Khopesh - Contact Tracing Without Sacrificing Privacy

Friedrich Doku[1(✉)] and Ethan Doku[2]

[1] University of Pittsburgh, Pittsburgh, PA, USA
frd20@pitt.edu
[2] Irondale High School, New Brighton, MN, USA
2021dokue58@moundsviewschools.org

Abstract. Secure contact tracing has proven challenging to implement, because even if a user's contact data is encrypted, it is still difficult to hide the user's metadata, which could be used to determine a user's identity, furthermore, many existing contact tracing software implementations may require a user to send sensitive information such as location data to be processed by a central authority. Current systems such as DP-3T [9] and COVIDSafe [10] do not provide anonymity or much privacy protection. It is also not guaranteed that the data collected will not be used for marketing, commercial gain, or law enforcement.

Khopesh is a new secure contact tracing system that offers strong privacy guarantees, hiding both a user's contacts and location data. This is made possible through the use of identity-based encryption, mix networks, and a novel technique called secure-contact contract signing, which enables groups of users to view each other's reports.

Keywords: Anonymity · Privacy · Communication

1 Introduction

In the wake of the Covid-19 virus, we find that the world is underprepared to track the spread of pandemics. Existing methods to track the spread of the virus heavily rely on the government to deploy large-scale surveillance operations to track people. These existing methods provide users with almost no privacy and anonymity whatsoever. However, many of these contact tracing services allow users to opt-in or opt-out, but if governments expect the number of users using contract tracing apps to increase, the designers need to design protocols with stronger privacy guarantees. Users should not have to sacrifice their privacy to help stop the spread of the Covid-19 virus, and companies should not be able to take advantage of the current situation to increase their profits.

Many users would like secure contact tracing, but current contact tracing software does not guarantee privacy. Even if the data is encrypted, adversaries can still learn a lot about a user based on metadata, which can be observed

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 475–486, 2020.
https://doi.org/10.1007/978-3-030-63095-9_30

through traffic analysis. Contact tracing software needs to be designed for security, adding encryption on top of an insecure protocol does not make it more secure.

Covid-19 is indeed a pandemic, and current tracing technology, such as COVIDSafe [10], VirusRadar [1], etc, help monitor the spread of Covid-19 and provide users with some security. However, these systems still do not provide users with anonymity.

This paper presents Khopesh, a secure contact tracing system that protects the identity and contact data of its users. Khopesh efficiently tracks the spread of infectious diseases like Covid-19 while providing users with anonymity and privacy. Khopesh can keep its users completely anonymous as long as two users are active. Furthermore, it is still possible for a user to know whether a friend or family member has been infected with the virus despite their anonymity. Khopesh does not collect location data for contact tracing.

Khopesh keeps passive adversaries that monitor network traffic from learning the identity of its users by exploiting a mixnet. Users will generate a random permutation of mixer public keys. Next, they will onion-encrypt their report data with the public keys in the order determined by the permutation and send it to the mixnet, which will shuffle reports and decrypt layers of encryption before sending it to a mailbox server where users will download their reports. Khopesh operates in rounds, and each round users will follow this process.

Khopesh's drawback is that to protect its users from a passive attacking adversary that monitors network traffic, it must make some performance sacrifices. For example, Khopesh's users must download and decrypt many reports before viewing their reports because many users will share a mailbox, and inside each mailbox are encrypted reports for specific users. This downloading and decrypting of reports exploits significant amounts of bandwidth and CPU resources.

Nevertheless, Khopesh is still more secure than traditional contact tracing systems. Khopesh may use more computing resources than other contract tracing systems, but the extra computing resources are all used to protect a user's personal information, providing Khopesh with stronger privacy guarantees.

All in all, we make the following contributions:

- Analyze the design of Khopesh, a secure contact tracing system that keeps its users anonymous and their contacts protected.
- Introduce a novel technique called secure-contact contract signing, which allows users to form groups and share their reports securely.

2 Threat Model and Goals

Khopesh aims to keep its users anonymous and their contacts private, while effectively tracing the spread of infectious diseases. In this section, we describe Khopesh's threat model and goals.

