Title	Efficient Multi-Party Contact Tracing
Author(s)	De Goyon, Mathieu; Miyaji, Atsuko; Tian, Yangguang
Citation	2021 9th International Symposium on Computing and Networking, CANDAR 2021. 2021, p. 10-18
Version Type	AM
URL	https://hdl.handle.net/11094/89663
rights	© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Note	

Efficient Multi-Party Contact Tracing

Mathieu de Goyon[2,a)]   Atsuko Miyaji[1,2,b)]   Yangguang Tian[3,c)]

,

Abstract: Since the beginning of the Covid-19 Pandemic, Contact Tracing Apps have been implemented in many countries as a way to detect if someone has been in contact with a patient within a minimum amount of time. However, most existing solutions only consider users in pairs. Since many people meet at the same time in real-life scenarios, those applications aren't able to accurately reflect the situation. Moreover, extending current schemes to a multi-party setting could cause scaling problems and place a heavier load on the device. In this paper, we propose a new Contact Tracing protocol that works in a multi-party setting. We evaluate our scheme to show its efficiency.

Keywords: Contact Tracing, System Security, Multi-signature, Ring Signature

1. Introduction

During the Covid-19 Pandemic, monitoring the spread of the virus quickly became an efficient tool for reducing the number of contaminations. By detecting if someone has been in contact with a Covid-19 patient, that person could be warned in advance and self-isolate before contaminating other people. However, doing so manually is very difficult. Contact Tracing apps were introduced by researchers as a tool to perform this task automatically and were adopted by many countries. Contact Tracing apps are able to communicate with other users and store who has been in contact. Once someone is tested positive for the Covid-19, the app will be able to use the information stored to send an alert to contacts of the patient before they have the time to contaminate other people. Contact Tracing apps become more efficient as more people use them. However, many people are hesitant to use them because they are worried for their privacy[1]. Indeed, many people are reluctant to use Contact Tracing apps because they are afraid a malicious user or even governments[2] would be able to track their movements and learn with whom they have been in contact. Contact Tracing apps must guarantee their users' privacy so that many users will make use of them and generate a favorable impact. Another issue is that those apps must be used continuously to be accurate while the user is outside of his home. As a result, users must carry their smartphones constantly with them the whole time which can also be a load on the device. The app needs to be efficient and limit the number of computations and required communications[3].

Close contact is the name given to someone who has been near a positive patient for a long enough time for there to be a risk of infection. When three or more people are close contacts, every Contact Tracing app currently used by countries will consider them in pairs. Let Alice, Bob, and Charlie be three users of a Contact Tracing app, staying close to each other for long enough for the app to recognize them as close contacts. From the viewpoint of Alice's app, the app will save Bob's and Charlie's identifiers separately in the smartphone instead of together: Alice with Bob, Alice with Charlie[4][5]. As a result, the app will consider a meeting between several people as a repetition of meetings between 2 people, which does not represent the actual situation. Moreover, if the Contact Tracing app needs to verify the identity of other users, considering each user separately will significantly increase the load on the phone and make it use up its battery more quickly.

Contact Tracing apps can generally be divided into two types: decentralized contact tracing apps which use information related to the patient to send alerts to the close contacts and centralized contact tracing apps which use information about the close contacts. In 2020, Vaudenay[6] released a paper discussing the benefits and demerits of the two types while proposing possible future directions. The first centralized Bluetooth-based Contact Tracing App, TraceTogether, was released in Singapore while DP3T (Decentralized Privacy Preserving Proximity Tracing)[7] and PACT (Private Automated Contact Tracing) (East and West) which are decentralized Contact Tracing Apps were released shortly afterwards. Vaudenay[8] later improved the DP3T protocol to work against replay/relay attacks, and Pietrzak[9] proposed a non-interactive protocol version that works better with Bluetooth Low Energy. In 2020, researchers discovered that using Bluetooth as an accurate way to measure a distance could prove problematic in environments such as trains. A first paper was released by Luo et al.[10] and a second one by Meklenburg et al.[11], both using ultrasonic signals instead of Bluetooth.

Following the continuous spread of the pandemic, Apple and Google, the two leading providers of smartphone operating systems, decided to work together to release GAEN (Google and Apple Exposure Notification), a framework and protocol to facilitate digital Contact Tracing. Many countries' Contact Tracing apps are based on this protocol, such as Japan, New Zealand, Ger-

---
1   Japan Advanced Institute of Science and Technology
2   Osaka University, Osaka, Japan
3   University of Surrey, Guildford, England
a)   mathieu@cy2sec.comm.eng.osaka-u.ac.jp
b)   miyaji@comm.eng.osaka-u.ac.jp
c)   yangguang.tian@surrey.ac.uk

many, and the US. In the same year, Liu et al.[12] proposed of a centralized Privacy Preserving Protocol using a Zero-Knowledge Proof. This app allows the identity patient's close contacts to be hidden from everyone, including the government, despite using a centralized approach. In October 2020, the French Government released a new version of their Contact Tracing app, TousAnti-Covid, using a centralized system. In June 2021, Peng et al.[13] developed a decentralized Contact Tracing app using Blockchain to prevent the user from modifying the data afterwards, as well as a zero-knowledge verification scheme that ensures user privacy.

Our Contribution– We develop a new Contact Tracing that is not limited to meetings between two users but considers meeting of an entire group. Moreover, we use multi-signature schemes introduced by Drijvers et al.[14] and by Mitomi and Miyaji[15] to generate proof of the meeting which all participants will store on their phone. Finally, we use a ring signature scheme by Chow et al.[16] to send alerts to all of the close contacts of the patient while preserving the patient privacy. We will also compare the number of computations and communications with an existing scheme[12].

This paper is the full version of the paper presented at CAN-DAR2021[17]. In our preliminary work[17], we have constructed an efficient multi-party contact tracing protocol using a multi-signature in the Meeting Phase to generate proof of a meeting and a ring signature to alert the close contacts of a patient. In this journal, the following parts were added to the CANDAR2021 paper.

- Improved and redefined the threat model, by adding a Traceability Completeness property and removing the Close Contact Privacy property.
- Redefined Theorem 2 in detail to fit for the redefined Threat Model.
- Provided the precise proof of Theorems 1 and 2.
- Newly provided theorem and proof of Traceability Completeness in Theorem 3.

Our paper is organized as follows. We introduce important building blocks used for our protocol in Section 2. Then we define the system model in Section 3. We propose our multi-party setting protocol in Section 4 and present a security analysis in Section 5. We compare our protocol to several existing schemes in Section 6. Finally, we conclude our work in Section 7.

2. Building Blocks

In this section, we present the underlying building blocks, which will be used in constructing our multi-party Contact Tracing protocol. We rely on two multi-signature schemes [14], [15] to generate proof of the meeting, which will be stored in each participant's smartphone. We call the underlying multi-signature schemes DGNW-multi-signature and MM-multi-signature, respectively. The DGNW-multi-signature is based on asymmetric bilinear pairings while MM-multi-signature introduces the concepts of *message flexibility* and *order flexibility*. We note that the MM-multi-signature scheme is based on DLP, and is easily changed to ECDLP. Thus, we apply their scheme based on ECDLP.

2.1 Mathematical Notations

Here we describe the notation, which is used for the rest of our paper.

- λ: a security parameter
- i, j: users
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, cyclic groups of prime order q
- g_1, g_2, generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$: bilinear pairing
- H: $\{0, 1\}^* \to \mathbb{Z}_q$, $H_1 : \{0, 1\}^* \longrightarrow \mathbb{G}_2$ hash functions
- $x_G \in \mathbb{Z}_q^*$ the master secret key, $y_G = g_1^{x_G}$ the corresponding public key
- $h \in \mathbb{G}_1$
- p a large prime number
- n the number of participants in a meeting

We also use the following definitions[16] in the ring signature scheme.

