A Practical Fog-based Privacy-preserving Online Car-hailing Service System

Jianfei Sun, Guowen Xu, Tianwei Zhang, Mamoun Alazab, Robert H. Deng, *Fellow, IEEE*

***Abstract*—Aiming for minimizing passengers waiting time and vehicles vacancy rate, online car-hailing service systems with fog computing has been deployed in various scenarios. In this paper, we focus on addressing the security and privacy issues in such a promising system by customizing a new cryptographic primitive to provide the following security guarantees: (1) private, fine-grained and bilateral order matching between passengers and drivers; (2) authenticity verification of passengers orders in the form of ciphertext, and (3) temporal assurance of passengers' ciphertext orders. To the best of our knowledge, no previous system has been designed to meet all three requirements. Existing cryptographic primitives (including forward/puncturable encryption (FE/PE) and attribute based matchmaking encryption (AB-ME)) may be leveraged to partially address some of challenges, but there lacks a comprehensive solution. Moreover, the integration of existing works is hampered by the heterogeneity and the weak coupling between distinct cryptographic primitives. As a result, it is infeasible to directly exploit them for the online car-hailing service. To tackle that, we put forward a new cryptographic primitive called Fine-grained Puncturable Matchmaking Encryption (`FP-ME`) by modifying AB-ME and incorporating PE technology. `FP-ME` can simultaneously implement fine-grained and bilateral order matching, the authenticity of passengers orders, and meeting the time constraint of passengers orders. We formalize the adversarial models for the proposed `FP-ME` and then present rigorous security analysis to prove the security of the proposed system. Additionally, we study performance of the system via simulations to demonstrate its practicability and effectiveness in the real-world applications.**

***Index Terms*—fine-grained and bilateral order matching, temporal assurance, authenticity.**

I. INTRODUCTION

WITH the advent of the mobile Internet era, mobile applications have become the first preference for people to deal with all aspects of life [1]–[3]. As one of frequently and widely used applications, fog-based online car-hailing apps, such as Uber, Didi and Lyft, have emerged as among the most popular platforms to render on-demand transportation service [4]–[6]. To hire vehicles, passengers simply need to type in their targeted pickup location and destination in the app and submit the request to a service platform. In response, the platform generally uses fog nodes to either forward the request to some drivers close to the pickup location, or directly schedule a close-by driver to take the order. Compared to the traditional "besiege and chase" trip mode, the online car-hailing mode not only is much more convenient, efficient and flexible for passengers, but also greatly alleviates the burdens of drivers about prospecting for customers all the time.

Jianfei Sun, Guowen Xu and Tianwei Zhang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore; (email: {jianfei.sun,guowen.xu,tianwei.zhang}@ntu.edu.sg). Guowen Xu is the corresponding author.

Mamoun Alazab is with the College of Engineering, IT and Environment, Charles Darwin University, NT0810, Australia. (email: alazab.m@ieee.org).

Robert. H Deng is with the School of Computing and Information Systems, Singapore Management University, Singapore; (email: robertdeng@smu.edu.sg).

A. Security and Privacy Issues

Despite the great convenience and benefits for both drivers and passengers brought by online car-hailing service platforms, there exist distinct types of critical security and privacy challenges to be tackled [7]–[9]. In this paper, we consider the following three potential problems as well as the corresponding security requirements.

(1) Privacy and authenticity of passengers' orders. In an online car-hailing system, passengers' order messages are often sensitive and should be properly encrypted so that only legitimate recipients can access them. Moreover, the authenticity of the order information should be effectively verified. Obviously, a malicious user may intercept the wireless communication signal to eavesdrop, delete, edit the order message of a legitimate user, or disguise himself as a legitimate user to intentionally send some malicious order requests. Once the user's order is completed in the case of malicious manipulation, it easily gives rise to disputes between passengers and drivers.