2.1 Threat Model

Khopesh's design assumes that a passive attacking adversary is monitoring all of Khopesh's traffic, controls all but one of Khopesh's mixer servers (users do not need to know which one), and controls an arbitrary number of users. We also assume that the adversary can block, delay, or inject traffic. A user, Alice, sending reports through Khopesh should have their communication protected as long as one mixer is uncompromised. Since users will send their reports over multiple rounds, we assume that the adversary may interfere with the users more than one round. The adversary's goals include finding out who is in the network and who is sending reports to whom.

All cryptographic assumptions are standard (the adversary will be unable to break cryptographic primitives). We assume secure public key encryption, hash functions, and key-exchange mechanisms. We assume that Khopesh's mixer public keys are known to all users.

Finally, we assume that honest Khopesh servers and users correctly implement the Khopesh protocol, and no data leakage exists through side channels. In the case of a server or client being controlled by an adversary, it is assumed that they are not following the protocol, but honest servers and users are assumed to be running bug-free implementations of Khopesh.

Khopesh is not a fault-tolerant system. Khopesh can recover from minor server failures, but it does not protect against server component failures or denial-of-service attacks. Since we do not cover availability attacks on Khopesh, we will leave this out of scope.

Cryptographic Primitives. Khopesh relies on the following cryptographic primitives.

Identity Based Encryption. Khopesh exploits the Boneh-Franklin Identity-Based encryption scheme [5] to create public and private key pairs for secure-contact contracts and for users, which consists of the following algorithms:

- $(mk, params) \leftarrow \mathsf{Setup}(k)$. Generates the system parameter $params$ and the master-key mk, from a security parameter k.
- $(d) \leftarrow \mathsf{Extract}(params, mk, ID)$. Generates the user's private key d given the $params$, mk, and the user's id ID.
- $(C) \leftarrow \mathsf{Encrypt}(params, ID, M)$. Encrypts a message M from $params$ and ID to create a ciphertext C
- $(M) \leftarrow \mathsf{Decrypt}(params, C, d)$. Decrypts C from d, $params$, and mk to generate M.

XSalsa20 Authenticated Encryption. Khopesh exploits XSalsa20, which provides immunity to timing attacks. Each mixnet server and user in Khopesh uses XSalsa20 for encrypting and decrypting reports.

- $(C) \leftarrow \mathsf{AEncrypt}(n, M, pk)$. Encrypts a message M from a public key pk and a cryptographic nonce n to create a ciphertext C
- $(M) \leftarrow \mathsf{ADecrypt}(n, pk, C, sk)$. Decrypts C from the recipients pk, private key sk, and n.

2.2 Goals

Khopesh has three primary goals.

Correctness. Informally, Khopesh is correct if every honest user can successfully send her report to the central authority, and the central authority can process that data and inform users about infected users.

Anonymity. Khopesh achieves the goal of keeping the identity of its users anonymous if the passive attacking adversary described in Sect. 2.1 is unable to uncover the identity of Khopesh's users by no more than random guessing.

Privacy. Khopesh achieves the goal of protecting privacy if users can have their contacts hidden from the passive attacking adversary described in Sect. 2.1 and can inform other trusted users of their current status.

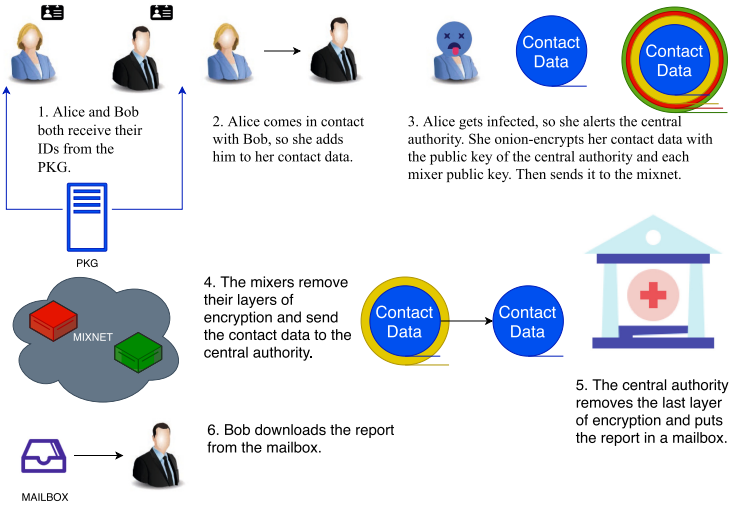

Fig. 1. An overview of Zephyr's protocol.