Definition 1 Given a generator g of a group $\mathbb{G}$ and a 3-tuple (g^a, g^b, g^c), the Decisional Diffie-Hellman problem (DDHP) is to decide if $c = ab$.

Definition 2 Given a generator g of a group $\mathbb{G}$ and a 2-tuple (g^a, g^b), the Computational Diffie-Hellman problem (CDHP) is to compute g^{ab}.

Definition 3 We define $\mathbb{G}$ as a Gap Diffie-Hellman (GDH) group if no algorithm can solve CDHP with non-neglibible advantage within polynomial time even when the DDH oracle is accessible by the algorithm.

2.2 Multi-Signature

First we review DGNW-multi-signature[14]. DGNW-multi-signature is a pairing-based, forward-secure multi-signature optimized for use in blockchains. It has relatively low bandwidth, storage, and verification requirements. Verification can be achieved with three pairings, one exponentiation and $(n - 1)$ multiplications. In our scheme, we use those low requirements to reduce the number of computations in the Meeting Phase and make it as efficient as possible. We also use the necessary computations as part of the verification process. Note that the forward security described in [14] is beyond the scope of this work.

Before explaining the multi-signature, we start with the signature scheme on which it is based on. Let x be a user secret key, y be the user's public key such that $y = g_2^x$ and, r fresh randomness used for signing. The signature σ on a message M $\in \mathbb{Z}_q$ is as follows: $\sigma = (\sigma', \sigma'') = (h^x \cdot H(M)^r, g_2^r)$. On the other hand, Verification is done as follows: $e(\sigma', g_2) = e(h, y) \cdot e(H(M), \sigma'')$.

Now we consider n different users of public keys y_i, secret keys x_i and fresh randomness r_i, $\forall i \leq n$. All of these will compute the signature $\sigma_i = (h^{x_i} \cdot H(M)^{r_i}, g_2^{r_i})$ on the same message M. The multi-signature Σ can be computed as follows: $\Sigma = (\Sigma', \Sigma'') = (h^{x_1 + \cdots + x_N} \cdot H(M)^{r_1 + \cdots + r_N}, g_2^{r_1 + \cdots + r_N})$.

Verification can be done by computing a compressed version of all of the n-signer public keys: $Y = y_1 \cdots y_N$. We then use the verification equation defined previously on the compressed public key: $e(\Sigma', g_2) = e(h, Y) \cdot e(H(M), \Sigma'')$.

Next, we will review MM-multi-signature[15]. MM-multi-signature stands out compared to other multi-signature schemes by achieving two useful properties: *message flexibility* and *or-*

der flexibility. In MM-multi-signature, the message on which the multi-signature is generated is constructed progressively by each signer so it doesn't have to be fixed beforehand. Furthermore, each signer will modify the message and update the multi-signature before sending it to the next signer, but the order itself has no impact on the multi-signature. As a result, the order doesn't have to be fixed beforehand.

We use the following notations: M_1 is a message chosen by user 1. Each $M_{1,2,\cdots,i}$ denotes a message modified by user i. The difference between $M_{1,2,\cdots,i}$ and $M_{1,2,\cdots,i-1}$ which is the modification added by user i is written as $m_i = \text{Diff}(M_{1,2,\cdots,i-1}, M_{1,2,\cdots,i})$.

$Sign(x_i, m_i) = sgn_i$ and $Rec(y_i, sgn_i) = m_i$ are the signature generation and the recovery function respectively.

Before explaining the multi-signature, we start with the one-party signature. The user will generate a signature on a message m by using his secret key x, with $y = g_2^x$ his public key. The user will first generate a random number $k \in \mathbb{Z}_q$, then compute: $R = g_2^k$, $r = R + m$, and $s = (xr + 1)k^{-1}$. The signature on m is (r, s) and m can be recovered by computing $R' = g_2^{s^{-1}} y^{r.s^{-1}}$ and $m = r - R'$.

We now consider n different users. Each user generates a secret key x_i and a public key y_i with $y_i = g_2^{x_i}$ and publishes y_i with his identity information $\text{ID}_i \ \forall i \leq n$. The first user generates the signature on an original message m_1. The first user generates a random number $k_1 \in \mathbb{Z}_q$ and computes: $R_1 = g_2^{k_1}$, $r_1 = R_1 + H(m_1\|ID_1)$, and $s_1 = (x_1 r_1 + 1)k_1^{-1}$. The user then sends the message m_1 and the multi-signature (r_1, s_1) to the following signer. User i will receive $M_{1,2,\cdots,i-1}$ and modify it to $M_{1,2,\cdots,i}$. The user will then generate a signature on $m_i = Diff(M_{1,2,\cdots,i-1}, M_{1,2,\cdots,i})$. To do so, the user generates a random number $k_i \in \mathbb{Z}_q$ and computes: $R_i = g_2^{k_i}$, $r_i = R_i + H(m_i\|ID_i)r_{i-1}$, and $s_i = (x_i r_i + 1)k_i^{-1}$, where user i signature on m_i is (r_i, s_i). The multi-signature on the message $M_{1,2,\cdots,n}$ by the n users is given by $(ID_1, s_1, m_1), \cdots, (ID_{n-1}, s_{n-1}, m_{n-1}), (ID_n, s_n, r_n, m_n)$.

Verification is done as follows: For $j = n, n-1, \cdots, 2$; the verifier will compute: $R'_j = g_2^{s_j^{-1}} y_j^{r_j s_j^{-1}}$, $T_j = r_j - R'_j$, and $r_{j-1} = T_j(H(m_j\|ID_j))^{-1}$, by using the signer I_j public key y_j.

Finally, the user will compute $R'_1 = g_2^{s_1^{-1}} y_1^{r_1 s_1^{-1}}$, $T_1 = r_1 - R'_1$, and verify that $T_1 = H(m_1\|ID_1)$.

We note that in both DGNW-multi-signature and MM-multi-signature, we can apply batch verification technique[18] during the verification subphase to reduce the number of computations.

2.3   Ring Signature

We describe the ring signature scheme [16] which we use in our protocol for its ability to keep the identity of the signer anonymous.

We assume that $\mathbb{G}_1, \mathbb{G}_2$ are GDH groups. This signature scheme is an ID-based signature scheme which removes the need to check the validity of certificates and allows a spontaneous user to generate the ring signature. It is appealing because it is very efficient, only needing two pairings regardless of the group size.

Setup: The Government GV randomly chooses $x_G \in \mathbb{Z}_q^*$ as the master secret key and computes the corresponding public key $y_G = g_1^{x_G}$. Public parameters are $(G_1, G_2, e, q, g_1, y_G, H, H_1)$ as defined in Section 2.1.

KeyGen: A signer with identity $\text{ID}_i \in \{0, 1\}^*$ submits ID_i to GV. GV sets the signer's public key Q_i to be $H_1(\text{ID}_i) \in \mathbb{G}_2$, computes the signer's private signing key S_i by $S_i = Q_i^{x_G}$. Then GV sends the private signing key to the signer via a secure channel.

Sign: Let $L = \{\text{ID}_1, \text{ID}_2, \cdots, \text{ID}_n\}$ be the set of all identities of n users. The actual signer, with index s, will follow the following protocol to give the ID-based ring signature on a message M:

- $\forall i\{1, 2, \cdots, n\} \setminus s$ Choose $U_i \in \mathbb{G}_2$, compute $h_i = H(M\|L\|U_i)$.
- Choose $r'_s \in \mathbb{Z}_q^*$, compute $U_s = Q_s^{r'_s} / \prod_{i \neq s}^n U_i \cdot Q_i^{h_i}$
- Compute $h_s = H(M\|L\|U_s)$ and $V = S_s^{h_s + r'_s}$.
- Output the signature on M as $\sigma = \{\cup_{i=1}^n U_i, V\}$

Verify: A verifier can check the validity of a signature $\sigma = \{\cup_{i=1}^n U_i, V\}$ for the message M and a set of identities L as follows.