(2) Private, fine-grained and bilateral order matching between passengers and drivers. Considering that in practice, passengers sometimes would not like to choose vehicles that the online car-hailing platform recommends, and they usually prefer to selectively designate some vehicles which meet their requirements, such as " proximity to pickup location", "punctuality", "high-end car model", "high positive rating" and "low expense", to take orders for the better trip experience. Correspondingly, drivers also hope to specify some requirements, such as "close pick up location", "high praise rating", "high expense", "appropriate destination", and "additional gratuity", to filter out some ideal passengers to take trips. Obviously, such a personalized service is feasible in plaintext. However, since the personalized settings of passengers and drivers may contain sensitive information about each other, this requires us to implement the above requirements in ciphertext. Unfortunately, the unlinkability between ciphertexts makes it very difficult to perform matching between messages.

(3) The timeliness of passengers' ciphertext orders. When passengers forward the ciphertext requests via the system platform for finding the suitable drivers, they usually hope that the order requests can be quickly responded within a certain time period. Once the time to take orders exceeds a certain threshold, the order would be invalid. As a result, the plaintext information of the invalid orders should not be accessed by anyone except the data sender. In this case, the passenger needs to regenerate a new order if it still needs to be served. Of course, delivery of newly generated order also needs to satisfy the property of timeliness.

B. Challenges

Incompleteness of existing works: to the best of our knowledge, there is no existing effort in the literature to solve the above three problems in the scenario of online car-hailing. Existing cryptographic primitives, including dual policy attribute-based encryption (DP-ABE) [10]–[12], attribute based signcryption (ABSC) [13]–[15], forward/puncturable encryption (FE/PE) [16]–[19], puncturable attribute based encryption (PABE) [20], [21] and attribute based matchmaking encryption (AB-ME) [22], [23] may be utilized to alleviate the above problems; however, these methods are unilateral in nature and only applicable to certain problems (see Section II for detailed discussion). In brief, with the properties of supporting policy-based data access control between entities, although DP-ABE can be employed to achieve fine-grained and bilateral order matching between passengers and drivers, it does not meet the security requirements (1) and (3) mentioned above. Similarly, ABSC can only be used to partially address the security requirements (1) and (2), as it enables to verify the authenticity of the data and provide a one-way order matching. To satisfy the security requirement (3), FE/PE and PABE are feasible solutions since they are both designed to ensure the timeliness of data. As a promising primitive, AB-ME is feasible to address security requirements (1) and (3), due to its inherent properties of supporting data access control and verifiability, however, it cannot guarantee the timeliness of data.

Potential solutions as well as challenges: security requirements (1) to (3) may be settled by the integration of the above-mentioned primitives. A natural approach is to incorporate the principles of AB-ME and PE or PABE and ME to construct fine-grained puncturable matchmaking encryption (`FP-ME`). However, realizing this goal seamlessly and efficiently is non-trivial due to the following challenges: **(a)** constructing such an `FP-ME` scheme is not simply combining the two primitives together (i.e., PABE and ME or AB-ME and PE). For the approach of combining PABE and ME, it is firstly intractable to create a proper encryption key that can ensure the original structure of the PABE-based ciphertext would not be destroyed. This is because the encryption key related to the attributes of a sender (i.e. passenger) commonly affects the ciphertext associated with the access policy; **(b)** another challenge is to guarantee that the access policies built by a passenger and a driver are simultaneously checked to avoid any privacy exposures. The reason of privacy leakage stems from that once an encrypted order is correctly decrypted (resp. is failed to decrypt), then a driver can infer that his/her specified policy is matched completely (resp. is unmatched) by the passenger's attribute information. For the combination approach of AB-ME and PE, there is no straightforward way of integrating the attribute-based decryption key and punctured key due to the fact that these two types of keys are separately produced but must be integrated in a coherent manner depending on the same master secret key. Even though this approach is feasible, the key size would be the sum of the sizes of the punctured keys and attributes in an access policy.

Another possibly technical solution is to incorporate the principles of DP-ABE, ABSC and FE/PE. For the combination of DP-ABE, ABSC and FE, regardless of whether these primitives can be perfectly integrated, this solution is actually impractical. This is because **(a)** all drivers need to send online-requests to the key distribution center for updating their secret keys to decrypt the latest encrypted orders once the previous order is invalid, which apparently results in prohibitive costs for drivers and requires a key distribution center to be always online; **(b)** for the integration of DP-ABE, ABSC and PE, it is possible but pretty intractable to construct such an efficient scheme, which originates from that in addition to these reasons for heterogeneity and low coupling, the complexity and inefficiency of these primitives themselves are also reasons to be considered.

