledge proof for each close contact of a confirmed patient. As the number of app users increasing, the costs on computation, communication and storage are all relatively growing.

Our contributions

The above mentioned privacy-preserving tracing solutions do not pay enough attentions to protect the privacy of close contacts of confirmed patients. Therefore, focusing on the privacy protection for contacts, we follow the work of Liu et al. [33] and propose a privacy-preserving COVID-19 contact tracing solution based on blockchain technology. Besides combining with the advantages of blockchain, we also adopt zero-knowledge proof and aggregate signature to jointly support the privacy protection for tracing app users and offer a better performance on efficiency aspect when comparing to the work of Liu et al. [33]. The main cryptographic tool we adopt is an aggregate signature variant. This is because the general aggregate signature [34] is vulnerable to the rogue public-key attack, and the contact tracing framework we use requires several users to sign the same identity message. Therefore, a multi-signature with public key aggregation [35] scheme can be very suitable in our solution, which further saves the storage space and improves the verification speed on chain.

2. Preliminaries

In this section, we introduce some cryptographic primitives regarding to our scheme, including bilinear mapping, digital signature [36], multi-signature with public-key aggregation [35], group signature [37], and zero-knowledge proof. All the primitive definitions and security notions are following up the corresponding references as we cited.

2.1. Bilinear mapping

Let $G_1, G_2, G_T$ be cyclic groups of prime order $q$. Let $u$ and $g$ be generators of $G_1$ and $G_2$, respectively. If a mapping $e: G_1 \times G_2 \rightarrow G_T$ holds for the following properties, we called it a bilinear mapping.

- **Bilinearity:** for all $u, g \in G_1$ such that $e(u^a, g^b) = e(u, g)^{ab}$
- **Non-degeneracy:** for all $u, g \in G_1$ such that $e(u, g) \neq 1_{G_T}$, where $1_{G_T}$ is the identity of $G_T$.
- **Computability:** There are efficient algorithms to compute $e(u, g)$ for all $u \in G_1$ and $g \in G_2$.

2.2. Discrete logarithm problem

The discrete logarithm problem is defined as: given a cyclic group $G$, a generator $g$ of $G$ and an element $a \in G$, to find the discrete logarithm $x \in \mathbb{Z}_q$ such that $a = g^x$, which is denoted by $DL_q(a)$ in this paper.

2.3. Decisional Diffie–Hellman assumption

Let $G$ be a cyclic group of prime order $q$, $g$ is a generator of $G$. The decisional Diffie–Hellman (DDH) assumption states that given $g^x$ and $g^y$, where $x, y$ are chosen uniformly and independently from $\mathbb{Z}_q$. The following two distributions are computationally indistinguishable (in the security parameter, $n = \log(q)$) for any polynomial-time algorithm:

- $(g^x, g^y, g^{xy})$, where $x$ and $y$ are chosen uniformly and independently from $\mathbb{Z}_q$.
- $(g^x, g^y, g^z)$, where $x, y, z$ are chosen uniformly and independently from $\mathbb{Z}_q$.

2.4. Digital signatures

In this section, we introduce three different notions of digital signature, including basic digital signature [36], multi-signature with public-key aggregation [35], and group signature [37], which are all taken as building blocks in our solution.

**Digital signature** A digital signature scheme consists of three polynomial-time algorithms $(Gen, Sign, Vrfy)$ such that

- The key generation algorithm $(pk, sk) \leftarrow Gen(\lambda)$: Take as input a security parameter $\lambda \in \mathbb{N}$ and output a key pair $(pk, sk)$.
The signing algorithm $\sigma \leftarrow \text{Sign}_a(m)$: Take as input private key sk and message $m$ to generate a signature $\sigma$ as output.

The verification algorithm $b \leftarrow \text{Vrfy}_{pk}(m, \sigma)$: Take as input the public key $pk$, message $m$, and signature $\sigma$, and output a bit $b$. If $b = 1$, the signature is valid. Otherwise, the signature is invalid.

**Multi-signature with public-key aggregation** A multi-signature with public-key aggregation scheme based on bilinear pairing consists of four polynomial-time algorithms, i.e., \( (\text{KeyGen}, \text{Sign}, \text{AggregateAggrega} \text{t}, \text{AggregateVerify}) \). We choose two hash functions as random oracles, i.e., \( H_1 : \mathcal{M} \rightarrow G_1, H_2 : G_2^m \rightarrow \mathbb{R}^n \), where \( R := \{1, 2, \ldots, 2^{128}\} \) and \( 1 \leq n \in \mathbb{N} \).

- The key generation algorithm \( \text{KeyGen} \): Randomly select $\alpha \leftarrow Z_q$ as the private key, calculate $v \leftarrow g^\alpha$ as the public key, and output the public key $pk = v$ and private key $sk = \alpha$.
- The signing algorithm \( \text{Sign} \): Take as input the private key $\alpha$ and message $m$, and output the signature $\sigma = H_1(m)^\alpha \in G_1$.
- The multi-signature aggregation algorithm \( \text{Aggregate} \): Take as input \( ((pk_1, \sigma_1), \ldots, (pk_n, \sigma_n)) \), calculate \( (t_1, \ldots, t_n) = H_2(pk_1, \ldots, pk_n) \in \mathbb{R}^n \), and output aggregate signature $\sigma' = \sigma_1 \cdot \ldots \cdot \sigma_n \in G_1$.
- The multi-signature verification algorithm \( \text{AggregateVerify} \): Take as input \( (pk_1, \ldots, pk_n, m, \sigma) \), calculate \( (t_1, \ldots, t_n) = H_2(pk_1, \ldots, pk_n) \in \mathbb{R}^n \), then compute the aggregate public key $apk = pk_1 \cdot \ldots \cdot pk_n \in G_2$. If $e(g, \sigma) = e(apk, H_1(m))$ is satisfied, then accept. Otherwise, reject.