- Compute $h_i = H(M\|L\|U_i) \forall i \in \{1, 2, \cdots, n\}$.
- Check whether $e(y_G, \prod_{i=1}^n U_i \cdot Q_i^{h_i}) = e(g_1, V)$.
- Accept the signature if it is true, reject otherwise.

Correctness:

$$
\begin{aligned}
e(g_1, V) &= e(g_1, S_s^{h_s + r'_s}) \\
&= e(g_1, Q_s^{(h_s + r'_s) \cdot x_G}) \\
&= e(y_G, Q_s^{h_s} \cdot Q_s^{r_s}) \\
&= e(y_G, Q_s^{h_s} \cdot U_s \cdot \prod_{i \neq s}^n (U_i \cdot Q_i^{h_i})) \\
&= e(y_G, \prod_{i=1}^n (U_i \cdot Q_i^{h_i}))
\end{aligned}
$$

3.   Multi-party Tracing

This section describes a 2-party Contact Tracing Protocol which we will compare to our protocols in Section 6 as well as the system model used by our Contact Tracing app. As stated before, we refer to two users who have been close for a long enough time to risk infection as *close contacts*. The most common values taken are within 2 meters and for longer than 15 minutes which we will also be using[12].

3.1   Primitive Multi-Party Tracing

A Zero-knowledge proof protocol between 2 users is proposed by Liu et al. [12] which we will refer to as *Primitive Multi-Party Tracing*.

Primitive Multi-Party Tracing is a *centralized* Contact Tracing app, which means that when a patient is tested positive for the Covid-19, information related to his close contacts will be accessible to everyone. This protocol is interesting because it manages to achieve *close contact privacy* despite being centralized using a zero-knowledge proof between the doctor and the patient, which hides the identity of close contacts. We describe their scheme in detail first. Then, we apply their scheme to a multi-party setting, and compare it to our protocol in Section 6.

Their protocol is divided into four different Phases: Registration Phase, Meeting Phase, Medical Treatment Phase and Tracing Phase.

In the **Registration Phase**, a trusted entity, the Government GV will select generators $u, u_1, u_2 \in \mathbb{G}_1$ and $g, g_1, g_2 \in$

$\mathbb{G}_2$. GV will select his secret and public key $(x_G, y_G) = KeyGen(\lambda)$. GV will then publish the public parameters $\{\lambda, H, y_G, u, u_1, u_2, g, g_1, g_2\}$.

Each day, a negative user, Alice, will choose a secret key x_A and compute the associated public key $y_A = g^{x_A}$. She will then register on the government website by uploading her personal information and y_A. GV will randomly generate an identifier $\mathsf{ID}_A \in \mathbb{Z}_q$ and a signature $\sigma_A = Sign(x_G, \{''-VE'', y_A, \mathsf{ID}_A, DATE\})$, for the user and send both σ_A and ID_A to the user. Alice will verify this signature by running $Verify(y_G, \sigma_A, \{''-VE'', y_A, \mathsf{ID}_A, DATE\})$, and accept it if it is valid.

In the **Meeting Phase**, Alice will use Bluetooth to broadcast the hash $h_A = H(''-VE'', \mathsf{ID}_A, y_A, \sigma_A)$ to the surrounding periodically. For a positive user, it would be $'' + VE''$ and without hashing instead. Once a user has received enough hash within a certain time (15 minutes) Alice and the other user Bob will be considered as close contacts. Alice and Bob will go through the following protocol so that they record the information of the other party on their smartphones.

Bob will ask Alice to send him $(\mathsf{ID}_A, y_A, \sigma_A)$, and will compute $h'_A = H(''-VE'', \mathsf{ID}_A, y_A, \sigma_A)$, to verify if $h_A = h'_A$. He will abort if it is not equal. Alice will do the same. Bob will then randomly compute a challenge number $r_B \in \mathbb{Z}$ and send it to Alice. Alice will generate a Schnorr signature on this challenge number r_B as follows:

- Randomly choose $k \in \mathbb{Z}_q$
- Compute $t = H(g^k, r_B)$
- Compute $s = k - x_A \cdot t \pmod{q}$
- Output the signature $\sigma' = (s, t)$ for the challenge number r_B

Alice will send σ' to Bob for verification. Bob will check that $Verify(y_G, \sigma_A, y_A, \mathsf{ID}_A, DATE)$. If it is valid, he will verify that $t = H(g^s y_A^t, r_B)$. If it is equal, he will store Alice's package $\{\mathsf{ID}_A, y_A, \sigma_A\}$ in his app. At the same time, Alice will go through the same protocol and accept $\{\mathsf{ID}_B, y_B, \sigma_B\}$ in her app.

Alice and Bob will then go through a **mutual commitment** subphase so that they can generate a zero-knowledge proof during the medical treatment Phase. In order to do so, they will use the following protocol: Bob uses his secret key x_B and Alice identifier ID_A to generate $\sigma''_B = u^{1/(H(\mathsf{ID}_A)+x_B)}$, and sends σ''_B to Alice.

Alice will check if $e(\sigma''_B, g^{H(\mathsf{ID}_A)} y_B) = e(u, g)$. If it is equal, Alice will store $\{y_B, \mathsf{ID}_B, \sigma''_B, DATE\}$ in her app.

The opposite with Alice generating σ''_A and Bob verifying it will happen at the same time. Since this protocol isn't adapted to a multi-party setting, this process would be repeated $(n-1)$ times with each close contact in a n-party setting.

In the **medical treatment Phase**, Alice has been confirmed as a positive patient. Alice has to tell the medical doctor D about her close contacts without disclosing their identities. In order to do so, she will generate a pseudo public key of each of her close contacts (e.g Bob) together with a zero-knowledge proof from their mutual commitment to prove that they are indeed her close contacts.

Alice and D will go through the following protocol:
- D authenticates Alice by executing the verification of the Meeting Phase with Alice and obtain ID_A

- For each possible spreading day (last 14 days), Alice retrieves her close contacts.
- Alice generates the pseudo public key for Bob by randomly choosing $k \in \mathbb{Z}_q$ and computes: $h = e(u, g)^k$, $\hat{B} = e(u, y_B)^k = e(u, g^{x_B})^k = h^{x_B}$.
- Alice needs to prove to D that $(h, \hat{B})$ is correct. Conceptually, Alice needs to prove that: $h = e(u, g)^k$, $\hat{B} = e(u, y_B)^k$, and $e(\sigma''_B, g^{H(\mathsf{ID}_A)} y_B) = e(u, g)$.

In order to instantiate this proof, Alice will generate randomly $s_1, s_2, t \in \mathbb{Z}_q$ and computes: $A_1 = g_1^{s_1} g_2^{s_2}$, $A_2 = y_B \cdot g_1^{s_1}$, and $C = \sigma''_B u_1^t$.

Alice sends A_1, A_2 and C to D and proves the following system of equations $PK\{(s_1, s_2, t, \alpha_1, \alpha_2, \beta_1, \beta_2, x)\}$: $A_1 = g_1^{s_1} \cdot g_2^{s_2}$, $A_1^k = g_1^{\alpha_1} \cdot g_2^{\alpha_2}$, $A_1^t = g_1^{\beta_1} \cdot g_2^{\beta_2}$, $h = e(u, g)^k$, $\hat{B} = e(u, A_2^k \cdot g_1^{-\alpha_2})$, and $e(Cu_1^{-t}, g^{H(\mathsf{ID}_A)} A_2 g_1^{-s_2}) = e(u, g)$.