C. Our Contributions

In this paper, we for the first time consider and design a fog-based privacy-preserving online car-hailing service system by using our new cryptographic primitive referred to be as fine-grained puncturable matchmaking encryption (`FP-ME`), which is capable of ensuring the order authenticity, supporting the fine-grained access strategies specified by both participants in encrypted form and realizing the timeliness of users' orders. Specifically, a passenger can encrypt-then-sign his/her order message containing his/her personal information and designate a policy that the order can be publicly authenticated and taken only by his/her ideal drivers. A driver can decrypt the encrypted order if and only if a passenger can be regarded as his/her ideal partner as specified through a policy at the specified time. In brief, the contributions can be summarized in the following.

- *Order source authenticity*. To ensure the authenticity of the order to be transmitted, the proposed `FP-ME` scheme permits a driver to embed a signature associated with a set of attributes into a ciphertext order, such that both the generated order and the designated access policies retrieved and derived by the drivers would not be tampered with, forged or replaced.
- *Fine-grained and bilateral order matching*. To provide fine-grained and bilateral order matching strategy for both participants, the proposed `FP-ME` scheme allows both participants (i.e., senders and recipients) to specify fine-grained access policies for the encrypted order, such that the encrypted order can be revealed in case that both strategies are only matched by the respective counterpart.
- *Timeliness of passengers' ciphertext orders*. To enable the timeliness and confidentiality of the orders, the proposed `FP-ME` scheme empowers a passenger to embed a timestamp to specify the timeliness of his/her order.

Furthermore, once the order expires, it also permits a passenger to regenerate a new order if it still needs to be served. During the whole order implementation, the newly generated instructions can still satisfy the property of timeliness.

- *Security and efficiency*. To realize stronger security, the FP−ME scheme reduces the security assumptions (e.g., both participants are assumed to be malicious), thereby making the security model stronger. Additionally, to achieve higher efficiency, the heavy workload (e.g., signature verification) is migrated to a semi-trusted third party (i.e., fog node) to complete. Both considerations can make FP−ME more practical and feasible in this applied scenario. We give rigorous security proofs to indicate that the FP−ME scheme is secure against chosen plaintext attacks and forgery attacks. Moreover, the simulation results exhibit the security-performance tradeoff of the FP−ME scheme to be applicable in real-world applications.

II. Related Works

In summary, existing approaches used in the scenario of online car-hailing could be roughly divided into three categories: Fine-grained access control, PE/FE-based and ME-based techniques. In this section, we show them separately and compare their strengths and weaknesses.

1) Fine-grained Access Control: To alleviate the existing limitations in the conventional coarse access strategy, the notion of attribute based encryption (ABE) as a versatile and fine-grained cryptographic tool was first formulated by Goal *et al.* [25]. There are two primary variants of ABE: ciphertext-Policy ABE (CP-ABE) [26]–[28] and key-policy ABE (KP-ABE) [29], [30]. In a CP-ABE/KP-ABE, an access strategy is attached to ciphertexts/keys and keys/ciphertexts are associated with a set of attributes. A key is capable of recovering the message hidden in ciphertexts on condition that the set of attributes satisfy the access policy. Though the use of ABE can solve fine-grained access control to the encrypted order for either of two participants in the scenario of online car-hailing, it cannot realize a bilateral order matching strategy with fine-granularity. To handle the one-way fine-grained access control limitation, dual-policy ABE (DP-ABE) [10]–[12] as an aggregation of CP-ABE and KP-ABE was proposed, which enables a sender to choose both an access strategy and an attribute set to encrypt the message, such that the encrypted message can be revealed with the matched recipient's decryption key describing his/her policy and attributes. Correspondingly in the online car-hailing application, the encrypted order can be recovered and taken if both strategies of passengers and drivers are satisfied by the respective counterpart. Indeed, DP-ABE could realize fine-grained and bilateral order matching between passengers and drivers. It however cannot guarantee the order authenticity. Generally speaking, the data source authenticity can be ensured with digital signatures. Attribute-based signcryption (ABSC) [13]–[15] as a potential and state-of-the-art primitive can simultaneously support fine-grained access control as well as data authenticity, which of course can ensure the order authenticity. But the fly in the ointment is that it can neither support fine-grained bilateral access strategy nor provide lightweight decryption operations for mobile users.