**Group signature** A group signature scheme consists of four polynomial-time algorithms \( (\text{Gkg}, \text{GSig}, \text{GVf}, \text{Open}) \) such that:

- The group key generation algorithm \( \text{Gkg} \): Execute the Gkg algorithm to generate a tuple \( (gpk, gmsk, gsk) \), where \( gpk \) is group public key, \( gmsk \) is group manager secret key, \( gsk \) is an $n$ dimensional vector composed of $n$ different secret signing keys $gsk[i]$’s for all $i \in [n]$.
- The group signing algorithm \( \text{GSig} \): Take as input secret signing key $gsk[i]$ (where $i \in [n]$) and a message $m$, output a group signature $\sigma$ on $m$.
- The group signature verification algorithm \( \text{GVf} \): Take the group public key $gpk$, message $m$, and group signature $\sigma$ as input, and output 1 or 0. If the output is 1, the group signature is valid. Otherwise, it is invalid.
- The opening algorithm \( \text{Open} \): Take the group manager secret key $gmsk$, message $m$, and group signature $\sigma$ as input, and output the identity $i$ of the group member (i.e., the signer), or the symbol $\perp$ to indicate the failure to open the signature.

### 2.5. Zero-knowledge proof

Zero-knowledge proof is a two-party interactive proof between a prover and a verifier. Let $\mathcal{A}$ be an efficient relation. \( L_{\mathcal{A}} \) is a language defined by $\mathcal{A}$ such that $L_{\mathcal{A}} = \{(x, \omega) \in \mathcal{A} \}$, where $\omega$ is a witness for the statement $x$. It requires that the prover can prove $x \in L_{\mathcal{A}}$ without revealing any knowledge of $x$ to the verifier. If the interaction between two parties holds for the following three properties, we call it a zero-knowledge proof.

- **Completeness**: The verifier always accepts when the prover and the verifier are both honest.
- **Soundness**: The verifier accepts with negligible probability when $x \not\in L_{\mathcal{A}}$, for any dishonest prover.
- ** Honest Verifier Zero-Knowledge**: There exists a simulator $\mathcal{S}$ such that $\mathcal{S}(x)$ is indistinguishable from a real proof execution.

### 3. Assumptions and threat model

In this section, we will illustrate some assumptions and the threat model used in our solution. In our scheme, there are four participants involved in, which are government, doctor, user, and blockchain.

#### 3.1. Assumptions

Our solution follows the following reasonable assumptions:

- The app users cannot modify the app, but the owner of smartphone can read all data generated, stored and communicated through the app.
- The app and the smartphone are both connected to the well-functioning Internet.
- The smartphone is enabled with Bluetooth connectivity.
- The app users will not reveal their social activities to the public, e.g. uploading a photo with someone on Instagram.
- The app users will not share the secret keys with other people.

#### 3.2. Threat model

We assume that all attackers are honest but curious and we only consider cryptographic attacks in this paper. Attackers can be divided into the following categories:

- Doctors are honest but curious, and they may be interested in the identity of close contacts of confirmed patients.
- Users may be interested in the identities of other users, and they may pretend to be confirmed patients and cause public panic.
- External attackers may be interested in user identities.

For simplicity, we use the following scenario as shown in Fig. 2, to illustrate the threat model utilized in our solution. We assume there are five users in our case, they are Alice, Bob, Charlie, Peter and John, Alice is a COVID-19 positive patient, Bob and Charlie are close contact of Alice, Peter and John are COVID-19 negative. We hope the contact tracing framework can satisfy the following three properties.

- **The security of confirmed patient**: Except for the government and doctors, any polynomial-time adversary cannot identify the identity of the confirmed patient. When some confirmed patients have close contacts, they will not reveal their identities information or healthy status to their contacts. In Fig. 2, Alice is confirmed as a COVID-19 positive patient. Bob and Charlie are both close contacts of Alice.
However, Bob and Charlie will not know the current healthy status of Alice.

- **The security of contacts**: Nobody (including the government and doctors) will know any information about the contacts. Any polynomial-time adversary cannot identify who is the contact of a confirmed patient. In Fig. 2, no one can know Bob and Charlie are close contacts of Alice.

- **The precaution of false positive reports**: Any undiagnosed user cannot forge a false report to claim as a confirmed COVID-19 positive patient for maliciously causing public panic. Moreover, any confirmed person cannot claim he has been contact with someone who is actually not a close contact. In Fig. 2, except for Alice, no one can pretend to be a confirmed COVID-19 positive patient.

### 4. System framework

In this section, we first introduce all participants involved in the contact tracing framework [33] and illustrate how does the interaction work among those entities. Then we propose an efficient blockchain-based privacy-preserving contact tracing solution using digital signature [36], multi-signature with public-key aggregation [35], group signature [37], and zero-knowledge proof as building blocks for achieving a better performance when comparing to the work of Liu et al. [33].

#### 4.1. Contact tracing framework

As the COVID-19 situation becomes fiercely and the awareness of personal privacy is awaken off, nowadays an efficient automated contact tracing solution with providing privacy protection property is very demanded. To best of our knowledge, the work of Liu et al. [33] proposes a delicate framework which can offer considerably comprehensive privacy attributes. Therefore, our proposal follows up their framework, where four entities (i.e., government, doctor, user, and blockchain) are involved. They mainly work as following procedure:

- **Government**: It is responsible for initializing the entire system, i.e., mainly serving for the registration of doctors and users. We assume the government has its own public key \( pk_0 \) and secrete key \( sk_0 \).

- **Doctor**: He or she is authenticated to be affiliated with some certain hospital. People only can be viewed as a confirmed positive patient when his test report is signed by an authenticated medical doctor.

- **User**: Referred as the person who has a smartphone loaded with the tracing app. For simplicity, we assume a scenario with five users as shown in Fig. 2. Alice is a COVID-19 positive patient, Bob and Charlie are close contacts of Alice, and Peter and John are COVID-19 negative.

- **Blockchain**: It works as a bulletin board for publishing some public information, where no one can tamper the data has been stored. App users have the right to access in and doctors can upload data to this blockchain.

The entire execution of this framework (as shown in Fig. 3) can be divided into four phases, i.e., registration phase, meeting phase, medical treatment phase, and tracing phase.

1. **Registration Phase**: This stage is mainly for user’s registration in the government with their own public key. The user can select his/her a fresh key pair everyday and upload the daily public key to the government for avoiding repeated registration, so that the government can link to the user’s authentic identity. In addition, the doctor will obtain an additional user secret key from his affiliation for generating a group signature to sign test reports on behalf of his hospital.