If the proof is correct, D generates a group signature $\sigma_D = GSign(USK, GPK, M_D)$ on the message $M_D = (h, \hat{B}, DATE)$ and publishes $\{\sigma_D, h, \hat{B}, DATE\}$ into the bulletin board BB. The doctor will also inform the government that Alice has been confirmed as positive and update the entry $\{''+VE'', y_A, \mathsf{ID}_A, DATE\}$ and the date every day until Alice is recovered.

In the **Tracing Phase**, Bob will scan all new entries and for each entry retrieve his secret key for that day SK_B. He will then check if $\hat{B} = h^{x_B}$. If yes, he will then run $GVerify(\sigma_D, GPK, \{h, \hat{B}, DATE\})$. If it is valid, he has in fact been in contact with a positive patient.

3.2 System Assumptions

We explain entities used in this paper as follows:
- User: A user of the Contact Tracing app on his smartphone.
- Government: It is the trusted entity responsible for the users registrations with their app, referred as GV.
- Database: Used to publish public information. Can be accessed by anyone.

We consider the following reasonable assumptions, to guarantee the functioning of the system:

- Every user has Bluetooth connectivity on his smartphone and the app is able to use it.
- Every user has Internet connectivity on his smartphone, and the app also has access to it.
- After first downloading the app, the user accepts that the app will upload his information on the database if he is tested as Covid-19 positive.
- The user has access to all information stored and communicated through the app, but he will not modify it.
- The user is unable to add incomplete information to the app (namely a list of identifiers without the corresponding multi-signature).
- A user tested positive for the Covid-19, referred as "patient" will use all information on the app while alerting his close contact with only the information on the app (i.e. all lists of identifiers will be used to compute ring signatures, and only those lists).

3.3 Threat Model

We limit ourselves to cryptographic attacks in our threat model,

which means other attacks such as networking attacks are not included. We do not make any assumptions about the users in this paper. Under those assumptions, we define the following threat model, which we will later justify in Section 5.1. We will then provide a security proof based on that threat model in Section 5.2.

Tracebility Completeness We define the concept of Traceability Completeness. Let there be a Covid-19 patient. We consider our protocol has traceability completeness if, after a correct execution of the protocol, all users considered as close contacts by the patient after the Meeting Phase are able to receive an alert during the Tracing Phase.

False Positive Case 1 We define the first case of False Positiveness. Let there be a Covid-19 patient. We consider there is False Positive Case 1 if a malicious user is able to pretend to be a close contact of that Covid-19 patient. The malicious user would then receive the alert sent to all of the close contacts without being one of them.

False Positive Case 2 Similarly, we define False Positive Case two. Let there be a negative malicious user. There is False Positive Case 2 if that malicious user is able to pretend to be a Covid-19 patient. The malicious user would then be able to trigger false alerts to all of his close contacts.

Patient Privacy We define the concept of Patient Privacy. Let there be a Covid-19 patient. The Contact Tracing app will upload information on the database related to that patient if the app is decentralized or related to his close contacts if it is centralized. We consider that there is patient privacy if no one (except the government) can know the identity of the patient based on this information (in the sense not better than guessing).

This threat model uses concepts defined in [12], but we redefine the concepts. The difference between the two Threat Models are explained in Section 5.1.

4. The Proposed Protocols

This section describes our two protocols based on DGNW-multi-signature[14] and MM-multi-signature[15]. The first proposal realizes order-flexible Multi-Party Tracing, called OF-MP Tracing. The second proposal realizes perfect-flexibility of order- and user-flexible Multi-Party Tracing, called PF-MP Tracing.

4.1 Design of Our Multi-party Tracing

We design the multi-party tracing in the following policies.

Order Flexibility (OF) We consider that the protocol has *Order Flexibility* if the order in which the users go through the Meeting Phase has no impact on the protocol. In other words, the order does not have to be fixed for the protocol to work.

User Flexibility (UF) We consider that the protocol has *User flexibility* if close contacts do not have to be fixed for the protocol to work, which corresponds to the concept of *message flexibility* defined in [15]. Since the message used in our protocol is a list of identifiers of close contacts, a multi-signature with message flexibility allows close contacts to be added flexibly during the protocol.

Multi-party proof (MP) We consider that the protocol has *Multi-party Proof* if at the end of the Meeting Phase, all par-

ticipants of the meeting will store an identical proof in their smartphones that all of them were present. This proof will differentiate a meeting between n-participants and a repetition of smaller meetings.

Our Protocols are composed of 4 different phases:

- **Registration Phase**: The users will receive the public parameters and their identifiers from the government website. The users will then generate their public and secret key pair at the beginning of each day.
- **Meeting Phase**: We assume a meeting between *n* different users staying close enough for the app to consider them as close contacts, $n > 1$ integer. The Meeting Phase itself will be divided into 2 subphases : a verification phase and a proof generation phase.
- **Alerting Phase**: We assume that one of the *n* users of the meeting has been tested positive for the Covid-19. The information of the patient will be uploaded to the database.
- **Tracing Phase**: The close contacts of the patient will be alerted that they have been in contact with a Covid-19 patient.

OF-MP Tracing and PF-MP Tracing use the same Registration, Alerting, and Tracing Phases, but some differences exist in the Meeting Phase during the proof generation subphase. We will compare the two of them in Section 6.

4.2 Registration Phase

Each user will download the app on their smartphones and register on the government website. They will receive an identifier generated by the government *ID* as well as the public parameters described in 2.1. At the start of each day, the users will generate their public and secret key pair y and x respectively, with $y = g_1^x$ and publish their public key on the government website. This pair will be kept in the user's smartphone for 14 days which corresponds to the period of transmission.

4.3 Meeting Phase

We assume a meeting between *n* users with $n \geq 2$, instead of the regular meeting between 2 users assumed in other protocols. We assume that every user is close enough to exchange information using Bluetooth. Every app is broadcasting a package to the surroundings indicating their identifier ID and their public key y, of the form $H(y, \text{ID})$. A package will be accepted if the other smartphone is within 2 meters of the user. The app uses the RSSI (Received Signal Strength Indicator) to calculate the distance between 2 smartphones. Once a user has received enough packages (for more than 15 minutes), the user will recognize the sender as a close contact. We assume that all *n* users have been together for long enough to recognize the other $(n-1)$ users as close contacts.

In the rest of this phase, we will refer to the users by their numbers i, j with $i, j \leq n$. Here, y_i and x_i are user *i*'s public and secret key respectively, $y_i = g_2^{x_i}$.

This Meeting Phase is composed of 2 subphases: a verification phase where each user will verify the identity of all of the other users and a proof generation phase where the participants will generate a multi-signature as proof that they were all close contacts.
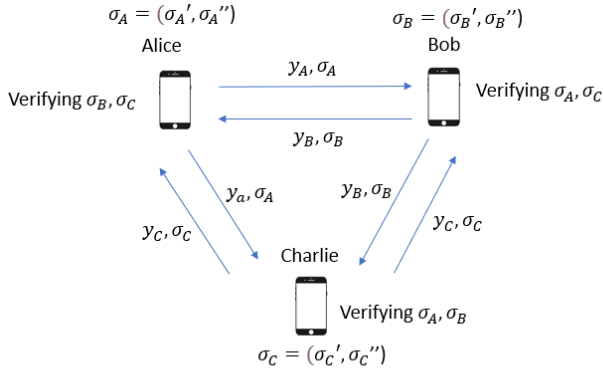
Fig. 1 Representation of the Verification Phase in case n=3

Verification - We will describe the protocol from the viewpoint of user i. Every other user will go through the exact same protocol simultaneously. We consider the message as the list of all participants IDs $M = \{ID_1, \cdots, ID_n\}$. All participants will also compute $H(M)$ and store it to reduce the computation load.