3 Design

We will now explain the details of Khopesh and how it protects itself from the passive attacking adversary that is described in Sect. 2.1.

3.1 Private Key Generators

Traditional key distribution servers leak data about who a user is communicating with based on the public-key the user receives from the server. To avoid key distribution, Khopesh exploits a private key generator (PKG), which is a trusted third-party server that is responsible for generating the IDs of users and private keys for secure-contact contracts using IBE.

When users start the Khopesh protocol, they will first authenticate themselves by sending their phone numbers to the PKG. Once the PKG has received the user's phone number, it will send a text message containing a unique code to the user's phone. Once the user returns the code to the PKG, the PKG will send the user her ID, which is a hash of their encrypted phone number using a cryptographically secure string as the public key.

To create a secure-contact contract, the users will authenticate themselves again but instead receive a shared ID and private key. The shared ID is computed by hashing an encrypted concatenation of each group member's phone number with a cryptographically secure string as the key. The PKG will store each user's ID and shared ID.

3.2 Mixnet

Khopesh exploits a mixnet to prevent reports from being traced back to their senders.

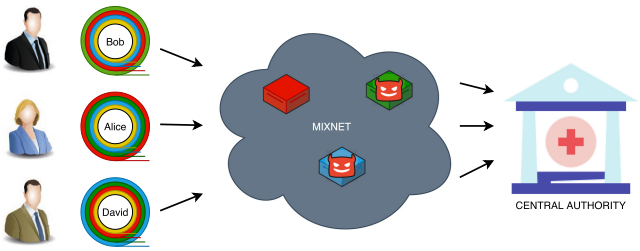

Fig. 2. Users send their data through a mixnet.

Once a mixer in the mixnet has received reports, it will shuffle them, remove layers of encryption, and send them to the next mixer. Figure 2 shows a group of users Bob, Alice, and David each has onion-encrypted their reports with mixer public keys in different orders after first encrypting their reports with the central authority's public key. Each report passes through the mixnet, and the last mixer to decrypt the final layer of encryption of a report sends the report to the central authority. As long as one mixer is honest, an adversary will be unable to figure out the destination of a user's report because each mixer shuffles their reports before sending them to the next mixer.

It is necessary to make sure that mixers are shuffling the reports. If the reports are not shuffled, it will be easier for adversaries to figure out the identities of Khopesh's users. To ensure that mixers are not misbehaving, Khopesh exploits zero-knowledge shuffle proofs for verifying the shuffling process of each mixer. These proofs prove that each shuffle output is a permutation of each shuffle input without revealing any knowledge of the permutation.

3.3 Mailboxes

Mailboxes are publicly known memory locations that reside in the central authority. Mailboxes provide Khopesh's users with greater identity protection because each mailbox is determined by the user's phone number modulus the number of mailboxes, which means that several users will share the same mailbox. Adversaries will be unable to figure out the identity of a user because every other user will download their reports from the same server, creating no distinction between other users.

The central authority will send all infection reports to a single mailbox and shared reports (reports created with secure-contact contracts) will be placed into the mailboxes of their recipients.

3.4 Users

We will now describe how users operate in Khopesh.

Receiving Contacts. Users advertise themselves as Khopesh users and scan for each other using Bluetooth Low Energy (BLE). Once a connection has been formed, users will exchange their contact data with each other using BLE and store the contact data in a general tree. This data structure will contain all the people a user was in contact with and any indirect interactions. However, the phone numbers nor the identity of the people a user interacts with will be stored. What will be stored in the data structure are the IDs of each person a user encounters. For example, Fig. 4 shows the user 8DK9S0D0 (Alice) who has been in contact with multiple users. As you can see, the names/phone numbers of all of the connected users are not shown. Alice's indirect connections are the two leaf nodes that are under the user with the ID D7G9ZZQ2 (Fig. 3).