2. **Meeting Phase**: The tracing app of each user (e.g., Alice) will use Bluetooth to broadcast a package to other users’ smartphones periodically (e.g., at one minute interval). Once a threshold number of the same package are received by someone (e.g., Bob) within a certain time slot (e.g., 20 minutes), then the package receiver (e.g., Bob) is viewed as a close contact of the package sender (e.g., Alice). Later on, a process of mutual package validation will be jointly executed between this pair of users (e.g., Alice and Bob). If both of them do not abort in the process, two different credentials would be separately stored in their own smartphone. If one of two users (e.g., Alice) is confirmed as a COVID-19 patient, the credential stored in his/her smartphone will be used to prove to the doctor that the opposite person (e.g., Bob) is a close contact.

3. **Medical Treatment Phase**: The confirmed patient (e.g., Alice) will perform a zero-knowledge proof protocol (generated by the credential) with a medical doctor, to prove he/she has close contact with someone (e.g., Bob). The doctor does not know anything about the identity of the close contact in the whole process. Then the doctor will sign the proof by a group signature user secret key (distributed by his affiliated hospital) and upload the group signature with the proof to the blockchain for public verification.

4. **Tracing Phase**: Users can perform some computations with their own secret keys to check whether the new entries published in the blockchain are referring to themselves.

#### 4.2. Detailed scheme

**Registration phase**

Take \( \lambda \in \mathbb{N} \) as input, where \( \lambda \) is a security parameter. We assume \( G_1, G_2, G_T \) are three multiplicative cyclic groups such that a bilinear function \( e: G_1 \times G_2 \rightarrow G_T \) and \( |G_1| = |G_2| = |G_T| = q \), where \( q \) is a prime number of \( \lambda \) bit. The government chooses \( u, u_1, u_2 \in G_1, p, p_1, p_2 \in G_2 \) as the generators of \( G_1, G_2 \) respectively. We denote \( \mathcal{H}: \{0, 1\}^\lambda \rightarrow G_2 \) as a cryptographic hash function. \( \mathcal{H} \) runs a key generation algorithm to generate its own public and private key pair \( (pk_0, sk_0) = \mathcal{Gen}(\lambda) \), and publish the system parameters \( \lambda, \mathcal{H}(pk_0.u, u_1, u_2, p, p_1, p_2) \), which will be downloaded together with the tracing app from
official app store into use’s smartphone. The process of registration phase is shown in Fig. 4.

Each user (e.g., Alice) registers in $\mathcal{F}$ everyday. For example, Alice selects her private key $sk_A$ as $a \in \mathbb{Z}_q$, computes and uploads the public key $pk_A$ as $A = p^a$ to $\mathcal{F}$ with her personal information. $\mathcal{F}$ generates an identifier $id_A$ and a certificate $Cert_A = \text{Sign}(sk_A, \{\text{Status}. A.id_A. Date\})$ for Alice, where $\text{Status}$ is referring to the healthy status of Alice on the date of $\text{Date}$. If $\text{Status} = 0$ (resp., $\text{Status} = 1$), it represents the user is confirmed COVID-19 negative (resp., COVID-19 positive). Alice checks whether $Cert_A$ is valid by $pk_A$, if so, then stores $Cert_A$ and $id_A$ in her smartphone. Otherwise, she aborts. Note that the users who are diagnosed as COVID-19 positive will no longer update their public keys.

In addition, each doctor $\mathcal{D}$ has an additional group signature user secret key $gsk$ (i.e., one of vector components from the $gsk$ of group signature) from his/her affiliation (where the hospital manager plays the role of the group manager regarding to the group signature). Each hospital will publish the group signature group public key $gpk$.

**Meeting phase** If Alice is confirmed negative, Alice will use Bluetooth to periodically broadcast her package $\mathcal{H}_A = \mathcal{H}(\text{Status}. id_A. A.Cert_A)$ to surrounding people in a regular time interval. Otherwise, $(\text{Status}. id_A. P. Cert_P)$ is broadcasted around instead. If this unhashed package has been received, the receiver should report to $\mathcal{F}$ after verifying the positive patient’s $Cert_P$. If some user (e.g., Bob) frequently receives the same hash package from Alice within a certain time frame, Alice and Bob are considered as close contact. Now we describe the following execution (as shown in Fig. 5) between Alice and Bob, so that Alice will identify Bob as her close contact. At the end of the process, Bob will also record Alice as his close contact.

1. Bob requests Alice to send him the package $(id_A. A. Cert_A)$. Bob then computes $\mathcal{H}_A = \mathcal{H}(\text{Status}. id_A. A.Cert_A)$, if $\mathcal{H}_A = \mathcal{H}_A$, he randomly chooses a challenge number $R_B \in \mathbb{Z}_q$ and sends to Alice. 
2. Alice uses her private key $a$ to generate a Schnorr signature $\sigma'_A$ on $R_B$ as the following steps. Then sends it to Bob for verification.
   a) Randomly selects $k \in \mathbb{Z}_q$.
   b) Computes $t = \mathcal{H}(pk_A. R_B)$.
   c) Computes $s = k - at mod q$.
   d) Outputs the signature $\sigma'_A = (s, t)$.
3. Bob uses the $pk_A$ to verify Alice’s public key $A$.

$$\text{Verify}(pk_A, Cert_A, (id_A, A. Date))$$

If it is valid, Bob will check Alice’s Schnorr’s signature $\sigma'_A = (s, t)$ by computing whether $t = \mathcal{H}(pA'. R_B)$. If so, Bob will store Alice’s package $(id_A, A. Cert_A)$ locally in his smartphone. Otherwise, he aborts. In addition, Bob does the same process as above to let Alice check whether his package is valid. If so, Alice will store Bob’s package $(id_B, B. Cert_B)$ in her smartphone.