User i will generate a random number r_i, and compute the signature σ_i of the following form: $\sigma_i = (\sigma_i', \sigma_i'') = (h^{x_i} \cdot H(M)^{r_i}, g_2^{r_i})$.

User i will then broadcast (y_i, σ_i) to the other participants. Every other participant will verify the following equation: $e(\sigma_i', g_2) = e(h, y_i) \cdot e(H(M), \sigma_i'')$.

Correctness:

$$
\begin{aligned}
e(\sigma_i', g_2) &= e(h^{x_i} \cdot H(M)^{r_i}, g_2) \\
&= e(h^{x_i}, g_2) \cdot e(H(M)^{r_i}, g_2) \\
&= e(h, g_2^{x_i}) \cdot e(H(M), g_2^{r_i}) \\
&= e(h, y) \cdot e(H(M), \sigma_i'')
\end{aligned}
$$

If the verification holds, the other users will accept user i as a close contact.

In order to speed up the process, we use a batch verification scheme [18]: Let $(\delta_1, \cdots, \delta_{n-1})$ be a random vector of l_b bit elements from $\mathbb{Z}_q$ representing all of the other participants of the meeting. Accept if $e(\prod_{j=1}^{n-1} \sigma_j'^{\delta_j}, g_2) = \prod_{j=1}^{n-1} e(h, y_j)^{\delta_j} \cdot e(H(M), \sigma_j'')^{\delta_j}$. User i will then store the signatures $\{\sigma_1, \cdots, \sigma_{n-1}\}$ into the smartphone.

Proof Generation - After verifying the identity of all participants, the users need to generate proof that they participated in the meeting together, so that it can be stored into each smartphone. In order to do so, we propose two alternatives with DGNW-multi-signature[14] and MM-multi-signature[15] as described in Section 2. We will compare both in Section 6.

OF-MP Tracing - To ensure that OF-MP Tracing functions correctly, the message $M = \{ID_1, \cdots, ID_n\}$ has to be fixed beforehand. One of the users, user i, can easily compute the multi-signature Σ as the product of the signatures σ generated in the verification step: $\Sigma = (\sigma_1' \cdots \sigma_n', \sigma_1'' \cdots \sigma_n'') = (h^{x_1 + \cdots + x_n} \cdot H(M)^{r_1 + \cdots + r_n}, g_2^{r_1 + \cdots + r_n})$. He will then send the multi-signature Σ to the other $(n-1)$ users.

To verify the multi-signature, the users must compute the compressed public key of each close contact public key $Y = y_1 \cdots y_n$.

We then have $e(\Sigma', g_2) = e(h, Y) \cdot e(H(M), \Sigma'')$.

Correctness:

$$
\begin{aligned}
e(\Sigma', g_2) &= e(h^{x_1 + \cdots + x_n} \cdot H(M)^{r_1 + \cdots + r_n}, g_2) \\
&= e(h^{x_1 + \cdots + x_n}, g_2) \cdot e(H(M)^{r_1 + \cdots + r_n}, g_2) \\
&= e(h, g_2^{x_1 + \cdots + x_n}) \cdot e(H(M), g_2^{r_1 + \cdots + r_n}) \\
&= e(h, y_1 \cdots y_n) \cdot e(H(M), \Sigma'') \\
&= e(h, Y) \cdot e(H(M), \Sigma'')
\end{aligned}
$$

PF-MP Tracing - PF-MP Tracing satisfies the feature of *user flexibility*, that is, close contact users do not have to be decided beforehand in the protocol. The 1st user generates a signature on his ID ID_1, as follows: $R_1 = g_2^{k_1}$, $r_1 = R_1 + H(ID_1)$, and $s_1 = (x_1 \cdot r_1 + 1) \cdot k_1^{-1}$, for a randomly chosen $k_1 \in \mathbb{Z}_q$. User 1 will then send (ID_1, r_1, s_1) to one of his close contacts which will then go through the same process. For $j \geq 2$, we then have $M_{1,2,\cdots,j} = \{ID_1, ID_2, \cdots, ID_j\}$ and $R_j = g_2^{k_j}$, $r_j = R_j + H(ID_j) \cdot r_{j-1}$, and $s_j = (x_j \cdot r_j + 1) \cdot k_j^{-1}$, with k_j a random number generated by user j.

Finally, the multi-signature on $M_{1,2,\cdots,n}$ is given by $(ID_1, s_1), (ID_2, s_2), \cdots, (ID_n, s_n, r_n)$. The final user, user n will send the complete multi-signature to the other users.

This multi-signature can be verified as follows: $R_n' = g_2^{s_n^{-1}} \cdot y_n^{r_n \cdot s_n^{-1}}$, $T_n = r_n - R_n'$, and $r_{n-1} = T_n(H(ID_n))^{-1}$, repeating the process for $n = n - 1$ until $n = 2$ and checking if $T_1 = H(ID_1)$.

4.4 Alerting Phase

In the Alerting Phase, we assume that user s is a Covid-19 patient. User s needs to upload information on the database so that close contacts of the patient can be alerted while ensuring the patient's privacy. In order to do so, the patient will use a ring signature which guarantees the patient anonymity. As in Section 2.3, we assume $\mathbb{G}_1$ and $\mathbb{G}_2$ are GDH Group.

For each meeting during the last 14 days, the patient will retrieve the list of identifiers of his close contacts $L = \{ID_1, ID_2, \cdots, ID_n\}$. The patient will compute his ring public key Q_s as $H_1(ID_s)$ and will receive his private ring key $S_s = Q_s^{x_G}$ via a secure channel from GV.

The patient will then carry out the following steps to compute the ring signature on the message M which is equal to the set of identifiers, $M\{ID_1, \cdots, ID_n\}$:

- $\forall j\{1, 2, \cdots, n\} \setminus s$, choose $U_j \in \mathbb{G}_1$, compute $h_j = H(M \| U_j)$.
- Choose $r_s' \in \mathbb{Z}_q^*$, compute $U_s = Q_s^{r_s'} / \prod_{j \neq s}^n U_j \cdot Q_j^{h_j}$.
- Compute $h_s = H_1(M \| U_s)$ and $V = S_s^{(h_s + r_s')}$.
- Output the signature on M as $\sigma_r = \{\cup_{j=1}^n U_j, V\}$.

The patient will then upload the ring signature on the database.