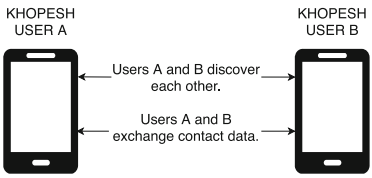

Fig. 3. Khopesh BLE handshake.

This general tree data structure makes it easy for users to receive reports about possible infections. If the central authority releases a report that a user has been infected, other users can check if they have been possibly infected by searching their trees for the infected user's ID.

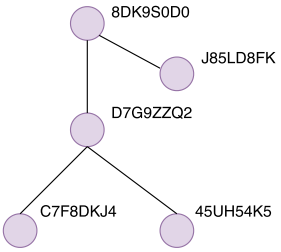

Fig. 4. Structure of user contacts.

Receiving Infection Reports. When a user gets infected, she will send her report through the mixnet to the central authority with a flag that shows that she has the virus. The central authority will send this report to a report mailbox, which is the mailbox Bob used to download Alice's infection report in Fig. 1. Users will download this report mailbox, which will contain the IDs of users that have the virus. If a user finds one or more of these IDs in their contact data, there is a chance they could have the virus.

Sending Reports. To protect their contact reports from being traced back to their identity, infected users are required to send their reports to be processed through the mixnet before reaching the central authority. At the end of each round, users will encrypt their report with the central authority's public key. Next, they will generate a random permutation of mixer public keys and encrypt their reports with each public key in the order determined by the generated random permutation, forming encapsulated layers of encryption. Upon adding a layer of encryption, a user will add the destination IP address of the mixer that is to decrypt the encrypted layer.

When reports are sent, any known information about a user's contacts such as names and phone numbers will be removed. The central authority will only be able to view the user's ID.

Secure-Contact Lists. Secure-contact lists enable groups of users to share their contact reports with each other securely. By sharing contact reports users in the group will be able to check the likelihood that group members have been infected. The creator of the contract will send a request to the PKG. The PKG will then show the contract to each user after authenticating them by sending a code to their phone as described in Sect. 3.1. If every user agrees to the contract then the contract is valid, and users in the group will be able to view each other's reports. Now users can use the shared public key to encrypt their reports. Users will add the mailbox address to their reports and send the to the central authority just as they would for an infection report. The central authority will place the reports in the mailboxes. If all users on the contract do not agree to the contract, it is invalid, and the group will not form (Fig. 5).

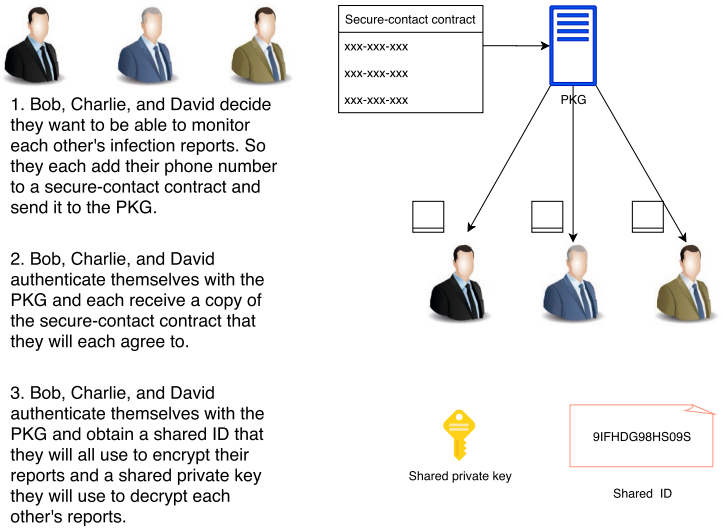

Fig. 5. An overview of secure-contact contracts.

4 Analysis

4.1 Correctness

If the protocol is carried out faithfully, then Khopesh's mixers will shuffle the reports, and the last mixer will send the reports to the central authority for processing. Infection reports will be placed in the user's mailboxes. Thus, Khopesh satisfies the correctness property.