Now Alice and Bob will step into a mutual commitment phase for generating a proof to prove they are each other’s close contact in a zero-knowledge way. First, they need to store the opposite side’s identification information. Later on, they generate a zero-knowledge proof to $\mathcal{F}$ as a close contact to a confirmed patient (if Alice or Bob is diagnosed as COVID-19 positive). Alice and Bob jointly perform the following steps.
gating the signatures of Bob and Charlie is as follows:

\[ D \]

space, but also improves the signature verification speed. Signature scheme) into a short signature, which not only saves storage commitments (note that the commitment here is actually generated by a multi-signature with public-key aggregation [35] to aggregate multiple separate commitment packages would result in a large number of commitments.)

\[ 1. \text{Bob uses his private key} \ b \in \mathbb{Z}_q \text{ and Alice’s} \ i_d_A \text{ to generate} \ \sigma_B = \mathcal{H}_1(i_d_A)^b \text{ and sends it to Alice. Alice checks whether} \ e(p, \sigma_A) = e(B, \mathcal{H}_1(i_d_A)) \text{ is satisfied, If so, } \{B, i_d_A, \sigma_A, \text{Date}\} \text{ is stored in Alice’s smartphone.} \]

\[ 2. \text{On the opposite side, Alice uses her private key} \ a \in \mathbb{Z}_q \text{ and Bob’s} \ i_d_B \text{ to generate} \ \sigma_A = \mathcal{H}_1(i_d_B)^a \text{ and sends it to Bob. Bob checks whether} \ e(p, \sigma_A) = e(A, \mathcal{H}_1(i_d_B)) \text{ is satisfied. If so, } \{A, i_d_A, \sigma_A, \text{Date}\} \text{ is stored in Bob’s smartphone.} \]

After completing the above interaction process, Alice will receive a large number of mutual commitments from her close contacts, therefore, these data would occupy her very huge local storage. More seriously, these separate commitment packages would result in a large number of zero-knowledge proof executions. For addressing this issue, we adopt a multi-signature with public-key aggregation [35] to aggregate multiple commitments (note that the commitment here is actually generated by a signature scheme) into a short signature, which not only saves storage space, but also improves the signature verification speed.

Suppose Alice is diagnosed as a COVID-19 positive patient by doctor \( \mathcal{D} \), Bob and Charlie are her close contacts. The process of Alice aggregating the signatures of Bob and Charlie is as follows:

\[ 1. \text{Alice takes as input the public keys} \ (B, C) \text{ of Bob and Charlie into a hash function} \ \mathcal{H}_2 : \mathbb{G}_2^* \to \mathbb{R}^n, \text{ where} \ R = \{1, 2, \ldots, 2^{128}\} \text{ and} \ 1 \leq n \in \mathbb{N}, \text{ to generate} \ (t_B, t_C) \mapsto \mathcal{H}_2(B, C) \in \mathbb{R}^n. \]

2. Then computes and outputs a short aggregate signature \( \sigma_B = \sigma_A = \mathcal{H}_1(i_d_A)^b \in \mathbb{G}_1 \) by taking \((t_B, t_C, \sigma_B, \sigma_C)\) as input.

Note that the verification process of this aggregate signature is as follows:

1. Takes the input \((B, C)\) to caculate \((t_B, t_C)\mapsto \mathcal{H}_2(B, C) \in \mathbb{R}^n. \)
2. Computes the aggregate public key \( \text{apk} = B^aC^a \in \mathbb{G}_2 \) with the input \((B, C, t_B, t_C)\).
3. If \( e(p, \sigma) = e(\text{apk}, \mathcal{H}_1(i_d_A)) \) is satisfied, then accept. Otherwise, rejects.

Medical treatment phase We assume Alice is a confirmed patient by a medical doctor \( \mathcal{D} \), but her status is not updated yet on the blockchain. Bob and Charlie are Alice’s all close contacts, Alice has the obligation to report to the doctor. In order to prove to \( \mathcal{D} \) that Bob and Charlie are her close contacts without revealing the identities of two close contacts, Alice needs to create pseudo-identity for Bob and Charlie, respectively, which would be sent to the medical doctor for zero-knowledge proof. \( \mathcal{D} \) publishes these pseudo-identities on the blockchain, all apps users could check on chain whether they are referred to in the updated entries everyday. In Fig. 6, we show the interaction of this medical treatment phase.

Now we illustrate this process in detail. After the doctor verifies Alice’s identity and obtains the \( i_d_A \), Alice generates pseudo-identities \( \hat{B} \) and \( \hat{C} \) for Bob and Charlie, respectively. That is, Alice randomly chooses \( x \in \mathbb{Z}_q \), then computes

\[ h = p^x, \hat{B} = (B^a)^x = (p^x)^a = h^a, \hat{C} = (C^a)^x = (p^x)^a = h^c \]

and sends \((h, \hat{B}, \hat{C})\) to \( \mathcal{D} \).

In order to prove to \( \mathcal{D} \) that \((h, \hat{B}, \hat{C})\) has a correct form as above. Alice needs to generate a zero-knowledge proof regarding to this form:

\[ \begin{align*}
& PK\{(\sigma, B^a, C^a, x) : h = p^x, \hat{B} = (B^a)^x, \hat{C} = (C^a)^x, \\
& e(\sigma, p) = e(\mathcal{H}_1(i_d_A), B^aC^a)\}
\end{align*} \]

where \( \sigma = \sigma_B \sigma_C \). The above Eq. (3) is equivalent to the following Eq. (4):

\[ \begin{align*}
& PK\{(\sigma, B^a, C^a, x) : h = p^x, \hat{B} = (B^a)^x, \hat{C} = (C^a)^x, \\
& e(\sigma, h) = e(\mathcal{H}_1(i_d_A), \hat{B} \hat{C})\}
\end{align*} \]

To instantiate this proof, Alice randomly generates \( s_1, s_2, t_1 \in \mathbb{Z}_q \) and computes

\[ E_1 = p_1^x p_1^x, \quad E_2 = B^aC^a p_1^x, \quad F = \sigma_1 \]

Alice will send \( E_1, E_2, F \) to \( \mathcal{D} \) and prove:

\[ \begin{align*}
& PK\{(s_1, s_2, t_1, \alpha_1, \alpha_2, x) : E_1 = p_1^x p_1^x \land E_1 = p_1^x p_1^x \land h = p^x \\
& \land \hat{B} \hat{C} = E_2 p_1^x \land e(F a_1, h) = e(\mathcal{H}_1(i_d_A), \hat{B} \hat{C})\}
\end{align*} \]
The above proof can be turned into a non-interactive type by running the following steps:

**Proof of generation**

1. Randomly selects \( r_1, r_2, r_3, r_4, r_5, r_6 \in \mathbb{Z}_q \)
2. Computes
   \[
   T_1 = p_1^{r_1} p_2^{r_2}, \quad T_3 = E_{x}^{r_3} p_1^{r_1} p_2^{r_2}, \quad T_4 = E_{x}^{r_4} p_1^{r_1}, \quad T_5 = e(u_{1}, h)^{r_5}
   \]
3. Computes the hash
   \[
   H = \pi(T_1, T_2, T_3, T_4, T_5, \Bar{B}, \Bar{C}, h, E_{i}, E_{j}, F, \text{Date})
   \]
4. Computes
   \[
   z_1 = r_1 - H_{s_{1}} \mod q, \quad z_2 = r_2 - H_{s_{2}} \mod q, \quad z_3 = r_3 - H_{s_{3}} \mod q \]
   \[
   z_4 = r_4 - H_{s_{4}} \mod q, \quad z_5 = r_5 - H_{s_{5}} \mod q, \quad z_6 = r_6 - H_{s_{6}} \mod q
   \]
5. Outputs the proof \( \pi : (H, z_1, z_2, z_3, z_4, z_5, z_6, r_1, r_2, r_3, r_4, r_5, r_6, q) \).

**Proof of verification** Computes

\[
T_1 = E_{x}^{r_1} p_1^{r_2}, \quad T_2 = E_{x}^{r_3} p_1^{r_1} p_2^{r_2}, \quad T_3 = h^{r_3} p_2^{r_2}
\]
\[
T_4 = (\Bar{B} \cdot \Bar{C})^{r_4} E_{x}^{r_4} p_1^{r_3}, \quad T_5 = \left( \frac{e(F, h)}{e(\pi(id_{s_{1}}, E_{i}))} \right)^{r_5} e(u_{1}, h)^{r_6}
\]

Accept the proof if and only if the following equation is satisfied.

\[
H = \pi(T_1, T_2, T_3, T_4, T_5, \Bar{B}, \Bar{C}, h, E_{i}, E_{j}, F, \text{Date})
\]

If the above proof is accepted, then \( \mathcal{D} \) will generate a group signature \( \sigma_{D} = \text{GSig}(gsk, gpk, M) \) on message \( M = (h, \Bar{B}, \Bar{C}, \text{Date}) \). Then \( \mathcal{D} \) uploads \( (\sigma_{D}, h, \Bar{B}, \Bar{C}, \text{Date}) \) to the blockchain. At the mean time, \( \mathcal{D} \) will report (with Alice’s id_{A} and A) to \( \mathcal{D} \) that Alice is confirmed as COVID-19 positive. Then \( \mathcal{D} \) will update the entry of Alice as \( \text{Status} = 1 \) and sign it every day (update Date only) until Alice is diagnosed as COVID-19 negative.

**Tracing phase** At the end of each day, every non-infected COVID-19 user can scan each entry of blockchain (as shown in Fig. 7) to check whether he/she is referred to. For example, if Bob finds his pseudo-identity \( \Bar{B} \) included in an entry \( (\sigma_{D}, h, \Bar{B}, \Bar{C}, \text{Date}) \). Then he uses his private key \( b \) corresponding to that Date to check whether \( \Bar{B} = h^{b_{u}} \) or not. If so, Bob runs the verification algorithm of the group signature \( \mathcal{G} \mathcal{V} \mathcal{F}(\sigma_{D}, gpk, (h, \Bar{B}, \Bar{C}, \text{Date})) \). If it is valid, Bob determines that he is a close contact of a confirmed patient at that Date.

5. Security analysis

In this section, we will analyse the security of our proposed solution in the threat model defined in the Section 3. Our solution not only guarantees the privacy of both confirmed patient and close contacts, but also prevents from the false positive report case. The security our proposal can provide is as good as that of Liu et al. [33].

- **The security of confirmed patient**: In our scheme, government and doctor are both trusted entities. There are two types of adversaries, i.e., the general public and the close contacts of a confirmed patient. The general public already knows what has been posted in the blockchain, and the close contact of a confirmed patient would like to know which particular close contact is a confirmed patient.

  For the general public, it is impossible to obtain information about the confirmed person from the blockchain. Now we suppose the information published in the blockchain is \( (\sigma_{D}, h, \Bar{B}, \Bar{C}, \text{Date}) \), where \( \sigma_{D} \) is the group signature signed by the doctor on message \( (h, \Bar{B}, \Bar{C}, \text{Date}) \). Now we look into \( h = p^{x} \), where \( p \) is the generator of \( \mathbb{Z}_q \) as a public parameter, and \( x \) is randomly selected from \( \mathbb{Z}_q \) by Alice. It is hard to compute \( x \) from \( p \) and \( h \) relying on the discrete logarithm assumption. In addition, \( \Bar{B}, \Bar{C} \) is generated by the public keys \( (B, C) \) of Bob and Charlie, all these information do not contain anything about Alice (i.e., a confirmed patient).

  For the close contact of a confirmed person, it is impossible to distinguish who is actually a confirmed patient from all his close contacts. In our scenario, Bob is a close contact of Alice. We can use a reduction proof that Bob is supposed to be able to tell who is the patient, therefore resulting in a contradiction that he can output a forged signature. We assume that Bob has \( n \) close contacts on the Date, including Alice (i.e., a confirmed patient). We suppose that the
The security of contacts

The precaution of false positive reports

\[ \hat{B} = h^\hat{h}_B \]

\[ GVf(\sigma_D, gpk, (h, \hat{B}, \hat{C}, Date)) \]

![Fig. 7. Tracing phase.](image)

probability Bob can correctly guess Alice is the confirmed patient is \(\rho\). If we have \(\rho > 1/n + \epsilon\), where \(\epsilon\) is negligible probability. Then Bob can distinguish the patient from other users with non-negligible probability. If he would like to distinguish the patient, he has to make use of the tuple \((\sigma_B, h, \hat{B}, \hat{C}, Date)\) published on blockchain. Since \((\sigma_B, h, \hat{C}, Date)\) cannot give any information to distinguish. The only element can be useful is \(B = B^{\alpha_B}\), where \(B\) is given by Bob to his close contact in the meeting phase. To distinguish each close contact among all, Bob has to give different \(B\) to each different close contact.