4.5 Tracing Phase

In the Tracing Phase, the users will access the database and check every new entry $\sigma_r = \{\cup_{i=1}^n U_i, V\}$. They will retrieve the list of all IDs of the meetings they participated in, which corresponds to the messages on which the ring signatures were generated $M = \{ID_1, \cdots, ID_n\}$. They will then verify each ring signature in the following way:

- Compute $h_i = H_1(M \| U_i) \forall i \in \{1, 2, \cdots, n\}$.
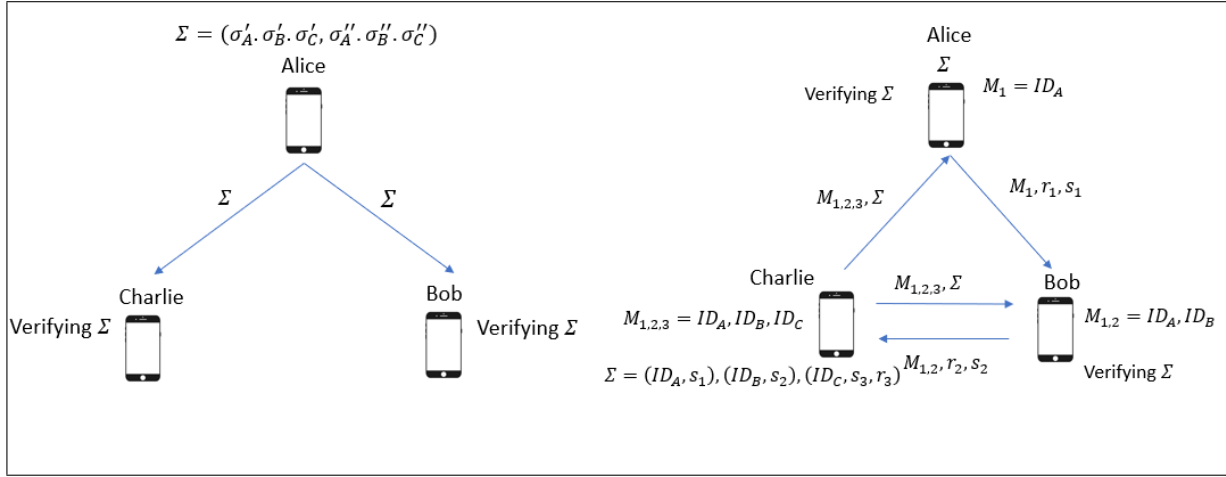
6

Fig. 2   Representation of the Meeting Phase for OF-MP Tracing and PF-MP Tracing

- Check whether $e(y_G, \prod_{i=1}^n U_i \cdot Q_i^{h_i}) = e(g_1, V)$.
- Accept the signature if it is true, reject otherwise.

If one of the signatures is accepted, it means that that user has been in contact with a positive patient. The user will be able to know in which meeting the user was in contact with the patient, but not the identity of the patient among the participants.

5. Security

In this section, we first justify the threat model defined in Section 3.3. Then, we present the security analysis of the proposed protocols.

5.1 Threat Model Justification

As explained in Section 3, we consider that a user is unable to modify information or add incomplete information on the user's app, such as adding a list of identifiers without the corresponding multi-signature or modifying a list of identifiers. Moreover, a user must use all information on the user's app while sending alerts, i.e. use all lists of identifiers stored on the user's app while computing ring signatures. Those assumptions are reasonable as a patient would be able to ignore information stored on the patient's app otherwise, preventing close contacts from being alerted.

As stated previously, our Threat Model uses concepts that were previously introduced in [12] but we redefine those concepts. More precisely we do not consider the users as honest-but-curious. However, our concept of Patient Privacy is weaker in the sense that it does not restrict the government from knowing the identity of the patient. Moreover, anyone can find out the identities of participants at a meeting with a patient but no one can to single out the identity of the patient. The concept of Contact Privacy defined is achievable by introducing a trusted entity (referred as doctor) to oversee the Tracing Phase (referred as the Medical Treatment Phase) while using a zero-knowledge proof to guarantee the privacy of the close contacts but that is beyond the current scope of this paper.

We will now justify our Threat Model. A threat model aims to identify threats and security requirements for a protocol. One of the first security requirements of a Contact Tracing app is to guarantee the privacy of its users, which is represented by **Patient**

Privacy. We assume that our protocol achieves Patient Privacy in the sense that nobody (except the government), including his own close contacts can single out his identity. The second security requirement is that all users that have been in a Meeting Phase with a patient (i.e. close contacts) are able to receive an alert during the Tracing Phase, which corresponds to **Traceability Completeness**. At the same time, it is important to prevent people from mistakenly receiving an alert, since it could cause them to needlessly self-isolate and get tested. This can happen either because they have received an alert from a patient that is not one of their close contacts, or because they have received an alert from one of their close contacts who is not a patient. Both of those cases would cause someone to be falsely considered as close contacts, and are referred in the threat model as **False Positive Case 1** and **False Positive Case 2** respectively.

5.2 Security Analysis

Specifically, the security of both false positives can be reduced to the unforgeability of the underlying ring and multi-signature schemes, respectively. The privacy of the patient and close contact can be reduced to the anonymity of the underlying ring signature schemes.

Theorem 1 False Positive Cases are negligible if the underlying multi-signature and ring signature schemes are unforgeable.

Proof The security of False Positive Case 1 and 2 is defined by a sequence of games. Game 0 is the original security game, where a simulator $\mathcal{S}$ simulates all system user's behaviours for a polynomial-time adversary $\mathcal{A}$ in the Meeting Phase. In Game 1, if $\mathcal{A}$ can pretend to be a positive patient/close-contact by generating a valid proof (i.e. a multi-signature) in the Meeting Phase while the patient/close-contact is honest, then $\mathcal{S}$ can use the generated multi-signature to break the existential unforgeability under chosen-message attacks (EUF-CMA) security of multi-signature scheme. Note that the generated multi-signature was not previously simulated by $\mathcal{S}$. Similarly, Game 2 is the original security game, where a simulator $\mathcal{S}$ simulates all system user's behaviours for a polynomial-time adversary $\mathcal{A}$ in the Alerting Phase. In Game 3, if $\mathcal{A}$ can pretend to be a positive patient/close-contact by generating a valid ring signature in the Alerting Phase, then $\mathcal{S}$

will break the EUF-CMA security of the ring signature.

We will first prove the security of False Positive Case 1. In order to pretend to be a close contact, a user needs to generate a multi-party proof in the Meeting Phase. We prove the security of **O**F-MP Tracing. The user has access to following algorithms:

- *Multi* which takes a message M, a list of public keys $\{y_1, \cdots, y_n\}$, a list of signatures $\{\sigma_1, \cdots, \sigma_n\}$ and outputs a multi-signature Σ.
- *MultiVerif* which takes a multi-signature Σ, a message M, an aggregated public key *agg* and outputs 1 if the signature is valid, or 0 if it is not valid.

Moreover, the adversary is able to make a signing query *Sign* which takes as a parameter a message M, uses the current secret key x and outputs a signature σ, and a *KeyUpdate* query which updates the current secret key.

In Game 0, $\mathcal{S}$ simulates all system user's behaviours for a polynomial-time adversary $\mathcal{A}$.

initialization:

(1) $(x_1, y_1) \leftarrow KeyGen(\lambda)$

(2) For $2 \leq i \leq n$, $(x_i, y_i) \leftarrow KeyGen(\lambda)$

(3) $j = 1$

Send (y_1) to $\mathcal{A}$

Upon receiving a key update query, with x_j the current secret key,

(4) if $j < n$, updates secret key to x_{j+1}

Upon receiving signing query on M, with current secret key x_j

(5) $\sigma_j^* \leftarrow Sign(M, x_j)$ send σ_j^* to $\mathcal{A}$;

at the end, $\mathcal{A}$ outputs

$\Sigma^* = Multi(M^*, y_1^*, \cdots, y_n^*, \sigma_1^*, \cdots, \sigma_n^*)$

(6) if $MultiVerif(agg, \Sigma^*, M^*) = 1$ for *agg* aggregate public key of $(y_1^*, \cdots, y_n^*)$, $y_1 \in \{y_1^*, \cdots, y_n^*\}$ and $\mathcal{A}$ never made a signing query on M^*:

Output 1

(7) else, output 0

We define Game 1 in a similar way but we add a necessary winning condition, which corresponds to the condition necessary to compute a valid multi-party proof in the Meeting Phase:

(6) if $MultiVerif(agg, \Sigma^*, M^*) = 1$ for *agg* aggregate public key of $(y_1^*, \cdots, y_n^*)$, $y_1 \in \{y_1^*, \cdots, y_n^*\}$, $\mathcal{A}$ never made a signing query on M^* and M^* is of the form $\{ID_1, \cdots, ID_n\}$:

Output 1

Let W_0 and W_1 be the event that $\mathcal{A}$ outputs 1 in Game 0 and Game 1 respectively. $\Pr[W_0]$ and $\Pr[W_1]$ are the corresponding probabilities. If W_1 occurs, $\mathcal{A}$ can output the generated multi-signature on the same message to win Game 0, so W_0 occurs. We then have $\Pr[W_1] \leq \Pr[W_0]$. Game 0 corresponds to the security game defined in DGNW-multi-signature scheme. So if W_1 occurs, W_0 occurs and $\mathcal{S}$ can use the generated multi-signature to break the existential unforgeability under chosen-message attacks (EUF-CMA) security of the DGNW-multi-signature scheme. Security of PF-MP Tracing follows in the same way.