2) Forward Security: Forward security is a desirable property that can perfectly ensure the confidentiality of past messages even if someone has the current compromised decryption keys. This feature can realize the timeliness of user orders in the scenario of online car-hailing. Currently, most cryptographic primitives to feature this property can be categorized into two flavors: forward encryption and puncturable encryption. Forward encryption (FE) [16], [17] can achieve that the user's key is invalid after a time period t to revoke the decryption capability of any ciphertext encrypted prior to t. Following FE, many FE-related works were proposed, such as forward secure identity-based encryption [31], forward secure ABE [32] and forward-secure predicate encryption [33], etc. However, all of these public key cryptosystems based on FE have a common defect: they either depend on producing many keys or frequently utilizing interactive forward secure protocols. Puncturable encryption (PE) [18], [19] as a remarkable technique enables the non-interactive forward secure property. Specifically, the secret keys in PE are puncturable to revoke the decryption capability for selected messages, receivers and time periods, thus achieving the privacy-protection of the past messages even though the compromise of the current secret key occurs. Distinct from the conventionally interactive FE technology, with the PE technique, authorized users can update their keys by themselves in case of no assistance from a third party. To date, PE-related proposals have also been enriched, such as puncturable identity-based encryption [34], puncturable ABE (PABE) [20], [21], etc. Although parts of PE-related works like PABE to be deployed in online car-hailing can realize one-way fine-grained access policy and the timeliness over encrypted order, the order source authenticity and bilateral access order matching cannot be ensured.

3) Matchmaking Encryption: Matchmaking encryption (ME) [22], [24] as a novel promising paradigm was proposed to solve the problem that both participants in the secret handshake protocol (SH) need to interact online at all times. In this scenario, both senders and recipients can designate strategies the other participant must meet in order for the encrypted plaintext to be recovered. Besides, a signature is attached to a ciphertext and the sender's access strategy such that the data authenticity can be guaranteed. During the whole process, both participants do not require to interact with their respective counterpart. To be applied to the online car-hailing scenario, these nice characteristics enable the authenticity of the order and realize the bilateral access order matching simultaneously. To further enrich ME, a fine-grained ME named attribute-based ME (ABME) [23] was recently proposed. In addition to inheriting the advantages of the original ME primitive, it can implement fine-grained bilateral access strategies to the encrypted message. However, all of the existing ME-related proposals cannot ensure forward secure messaging. Namely, if they are deployed in the application of online car-hailing, the timeliness of user order will fail to achieve.

TABLE I summarizes the characteristic comparisons among existing related solutions and ours. Specifically, only our

TABLE I: Functionality comparisons of our `FP-ME` and other related works

Type of scheme	Fine-grained access control	Bilateral matching	Data authenticity	Outsourced verification	Forward security	Key self-updating
CP-ABE [26]–[28]	✓	✗	✗	✗	✗	✗
DP-ABE [10]–[12]	✓	✓	✗	✗	✗	✗
ABSC [13]–[15]	✓	✗	✓	✓	✗	✗
FS-ABE [32]	✓	✗	✗	✗	✓	✗
ME [22]	✗	✓	✓	✓	✗	✗
MABE [23]	✓	✓	✓	✓	✗	✗
PABE [20], [21]	✓	✗	✗	✗	✓	✓
Our `FP-ME`	✓	✓	✓	✓	✓	✓

Note: CP-ABE: ciphertext-policy attribute based encryption; DP-ABE: dual-policy ABE; ABSC: attribute-based signcryption; FS-ABE: forward secure ABE; ME: matchmaking encryption; PABE: puncturable ABE; `FP-ME`: fine-grained puncturable ME.

`FP-ME` can support fine-grained bilateral order matching, order authenticity as well as timeliness of the encrypted order while the other proposals just achieve partial features. Compared to other solutions, the CP-ABE proposals [26]–[28] only support fine-grained access control; the DP-ABE works [10]–[12] simply realize both fine-grained access control and bilateral matching; the ABSC proposals [13]–[15] exclusively achieve fine-grained access control and outsourced verification; the FS-ABE primitive [32] gains fine-grained access control and forward security but fails to implement key self-updating; the ME primitive [22] realizes bilateral matching, data authenticity, outsourced verification but is incapable of supporting fine-grained access strategy; the MABE scheme [23] enables all listed functionalities but fails to ensure forward security; the PABE proposals [20], [21] support fine-grained access control and forward security.