For the completeness of the protocol, the verification algorithm shown in Eq. (1) should be run for each contact in the execution process. Now we assume Bob has \(n = 2\) contacts, therefore, two valid signatures \(\sigma_B\) on \(B\) and \(\sigma_B\) on \(B\) must exist. However, Bob can only obtain one single signature from the government \(\mathcal{G}\) each day. Thus, either \(\sigma_B\) or \(\sigma_B\) has to be output by Bob as the forged one.

- **The security of contacts:** The information from the confirmed patient Alice about her close contacts Bob and Charlie are the zero-knowledge proof of Eq. (3) and the tuple \((h, \hat{B}, \hat{C})\). If there exists an adversary who can identify Bob and Charlie, one could use standard game-hopping technique to change the tuples from \((h, \hat{B}, \hat{C})\) to \((h, \hat{R}_B, \hat{R}_C)\) where \(\hat{R}_B\) and \(\hat{R}_C\) are random group elements. These two tuples are indistinguishable by the advantage of breaking the Decisional Diffie–Hellman (DDH) assumption. Since now the zero-knowledge simulator is using a fake tuple \((h, \hat{R}_B, \hat{R}_C)\), the hoppings between two settings also require that the zero-knowledge proof for Eq. (3) is simulation-sound. The Σ-Protocol we use is indeed simulation-sound. If the proof for Eq. (3) is zero-knowledge, any efficient adversary cannot learn the information regarding to Bob and Charlie.

Now it turns to argue that the instantiation for the proof of Eq. (5) is indeed Honest-Verifier Zero-Knowledge (HVZK). We can see the three moves protocol itself is HVZK and the auxiliary \(E_1, E_2, F\) do not reveal any additional information. For any possible witnesses \((B, C, \sigma)\), there exists a unique randomness \((s_2, t_1)\) such that \(E_1 = B^{\alpha_B}C^s_2p_1^{t_1}\), \(F = \sigma_B\alpha_BC^s_2t_1\). For each possible \(s_2\), there exists a unique randomness \(s_1\) such that \(E_1 = p_1^{s_1}p_2^{s_2}\). The simulator can choose random values \(E_1, E_2, F\) and use the zero-knowledge simulator to simulate proof for Eq. (5). Since these random values also have the correct form. Therefore, it is hard to distinguish the simulated proof from the real proof, which means that these values cannot leak any information regarding to the close contacts Bob and Charlie.

- **The precaution of false positive reports:** Our solution can prevent from two types of false positive cases to guarantee that: (1) Nobody cannot maliciously pretend to be a confirmed patient to convince his/her all contacts that they are close contacts of a confirmed patient who are actually not. (2) Any confirmed patients cannot maliciously claim that he/she has been contact with a user who is actually not.

Case 1: We assume the adversary is an unconfirmed user (e.g., Peter in Fig. 2) who wants to play as a confirmed patient and convinces his close contact (e.g., John in Fig. 2) that he (e.g., John) is actually a close contact of some confirmed patient (e.g., Peter). Observe the execution of medical treatment phase that a user (e.g., John) believes he is a close contact of some confirmed patient, only if he downloads the entry which includes his pseudo-identity (e.g., \(\sigma_B, h, \hat{J}, \ldots, Date\)), where \(\hat{J}\) is referred to John’s pseudo-identity) from blockchain and checks that: (1) \(\sigma_B\) is a valid group signature; (2) the user’s pseudo-identity (e.g., \(\hat{J} = h^{\hat{h}_J}\), where \(\hat{J}\) is referred to John’s pseudo-identity) is correct. Therefore, the adversary (e.g., Peter) has to play the role of a confirmed patient (e.g., like as Alice) to compute the pseudo-identity for his target contact (e.g., John) as in Eq. (2), since the attacker knows the public key of this close contact (e.g., John). However, if the adversary (e.g., Peter) is successful, he has to forge a valid group signature \(\sigma_B\), which contradicts to the unforgeability of the underlying group signature scheme.

Case 2: We assume the adversary (e.g., Alice) is a confirmed patient who wants to convince the doctor that a certain user (i.e., John) is his/her close contact but who is actually not. Let \(J\) denote John’s public key, Alice has to produce a valid proof for Eq. (3). If the proof is sound, there exists an extractor to extract witnesses \((s, B^{\alpha_B}, C, x)\) such that \(s = s_B^\alpha s_C\hat{h}\) is the aggregate signature on \(\hat{i}_d\) under public keys \((B, C)\) for some \(x\) such that \((B = h^{\alpha_B}, C = h^{\alpha_C})\), and \((B = p_1^{s_B}, C = p_2^{s_C})\). For John to think he has been in close contact, \(\hat{B}\) or \(\hat{C}\) has to satisfy the relation that \(DL_{p_1}(J) = DL_{p_2}(B)\) or \(DL_{p_2}(J) = DL_{p_1}(C)\). By the soundness of the proof, this requires \(J = B\) or \(J = C\). If Alice has never been in close contact with John, it means Alice has to forge a signature \(\sigma_J\) from John on \(\hat{i}_d\). This is impossible since the underlying signature of the aggregate signature [35] is unforgeable against chosen message attack.