We then have that False Positive Case 1 is negligible if the underlying multi-signature scheme is unforgeable.

We will now prove that False Positive Case 2 is negligible. In order to pretend to be a positive patient, a user has to generate a valid ring signature on a list of IDs previously stored in the user's app which the user will then upload on the server in the Alerting Phase. The user can make the following queries:

- H, H_1 which takes an input as parameter and returns the value of the corresponding hash.
- *KeyGen* which takes as input an ID and returns the corresponding secret key $KeyGen(ID) = S_{ID}$.
- *Sign* which takes as input a message M, a list of identifiers $\{ID_1, \cdots, ID_n\}$ and returns a ring signature.

We define Game 2 in the following way:

initialization

(1) $n \leq \mathbb{N}$ For $1 \leq i \leq n$, $ID_i \leftarrow \{0, 1\}^*$

(2) x_G master secret key

Upon receiving a *KeyGen* query on an *ID*

(3) $S = KeyGen(ID)$ send S to $\mathcal{A}$;

Upon receiving a signing query on $\{ID_1, \cdots, ID_n\}$ and the message M

(4) $\sigma_r = Sign(\{ID_1, \cdots, ID_n\}, M)$ send σ_r to $\mathcal{A}$;

At the end $\mathcal{A}$ outputs σ_r on the message M^*

(5) if $Verify(\sigma_r, M^*) = 1$, and $\mathcal{A}$ never made a *KeyGen* query on a identifier in $\{ID_1, \cdots, ID_n\}$

Output 1

(6) else, output 0

We define Game 3 in the same way but we add a winning condition which corresponds to a valid ring signature in the Alerting Phase:

(5) if $Verify(\sigma_r, M^*) = 1$, and $\mathcal{A}$ never made a *KeyGen* query on a identifier in $\{ID_1, \cdots, ID_n\}$ and M^* is equal to the list of identifiers $\{ID_1, \cdots, ID_n\}$

Let W_2 and W_3 be the event that $\mathcal{A}$ outputs 1 in Game 2 and Game 3 respectively. $\Pr[W_2]$ and $\Pr[W_3]$ are the corresponding probabilities. If W_3 occurs, $\mathcal{A}$ can output the generated ring signature on the same message to win Game 2, so W_2 occurs. We then have $\Pr[W_3] \leq \Pr[W_2]$. Game 2 corresponds to the security game defined in the ring signature scheme. So if W_3 occurs, W_2 occurs and $\mathcal{S}$ can use the generated ring signature to break the existential unforgeability under chosen-message attacks (EUF-CMA) security of the underlying ring signature scheme.

Since we assumed $\mathbb{G}_1$ and $\mathbb{G}_2$ are GDH Groups, the advantage of $\mathcal{A}$, which corresponds to $\Pr[W_2]$ is negligible.

We then have that False Positive Case 2 is negligible if the underlying ring signature scheme is unforgeable which concludes the proof of Theorem 1.

Theorem 2 The patient privacy is maintained if the underlying ring scheme has the unconditional signer ambiguity property.

Proof The adversary can either be a close contact of the patient, i.e. part of the group represented by the signer, or an outsider. We will start with the adversary being a close contact. Let $\mathcal{A}$ be a polynomial-time adversary and let σ_r be a given ring signature uploaded on the database representing a meeting between n participants. We define $Adv_{\mathcal{A}}$ as the probability of finding the identity of the patient. If $\mathcal{A}$ is able to break the patient privacy, we have $Adv_{\mathcal{A}} > 1/n$ which would correspond to a wild guess. However, it would mean that an adversary $\mathcal{A}$ is able to find the identity of the signer with probability better than $1/n$ which would

Table 1 Comparison between the number of computations, communications and the characteristics of 3 protocols

	Communications	Computations	Characteristics
OF-MP Tracing	2	$2(n-1)E+(2n+1)P$	OF, MP
PF-MP Tracing	4	$(4n-1)E+2(n-1)P$	UF, OF, MP
Primitive	$4(n-1)$	$5(n-1)E+2(n-1)P$	OF, UF

E : Exponentiations   P : Pairings
n : number of Participants   OF : Order Flexibility
MP : Multi-party Proof   UF : User Flexibility

break the unconditional signer ambiguity property of the underlying ring signature. The reasoning holds if $\mathcal{A}$ is an outsider. We then have that the protocol has Patient Privacy if the underlying ring scheme has the unconditional signer ambiguity property.

Theorem 3 The protocol has Traceability completeness if it is correctly executed.

Proof As stated in Section 3, we consider that a patient will use all information stored in his app in order to send an alert. Moreover, a user is unable to modify the information after it has been stored in his app. For $1 \leq i \leq n$, let User i with identifier ID_i be a participant in a meeting. During the Meeting Phase, each user will verify the identity of the $(n-1)$ other users. They will then generate a multi-signature. As the protocol was correctly executed, the multi-signature will be valid and each user will store it, with the list of identifiers present in the meeting $\{ID_1, \cdots, ID_n\}$. Let User s be the patient during the Alerting Phase, $1 \leq s \leq n$. User s will retrieve the list of identifier $\{ID_1, \cdots, ID_n\}$ to generate the ring signature and upload it on the database. During the Tracing Phase, for $1 \leq i \leq n$, $i \neq s$, User i will verify the ring signature. As the protocol was correctly executed, $\{ID_1, \cdots, ID_n\}$ will have been stored in user i app and the verification will accept the ring signature. User i has been alerted that he has been in contact with a patient. We then know that the protocol has Traceability Completeness if it is correctly executed. □

6. Efficiency and Characteristics

In this section, we compare the characteristics, computations, and communications of PF-MP Tracing, OF-MP Tracing and Primitive Multi-Party Tracing Protocol introduced in Section 3.1. As in the rest of the protocol, we consider a meeting between n participants, $n \geq 2$. We have shown in Section 5 that all three protocols achieve Patient Privacy and False Positive Case 1 and 2. Moreover, Primitive Multi-Party protocol achieves Contact Privacy by using a zero-knowledge proof.

In OF-MP Tracing, all participants of a meeting will generate a multi-party proof, identical for every participant, and store it in their smartphone as proof of the meeting. Moreover, the order of the users has no impact on the protocol since users will go through the verification with all other users simultaneously. As a result, OF-MP Tracing achieves Order Flexibility and Multi-Party proof.

PF-MP Tracing achieves the same properties as OF-MP Tracing but also achieves User Flexibility. PF-MP Tracing has the message flexibility property, since the message on which the multi-signature is generated is updated by each signer. In our protocol, the message corresponds to the list of participants so it

doesn't have to be fixed beforehand.

In Primitive Multi-Party Tracing Protocol, the user will verify the identity of each user separately and generate a mutual commitment for all of them. Furthermore, each verification and mutual commitment are completely independent so the order itself does not have an impact on the protocol. Moreover, since the protocol is a 2-party protocol, the user will add all other users as close contacts one after the other so the list of users does not have to be fixed beforehand. However, the user will not store a multi-party proof, so the result is the same as if the user met the $(n-1)$ other users separately.