4.2 Sender Anonymity

Sender anonymity relies on the verifiability and zero-knowledge property of the verifiable shuffle. In the mixnet, every upstream mixer is the prover, and every downstream mixer is the verifier of the verifiable shuffle. Verifiability ensures that the protocol is carried out correctly. If it is not, the verifier will halt and throw an error. Apart from this, every honest mixer's permutation is unknown to the adversary. Therefore, the final permutation of the reports will also be unknown to the adversary, so the adversary will be unable to link the report to a user.

To protect itself from N-1 attacks, Khopesh's mixnet implements Mixminion's [6] "timed-dynamic-pool" batching strategy; mixers process reports every t seconds but wait until they have a threshold of reports before beginning processing.

4.3 Receiver Anonymity

The anonymity of mailbox downloads depends on whether or not the adversary knows which mailbox a user uses. If the adversary does know the mailbox of a user, receiver anonymity is still achieved to an extent because the user could be any user that uses that mailbox. However, if the adversary does not know the mailbox a user uses, then full receiver anonymity is achieved because the user could be any user in the network.

4.4 Privacy

Secure-Contact Contracts. Khopesh's secure-contact contracts enable users to create groups where each member can view the reports of each other group member. Communication to and from the PKG is encrypted, so an adversary will be unable to tamper with the secure-contact contract. All users are shown the list and choose to agree to the contract. If one or more users disagree, the contract will be invalid, and the group will not form. The privacy of secure-contact contracts depends on the integrity of each user in the group. If one user in the group leaks another group member's report, privacy is lost.

Regular Reporting. Traditional reporting (no secure-contact contracts) does not leak any information to the adversary. Users send their reports to the mixnet, and the mixnet sends their reports to the central authority for processing.

Bluetooth Sending. Reports contain no information about any user since phone numbers are removed from reports before being sent to other users. Only IDs of users remain and these cannot be linked to any user's phone number. Even if the adversary obtains these contact reports they will only be able to find relationships between anonymous users.

5 Limitations and Vulnerabilities

5.1 Bluetooth

Using Bluetooth to detect other users can lead to an increased number of false positives because the distance between users is not very accurate. To solve this problem, Khopesh could use Bluetooth localization techniques to improve location accuracy [2,11].

5.2 Verifiable Shuffling

Khopesh implements Bayer and Groth's [4] verifiable shuffle algorithm, which takes 2 min to prove and verify 100,000 reports. This is a fast algorithm, but it has not been optimized for Khopesh; its current implementation does not exploit parallelism, which is a performance loss for Khopesh. This implementation could be made faster by including support for multiple cores.

5.3 Forward Secrecy

Khopesh's current design does not include forward secrecy. This allows an adversary to collect reports of users from the mailboxes and decrypt them later once she has obtained a user's private key. To mitigate this, Khopesh could regenerate keys every round. However, this means users would have to reauthenticate themselves to receive their new private keys from the PKG.

5.4 PKG Attacks

Khopesh's PKG creates the private keys for secure-contact contracts and users and also authenticates users, which makes it a potential target. If the PKG is compromised, an adversary will be able to view all the phone numbers of users that will authenticate themselves with their phone numbers in the future. However, the phone numbers of current users will be protected as long as they do not create a secure-contact contract, which requires reauthentication.

6 Implementation

A prototype of Khopesh is available at https://github.com/MutexUnlocked/khopesh. The prototype implements every component of Khopesh's protocol, including a user prototype for Android smartphones.

7 Evaluation

Our evaluation quantitatively answers the following question:

– Can Khopesh support a large amount of users and reports?

7.1 Experimental Setup

To answer the above question, we ran an experiment on DigitalOcean Droplet servers (Linux-based virtual machines). All servers used had an Intel Xeon Skylake 2.7 GHz CPU with 4 cores, 8 GB of RAM, and 1 Gbps of network bandwidth. The servers run Linux version 5.7.7.

We use the following parameters in our experiment. Our mixnet consisted of 3 mixers, 1 PKG, and a central authority each corresponding to one VM. All reports are premade with a fixed size of 1192 bytes (including 144 bytes of encryption overhead). To ensure that clients do not bottleneck, clients are simulated using 4 additional VMs. Each user sends their reports regularly (no secure-contact contracts).