III. Problem Statements

In this section, the system & threat models of our proposed fog-based privacy-preserving online car-hailing service system are introduced. Besides, we also present the design requirements and goals of the proposed system.

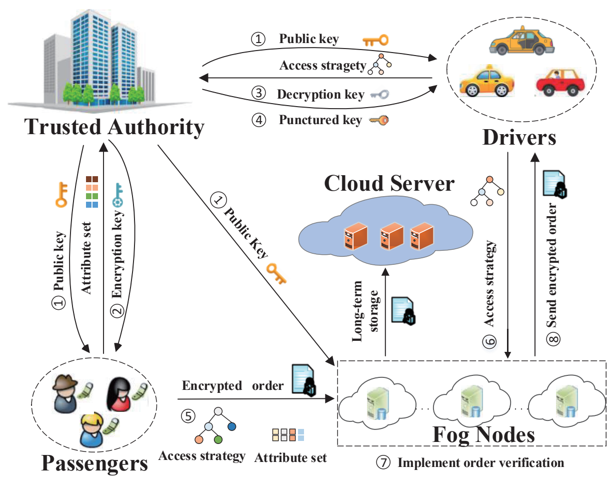


Fig. 1: System Model

A. System and Threat Models

We consider a fog-based privacy-preserving online car-hailing service system that allows the passengers to request for rides from the drivers through the edge server. As presented in Fig. 1, five different roles are included in the designed system: Passengers (Data senders), a cloud server (CS), fog nodes (FNs), a trusted authority (TA) and drivers (Data Recipient). The detailed descriptions for these types of entities are given below:

- TA takes charge of performing certificate verification for system users, initializing the system public keys and then distributing them to other entities (see step. ①). Additionally, it can not only produce the encryption keys for passengers to sign their order information (see step. ②), but also create the decryption keys and punctured keys for drivers for recovering the encrypted target data (see steps. ③ and ④).
- Passengers are deemed as data senders with mobile devices, who encrypt and sign their order information associated with the specified attribute sets with their encryption keys, and then outsource the generated ciphertext to FNs (see step. ⑤).
- FNs are edge servers acting as caches to store encrypted order information or working as intermediaries to communicate with other data recipients. In our model, after making the request for rides from drivers (see step. ⑥), FNs first retrieve their own local storage or interact with other neighbor FNs to assist data senders to find ideal drivers (see step. ⑦). If no responses to the requests are found, FNs forwards the queries to CS.
- Drivers are regarded as data recipients who take orders and provide target passengers with ride services, where the target passengers can be selected by the designated attribute set requirements (see step. ⑧).
- CS is considered as a remote server, which owns enough long-term storage resources to accommodate the data and forwards the shared encrypted data to FNs.

Under the above described models, we assume that the KGC is an entirely trusted entity due to the responsibility of creating the public keys, private keys and punctured keys for corresponding system users. FNs and CS are considered as semi-reliable entities in the sense that they honestly follow system operations but are curious about the information of passengers and drivers, such as pick-up location, destination location, etc. Passengers are deem to be untrustworthy who may pretend to be any other authorized passengers from tampering with, forging, modifying and even generating illegal order messages. Drivers are assumed to be dishonest, who can also pretend to be authorized drivers or collude with the other entities, such as CS or FNs, to obtain the orders from legitimate passengers.

Remark: The passenger picks an access strategy indicating the passengers driver preference and embeds it into the order ciphertext along with his/her own attribute set reflecting the drivers passenger preference. The produced ciphertext order is stored in the clouds or fogs for locally searching and matching. When a driver intends to take orders via an online car-hailing service platform, he/she also chooses and sends an access strategy indicating his/her requirement preference to the clouds/fogs for order matching. After receiving the request, the clouds/fogs proceed with the order matching operation between ciphertext orders and a requested access strategy of a driver by performing the verification algorithm. Since the ciphertext order embeds the attributes of a passenger and a driver sends his/her preference strategy to the clouds/fogs, the ciphertext order is matched successfully if the attributes of a passenger meet the strategy of the driver (i.e., the equality shown in the verification algorithm holds).