We can see that in PF-MP Tracing and OF-MP Tracing, the number of communications is fixed and does not depend on the number of participants in a meeting. In Primitive Multi-Party Tracing Protocol, the number of communications is linear. The number of computations required in PF-MP Tracing and OF-MP Tracing is also less than in Primitive Multi-Party Tracing Protocol while generating proof that all participants were present. PF-MP Tracing requires slightly more exponentiations, but allows the message to be generated during the protocol instead of being fixed.

7. Conclusion

We propose two Contact Tracing protocols in a multi-party setting. These protocols are divided into four phases and guarantee that a close contact of a Covid-19 patient will receive an alert. In both protocols, proof will be generated at the end of the Meeting Phase and stored on every participants' smartphones. This proof will differentiate a meeting between several participants and a repetition of smaller meetings. Moreover, the Patient Privacy is maintained by using a ring signature.

References

[1] Hassandoust, F.: Not just complacency: why people are reluctant to use COVID-19 contact-tracing apps (2020).
[2] McGowan, M.: Privacy concerns persist over Australia's coronavirus tracing app (2020).
[3] Nolsoe, E.: NHS COVID-19 app success hinges on battery life (2020).
[4] Tang, Q.: Privacy-Preserving Contact Tracing: current solutions and open questions, *IACR Cryptol. ePrint Arch.*, Vol. 2020, p. 426 (online), available from ⟨https://eprint.iacr.org/2020/426⟩ (2020).
[5] Gvili, Y.: Security Analysis of the COVID-19 Contact Tracing Specifications by Apple Inc. and Google Inc, *IACR Cryptol. ePrint Arch.*, Vol. 2020, p. 428 (online), available from ⟨https://eprint.iacr.org/2020/428⟩ (2020).
[6] Vaudenay, S.: Centralized or Decentralized? The Contact Tracing Dilemma, *IACR Cryptol. ePrint Arch.*, Vol. 2020, p. 531 (online), available from ⟨https://eprint.iacr.org/2020/531⟩ (2020).
[7] Troncoso, C., Payer, M., Hubaux, J.-P., Salathé, M., Larus, J., Bugnion, E., Lueks, W., Stadler, T., Pyrgelis, A., Antonioli, D., Barman, L., Chatel, S., Paterson, K., Čapkun, S., Basin, D., Beutel, J., Jackson, D., Roeschlin, M., Leu, P., Preneel, B., Smart, N., Abidin, A., Gürses, S., Veale, M., Cremers, C., Backes, M., Tippenhauer, N. O., Binns, R., Cattuto, C., Barrat, A., Fiore, D., Barbosa, M., Oliveira, R. and Pereira, J.: Decentralized Privacy-Preserving Proximity Tracing (2020).
[8] Vaudenay, S.: Analysis of DP3T, *IACR Cryptol. ePrint Arch.*, Vol. 2020, p. 399 (online), available from ⟨https://eprint.iacr.org/2020/399⟩ (2020).
[9] Pietrzak, K.: Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing, *Progress in Cryptology – IN-*

DOCRYPT 2020 (Bhargavan, K., Oswald, E. and Prabhakaran, M., eds.), Cham, Springer International Publishing, pp. 3–15 (2020).

[10] Luo, Y., Zhang, C., Zhang, Y., Zuo, C., Xuan, D., Lin, Z., Champion, A. C. and Shroff, N. B.: ACOUSTIC-TURF: Acoustic-based Privacy-Preserving COVID-19 Contact Tracing, *CoRR*, Vol. abs/2006.13362 (online), available from ⟨https://arxiv.org/abs/2006.13362⟩ (2020).

[11] Meklenburg, J., Specter, M. A., Wentz, M., Balakrishnan, H., Chandrakasan, A., Cohn, J., Hatke, G., Ivers, L., Rivest, R. L., Sussman, G. J. and Weitzner, D. J.: SonicPACT: An Ultrasonic Ranging Method for the Private Automated Contact Tracing (PACT) Protocol, *CoRR*, Vol. abs/2012.04770 (online), available from ⟨https://arxiv.org/abs/2012.04770⟩ (2020).

[12] Liu, J. K., Au, M. H., Yuen, T. H., Zuo, C., Wang, J., Sakzad, A., Luo, X. and Li, L.: Privacy-Preserving COVID-19 Contact Tracing App: A Zero-Knowledge Proof Approach, *IACR Cryptol. ePrint Arch.*, Vol. 2020, p. 528 (online), available from ⟨https://eprint.iacr.org/2020/528⟩ (2020).

[13] Peng, Z., Xu, C., Wang, H., Huang, J., Xu, J. and Chu, X.: P²B-Trace: Privacy-Preserving Blockchain-based Contact Tracing to Combat Pandemics, *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, Xi'an, Shaanxi, China, pp. 2389–2393 (online), DOI: 10.1145/3448016.3459237 (2021).

[14] Drijvers, M., Gorbunov, S., Neven, G. and Wee, H.: Pixel: Multi-signatures for Consensus, *USENIX*, pp. 2093–2110 (2020).

[15] MITOMI, S. and Miyaji, A.: A General Model of Multisignature Schemes with Message Flexibility, Order Flexibility, and Order Verifiability, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, Vol. 84-A, No. 10, pp. 2488–2499 (2001).

[16] Chow, S. S. M., Yiu, S.-M. and Hui, L. C. K.: Efficient Identity Based Ring Signature, *Applied Cryptography and Network Security* (Ioannidis, J., Keromytis, A. and Yung, M., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 499–512 (2005).

[17] de Goyon, M., Miyaji, A. and Tian, Y.: Efficient Multi-Party Contact Tracing, *Ninth International Symposium on Computing and Networking, CANDAR 2021, Matsue, Japan, November 23-26, 2021*, IEEE, pp. 10–18 (online), DOI: 10.1109/CANDAR53791.2021.00010 (2021).

[18] Ferrara, A. L., Green, M., Hohenberger, S. and Pedersen, M. Ø.: Practical Short Signature Batch Verification, *IACR Cryptol. ePrint Arch.*, Vol. 2008, p. 15 (online), available from ⟨http://eprint.iacr.org/2008/015⟩ (2008).
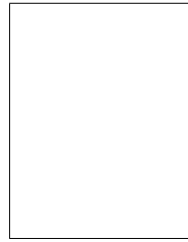
Yangguang Tian received his PhD degree in Applied Cryptography from the University of Wollongong, Australia. He is a lecturer at University of Surrey, UK. His research interests include applied cryptography, network security, blockchain technologies and privacy-preserving technologies.
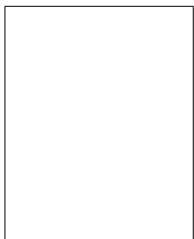
Mathieu de Goyon received his engineering degree from IMT Lille Douai in France in 2021. He joined Miyaji Laboratory in Osaka University in April 2022. He is currently a first year doctorate student at Osaka University. His research interest is contact tracing and post-quantum cryptography.
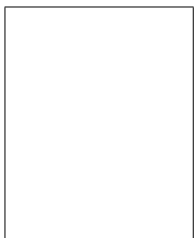
Atsuko Miyaji received the B. Sc., the M. Sc., and the Dr. Sci. degrees in mathematics from Osaka University, in 1988, 1990, and 1997 respectively. She is an IPSJ fellow. She has been a professor at Japan Advanced Institute of Science and Technology (JAIST) since 2007 and a professor at Graduate School of Engineering, Osaka University since 2015. Her research interests include the application of number theory into cryptography and information security.