7.2 Server Performance

Figure 6 shows that Khopesh scales linearly with the number of users and reports. The end-to-end latency for Khopesh includes authentication with the PKG, processing reports with the mixnet, and sending reports to the central authority.

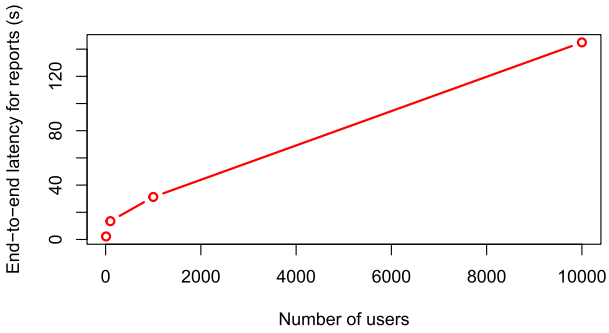

Fig. 6. Performance of Khopesh's protocol when varying the number of online users. Every user sends a report every round.

8 Related Work

8.1 Covid-19 Tracing Protocols

BlueTrace [3] is a privacy-preserving contact tracing protocol that is also the foundation for Covid-19 tracing apps such as COVIDSafe [10]. BlueTrace protects the contacts of users. However, it gives health authorities the power to obtain and use personally-identifiable information of infected users. In contrast, Khopesh's central authority is unable to view any information associated with users, which includes phone numbers. The central authority only processes the hashes of encrypted phone numbers.

DP-3T [9] is a decentralized Covid-19 tracing protocol that protects the contacts of users, thus, providing users with more privacy and security than traditional centralized Covid-19 tracing systems. However, DP-3T is subject to the targeted identification attack [8]. In contrast, Khopesh is not vulnerable to any such attack and keeps its infected users and non-infected users equally anonymous.

8.2 Protocols that Provide Anonymity

Khopesh implements a free-route mixnet similar to Mixminion's [6] mixnet, for example, both mixnet designs rely on TLS over TCP for communication between mixers. However, Khopesh's mixnet does not provide the same flexibility as Mixminion. For example, replies and message forwarding are not supported. In Khopesh every report that travels through the mixnet is treated the same.

Alpenhorn [7] is a protocol used for initiating connections between two users without leaking metadata. Khopesh uses Alpenhorn's idea of using IBE to create private keys for its users without revealing much metadata. However, Alpenhorn's PKG implements forward secrecy, making it safer from attackers that later compromise a user's private key.

9 Conclusion

Khopesh is a new secure contact tracing system that provides users with both anonymity and privacy. This is made possible through the use of identity-based encryption, mix networks, and a novel technique called secure-contact contract signing, which enable users to be notified if one or more of their contacts, whom they trust, have been affected.

References

1. Virusradar (2020). https://virusradar.hu/
2. Almaula, V., Cheng, D.: Bluetooth triangulator. Final Project, Department of Computer Science and Engineering, University of California, San Diego, pp. 1–5 (2006)
3. Bay, J., et al.: Bluetrace: a privacy-preserving protocol for community-driven contact tracing across borders. Technical Report, Government Technology Agency-Singapore (2020)
4. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_17
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
6. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: design of a type III anonymous remailer protocol. In: 2003 Symposium on Security and Privacy, 2003, pp. 2–15. IEEE (2003)
7. Lazar, D., Zeldovich, N.: Alpenhorn: Bootstrapping secure communication without leaking metadata. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 571–586. USENIX Association, Savannah, GA, November 2016. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/lazar
8. Tang, Q.: Privacy-preserving contact tracing: current solutions and open questions. arXiv preprint arXiv:2004.06818 (2020)
9. Troncoso, C., et al.: Decentralized privacy-preserving proximity tracing, April 12 2020. https://www.github.com/dp-3t/documents/raw/master.DP3T%20White%20Paper.pdf
10. Watts, D.: Covidsafe, Australia's digital contact tracing app: the legal issues. Australia's Digital Contact Tracing App: The Legal Issues (May 2, 2020) (2020)
11. Zafari, F., Gkelias, A., Leung, K.K.: A survey of indoor localization systems and technologies. IEEE Commun. Surv. Tutorials **21**(3), 2568–2599 (2019)