B. Design Requirements and Goals

Under aforementioned system and threat models, a fog-based privacy-preserving online car-hailing service system should meet the following requirements and goals:

- **Order confidentiality**: Due to the sensitive information (e.g., location, identity, phone number) involved in an order, the order sent and outsourced to edge clouds or clouds should be in the form of ciphertext and can only be revealed by authorized recipients. Furthermore, the confidentiality of the past order should also be kept once the time to access the order is beyond its timeliness.
- **Privacy-preserving of attributes**: In consideration of the attributes embedded in the ciphertext commonly indicating the special requirements of data senders or recipients, the leakage of attributes give adversaries more clues to snoop the users' privacy as well as more chances to forge a valid ciphertext. Hence, the privacy preserving of attributes should be protected to hinder malicious users from snooping the privacy or forging valid ciphertexts.
- **Efficiency**: In view of the commonly equipped mobile devices with limited resources, the system should be efficient for practical use and support ride-matching in the real-time fashion.
- **Security**: Various attacks can be launched by the cloud server or other authorized users to lead to the system impracticability or even system crashes, such as eavesdropping attacks, impersonate attacks, collusion attacks, etc. The system should be resistant against those threats such that it can prevent malicious adversaries from learning information they shouldn't know. For instance, the CS and FNs should know nothing about the pick-up locations or destinations of the riders or drivers except the final order-matching results.

Remark: It is noting that dynamic attributes are unsuitable for producing attribute-based secret keys. As we all know, the attributes used for private key generation in all attribute-based cryptographic solutions are static attributes. The reason originates from that the attributes used for secret key generation must be certified in advance by a key generation center (*i.e.*, trusted authority), which determines that the attributes are often static attributes. For the real-time dynamic attributes, it is really hard and even impractical for the trusted authority to simultaneously certify all the dynamic attributes of thousands of passengers and drivers. Besides, for a driver who may take orders from various geographically dispersed passengers, it is also inappropriate for a driver to always update his/her private key for taking distinct orders. In our fog-based privacy-preserving online car-hailing service system, the dynamic attributes (*e.g.*, location, waiting time, etc.) along with some necessary information are encrypted to produce the ciphertext order, such that the drivers who satisfy the strategy specified by a passenger can decode the encrypted order.

We also highlight that the privacy-preserving of attributes can be protected by our FP-ME. As described in [22], [23], matchmaking encryption can be thought of as a non-interactive secret handshake (SH) approach, which protects the privacy of both participants identities/attributes. This means that if the handshake successfully occurs between two participants, they can only derive that they both belong to the same group (yet, their identities/attributes are still anonymous to each other), whereas nothing will be leaked if the handshake fails. In our FP-ME, by designating fine-grained access policies to the encrypted data by both participants, a non-interactive SH function before data revealing can be realized to protect the attribute privacy of both participants.

IV. Preliminaries

This section briefly states the basic knowledge that is used for the whole manuscript, which contains the notation description, introduction of complexity assumptions, ABE, PE and the linear secret sharing structure (LSSS), and the outline of FP-ME.

A. Notation

Our FP-ME involves the following notations, which are summarized in TABLE II.

B. Complexity Assumptions

***Definition** 1 (Complexity Assumptions):* FP-ME is reliance on the following complexity assumptions:

- *DBDH Assumption*: Given the tuple $(g, g^{\lambda}, g^{\mu}, g^{\nu}, \mathcal{U})$, where $\lambda, \mu, \nu, \omega in \mathbb{Z}_p$, for the decisional bilinear Diffie-Hellman (DBDH) problem, it is difficult to discern whether the tuple is $(g, g^{\lambda}, g^{\mu}, g^{\nu}, \mathcal{U} = e(g,g)^{\lambda\mu\nu})$ or $(g, g^{\lambda}, g^{\mu}, g^{\nu}, \mathcal{U} = \mathcal{R})$, where $\mathcal{R}$ is a random of $\mathbb{G}_1$.
- *CDH Assumption*: Given the tuple $(g, g^{\lambda}, g^{\mu}, \mathcal{V})$, where λ, μ are also randomly selected from $\mathbb{Z}_p$, for the computational Diffie-Hellman (CDH) problem, calculating $\mathcal{V} = g^{\lambda\mu}$ is hard.