It remains to argue that the soundness for the proof of Eq. (5) indeed implies that of Eq. (3). If the proof for Eq. (3) is sound, one could extracting witness \((s_1, s_2, t_1, a_1, a_2, x)\). Moreover, since \(E_1 = p_1^{s_1}p_2^{s_2}\) and \(E_1 = p_1^{s_1}p_2^{s_2}\), we have \(a_1 = a_2x\). By \(J = E_2p_1^{x_3}\), we have \(J = (E_2p_1^{x_3})^x\). From \(e(Fu_1^{x_3}h) = e(\hat{\mathcal{H}}_j(id_0)^x, \hat{\mathcal{H}}_j(id_0^x))\), and \(h = p_1^x\), we have \(e(Fu_1^{x_3}h) = e(\hat{\mathcal{H}}_j(id_0^x))\). In other words, \(Fu_1^{x_3}\) is a valid signature on \(\hat{i}_d\) under public key \((E_2p_1^{x_3})\). Furthermore, since \(h = p_1^x\), \(DL_{p_2}(E_2p_1^{x_3}) = DL_{p_1}(E_2p_1^{x_3}) = DL_{p_1}(\hat{J})\). One could output witness \((Fu_1^{x_3}, E_2p_1^{x_3}, x)\) as the witness of the proof for Eq. (3).
6. Performance evaluation

In this section, we perform an experiment by respectively timing the execution of signature verification with and without the utilization of aggregate signature, to evaluate the efficiency and feasibility of our proposal when compared to the work of Liu et al. [33]. The instantiation of signature scheme in the test is based on elliptic curve cryptography, using a 342-bit elliptic curve and SHA-256 hash function. In Table 1, we show the size of each entry of data used in this evaluation, including public key, private key, ID, hash value, and signature.

Our simulation is implemented using JavaScript. The simulation experiment is configured within an i5-4590U processor, 16GB memory, and Windows 10 64-bit operating system. The code running environment is node.js (called PBC library for running calculation). We count the time cost under different cryptographic operations (as shown in Table 2), including addition, multiplication, subtraction, and mapping time. The result shows that the computational cost of our scheme is feasible even in a relatively large number of contacts scenario.

In our proposal, a multi-signature with public-key aggregation [35] is the main cryptographic tool to effectively improve the signature verification speed (as shown in Fig. 8). Multi-signature with public-key aggregation used in this paper is actually a variant of BLS signature [34]. Therefore, we abuse to use BLS in the following figure to denote the similar variant signature we adopt in this work. And we let elliptic curve digital signature algorithm (ECDSA) [38] to denote the signature used in the work of Liu et al. [33]. As the number of contacts increases, the signature verification speed using BLS is much better than that of ECDSA (without using aggregation technique). We can see from Fig. 8 that the time cost of signature verification in our scheme is growing very gently with a low rate. In addition, the work of Liu et al. [33] has to verify all the collected signatures from app users, our scheme only needs to verify the aggregate signature once, further improves the speed of signature verification and reduces the corresponding communication and computation costs. Moreover, the size of the aggregate signature generated from all contacts of a confirmed patient is equivalent to the size of the signature of one single contact, therefore, saving the storage space on chain.

To sum up, our solution possesses a better performance than [33] in the aspect of efficiency side and greatly enhances the overall efficiency of that tracing framework proposed by Liu et al. [33].

7. Conclusion

In this paper, an efficient privacy-preserving solution based on blockchain technology is proposed for the COVID-19 contact tracing scenario. It utilizes some cryptographic tools (mainly including a typical digital signature, a multi-signature with public-key aggregation, a group signature and a non-interactive zero-knowledge proof) to protect app user’s privacy. The main contribution of our work is the use of aggregate signature to accelerate the speed of signature verification on chain, and further reduces the overhead regarding to computation, communication and storage. Our scheme is relying on the discrete logarithm assumption and secure against the semi-honest adversaries. At the end of this article, we present an experimental comparison to show our solution can achieve a better performance than the work of Liu et al. [33].

In addition, due to the extremely contagious feature of the COVID-19 virus, the user’s geographic location data should be factored into the approach for more precisely contact tracing. However, Liu et al. [33] and ours both do not consider any location data into the tracing framework and can only achieve passive security in the semi-honest model. Inspired by the work of Alshahi et al. [39], where edge servers are utilized to assist medical authorities for contact tracing in the malicious model, we hope to take the public and private location (i.e., businesses, houses, and offices) data into tracing approach and achieve a much better tracing solution with higher-level security in the future.

| Table 1 | Size of the data used in our scheme. |
|---------|-------------------------------------|
| Data    | Size (bytes)                        |
| Public key | 48                                |
| Private key | 32                                |
| ID              | 5                                  |
| Hash            | 43                                  |
| Signature       | 96                                  |

| Table 2 | Computation times of cryptographic operations. |
|---------|-----------------------------------------------|
| Cryptographic operations | Time (ms) |
| Add                | 0.0002                          |
| Mul               | 0.0013                           |
| Sub               | 0.0002                           |
| Pairing           | 16.863                          |
| Hash to curve     | 8.5325                          |

![Fig. 8. Comparison of verification time consumption between using BLS and ECDSA.](image)

CRediT authorship contribution statement

Momeng Liu: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. Zeyu Zhang: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. Wenqiang Chai: Software, Validation, Data curation. Baocang Wang: Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61902303, U19B2021, 61972457), the Natural Science Basic Research Program of Shaanxi (Program no. 2020JQ-832), the Key Research and Development Program of Shaanxi (Grant no. 2020ZDLGY08-04), the Scientific Research Program Funded by Shaanxi Provincial Education Department (Program no. 21JK0651), the Young Talent fund of University Association for Science and Technology in Shaanxi, China (Program no. 20210116), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province, the Shaanxi

| Table 1 | Size of the data used in our scheme. |
|---------|-------------------------------------|
| Data    | Size (bytes)                        |
| Public key | 48                                |
| Private key | 32                                |
| ID              | 5                                  |
| Hash            | 43                                  |
| Signature       | 96                                  |

| Table 2 | Computation times of cryptographic operations. |
|---------|-----------------------------------------------|
| Cryptographic operations | Time (ms) |
| Add                | 0.0002                          |
| Mul               | 0.0013                           |
| Sub               | 0.0002                           |
| Pairing           | 16.863                          |
| Hash to curve     | 8.5325                          |

![Fig. 8. Comparison of verification time consumption between using BLS and ECDSA.](image)

CRediT authorship contribution statement

Momeng Liu: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. Zeyu Zhang: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. Wenqiang Chai: Software, Validation, Data curation. Baocang Wang: Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61902303, U19B2021, 61972457), the Natural Science Basic Research Program of Shaanxi (Program no. 2020JQ-832), the Key Research and Development Program of Shaanxi (Grant no. 2020ZDLGY08-04), the Scientific Research Program Funded by Shaanxi Provincial Education Department (Program no. 21JK0651), the Young Talent fund of University Association for Science and Technology in Shaanxi, China (Program no. 20210116), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province, the Shaanxi
Key Laboratory of Blockchain and Secure Computing (Program no. N-KY-XZ-1101-202110-7349), and the Research Foundation of Xi’an Polytechnic University for the Doctoral Scholars (Program no. BS201848).

References

[1] D. Cucinotta, M. Vanelli, WHO declares COVID-19 a pandemic, Acta BioMed. 91 (1) (2020) 157.
[2] S. Vaudeny, Centralized or decentralized? The contact tracing dilemma, IACR Cryptol. ePrint Arch. 2020 (531) (2020),https://eprint.iacr.org/2020/531.
[3] J. Wen, Q. Zhao, Z. Lin, et al., A study of the privacy of COVID-19 contact tracing apps, Proc. Secure Commun. 2020 (2020) 297–317.
[4] F. Farrell, Experts raise concerns about security of coronavirus tracing app COVIDSafe, AICB News (2020), https://www.abs.net.au/news/2020-05-14/experts-concerned-about-coronavirus-tracing-apps-covid-safe.
[5] N. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, Decent. Bus. Rev. (2008) 21260. https://bitcoin.org/bitcoin.pdf.
[6] Y. Li, G. Yang, W. Sunilo, et al., Traceable monero: anonymous cryptocurrency with enhanced accountability, IEEE Trans. Dependable Secure Comput. 18 (2) (2021) 679–691.
[7] R. Chen, Y. Li, Y. Yu, et al., Blockchain-based dynamic provable data possession for smart cities, IEEE Internet Things J. 7 (5) (2020) 4143–4154.
[8] Y. Li, Y. Yu, R. Chen, et al., Integritychain: provable data possession for decentralized storage, IEEE J. Sel. Areas Commun. 38 (6) (2020) 1205–1217.
[9] Y. Yu, Y. Li, J. Tian, et al., Blockchain-based solutions to security and privacy issues in the internet of things, IEEE Wirel. Commun. 25 (6) (2018) 12–18.
[10] Y. Li, W. Sunilo, G. Yang, et al., A blockchain-based self-tallying voting protocol in decentralized IoT, IEEE Trans. Dependable Secure Comput. 19 (1) (2020) 119–130.
[11] B. Bera, A. Vangala, A.K. Das, et al., Private blockchain-enhanced drones-assisted authentication scheme in IoT-enabled agricultural environment, Comput. Stand. Interfaces 80 (2022) 103567.
[12] P. Wang, W. Sunilo, Data security storage model of the internet of things based on blockchain, Comput. Syst. Sci. Eng. 36 (1) (2021) 213–224.
[13] X. Zheng, Y. Zhao, H. Li, et al., Blockchain-based verifiable privacy-preserving data classification protocol for medical data, Comput. Stand. Interfaces 82 (2022) 103655.
[14] D. Li, D. Han, Z. Zheng, et al., MOOCchain: a blockchain-based secure storage and sharing scheme for MOOCs learning, Comput. Stand. Interfaces 81 (2022) 103597.
[15] Y. Li, W. Sunilo, G. Yang, et al., Non-equivocation in blockchain: double-authentication-preventing signatures gone contractual, Proc. ASIACCS 2021 (2021) 859–871.
[16] H. Yuan, X. Chen, J. Wang, et al., Blockchain-based public auditing and secure deduplication with fair arbitration, Inf. Sci. 541 (2020) 409–425.
[17] G. Tian, Y. Hu, J. Wei, et al., Blockchain-based secure deduplication and shared auditing in decentralized storage, IEEE Trans. Dependable Secure Comput. (2021).
[18] Y. Yu, Y. Zhao, Y. Li, et al., Blockchain-based anonymous authentication with selective revocation for smart industrial applications, IEEE Trans. Ind. Inform. 16 (5) (2019) 3290–3300.
[19] T.K. Khuc, T.N. Nguyen, H.Q. Le, et al., Efficient unique ring signature for blockchain privacy protection, Proc. ACISP 2021 (2021) 391–407.
[20] H. Cho, D. Ippolito, Y.W. Yu, Contact tracing mobile apps for COVID-19: privacy considerations and related trade-offs, 2020, https://doi.org/10.48550/arXiv.2003.13511.
[21] D. Pervin, J. Lee, J. Yeo, J. Kek, A. Tan, et al., Bluetrace: a privacy-preserving protocol for community-driven contact tracing across borders, Gov. Technol. Agency-Singapore 18 (2020). https://bluetrace.io/static/bluetrace_whitepaper-938063656596c10462abe83eb13bc.pdf.
[22] R. Thomas, Z.A. Michaeloff, H. Greenwood, et al., More than privacy: Australians’ concerns and misconceptions about the COVIDSafe app, medRxiv (2020).
[23] R. Aviabate, V. Botta, V. Lovino, et al., Towards defeating mass surveillance and SARS-CoV-2: the pronto-c2 fully decentralized automatic contact tracing system, IACR Cryptol. ePrint Arch. 2020 (493) (2020),https://eprint.iacr.org/2020/493.
[24] J.K. Liu, M.H. Au, T.H. Yuen, et al., Privacy-preserving COVID-19 contact tracing app: a zero-knowledge proof approach, IACR Cryptol. ePrint Arch. 2020 (528) (2020),https://eprint.iacr.org/2020/528.
[25] D. Boneh, C. Gentry, B. Lynn, et al., Aggregate and verifiably encrypted signatures from bilinear maps, Proc. EUROCRYPT 2003 (2003) 416–432.
[26] J. Song, T. Gu, Z. Fang, et al., Blockchain meets COVID-19: a framework for contact information sharing and risk notification system, Proc. MASS 2021 (2021) 269–277.
[27] H. Xu, L. Zhang, O. Onireti, et al., Beetrace: blockchain-enabled privacy-preserving contact tracing for COVID-19 pandemic and beyond, IEEE Internet Things J. 8 (5) (2020) 3915–3929.

M. Liu et al. Computer Standards & Interfaces 83 (2023) 103643

Key Laboratory of Blockchain and Secure Computing (Program no. N-KY-XZ-1101-202110-7349), and the Research Foundation of Xi’an Polytechnic University for the Doctoral Scholars (Program no. BS201848).