TABLE II: Notations used in FP−ME

Notation	Description
$\Psi_{\mathcal{S}}/\Psi_{\mathcal{R}}$	Universe of attributes of the sender/receiver
$\mathbb{R}$	Receiver's access strategy
$\mathbb{S}$	Sender's access strategy
$\mathcal{R}$	Receiver's attribute set
$\mathcal{S}$	Sender's attribute set
$\mathcal{R} \models \mathbb{R}$	A receiver's attribute set $\mathcal{R}$ satisfies the strategy $\mathbb{R}$
$\mathcal{S} \not\models \mathbb{S}$	A sender's attribute set $\mathcal{R}$ does not satisfy the strategy $\mathbb{S}$
PK/MSK	Public key/master secret key
$\mathsf{DK}/\mathsf{EK}/\mathsf{KP}$	Decryption key/ encryption key/punctured key
$x\|y$	Concatenation of x and y

C. MABE and PE Descriptions

A matchmaking attribute-based encryption (MABE) [23] scheme is a tuple with the following syntax: (ABE.Setup, ABE.KeyGen, ABE.EKGen, ABE.Encrypt, ABE.Decrypt):

- $\mathsf{ABE.Setup}(1^\lambda) \rightarrow (\mathsf{PK}, \mathsf{MSK})$: On input a security parameter 1^λ, output a public key PK and a master secret key MSK.
- $\mathsf{ABE.KeyGen}(\mathsf{MSK}, \mathcal{R}) \rightarrow \mathsf{DK}$: On input the MSK, the attributes $\mathcal{R}$ of a receiver, output a decryption key DK.
- $\mathsf{ABE.EKGen}(\mathsf{MSK}, \mathcal{S}) \rightarrow \mathsf{EK}$: On input the MSK, the attributes $\mathcal{S}$ of a sender, output a encryption key EK.
- $\mathsf{ABE.Encrypt}(\mathsf{PK}, \mathsf{EK}, \mathbb{R}, M) \rightarrow \mathsf{CT}$: On input PK, EK, an access strategy $\mathbb{R}$ and a plaintext M, output a cipheretxt CT.
- $\mathsf{ABE.Decrypt}(\mathsf{PK}, \mathsf{CT}, \mathsf{DK}) \rightarrow M/\perp$: On input the PK, CT, and DK, output the plaintext M or $\perp$ if decryption fails.

A standard puncturable encryption (PE) scheme [22] is also a tuple with the following algorithms: (PE.KeyGen, PE.Encrypt, PE.Puncture, PE.Decrypt):

- $\mathsf{PE.KeyGen}(1^\lambda, \ell) \rightarrow (\mathsf{PK}, \mathsf{DK})$: On input a security parameter 1^λ, and a maximum number of tags per ciphertext ℓ, output a public key PK and an initial decryption key DK.
- $\mathsf{PE.Encrypt}(\mathsf{PK}, \{t_1, \ldots, t_\ell\}, M) \rightarrow \mathsf{CT}$: On input PK, a list of tags $\{t_1, \ldots, t_\ell\}$ and a plaintext M, output a cipheretxt CT.
- $\mathsf{PE.Puncture}(\mathsf{PK}, \mathsf{DK}, t) \rightarrow \mathsf{CT}$: On input PK, DK and a tag t, output a punctured key KP.
- $\mathsf{PE.Decrypt}(\mathsf{PK}, \mathsf{CT}, \{t_1, \ldots, t_\ell\}, \mathsf{KP}, \mathsf{DK}) \rightarrow M/\perp)$: On input the PK, CT, KP and DK, output the plaintext M or $\perp$ if decryption fails.

D. Linear Secret Sharing Scheme (LSSS)

***Definition** 2 (LSSS):* A secret sharing scheme Π over a set of parties $\mathcal{S}$ is regarded to be linear if the following properties hold:

- A vector over $\mathbb{Z}_p$ is formed by the shares for each party;
- There is a secret sharing-generation matrix $\{\mathbb{M}\}_{m\times n}$ for Π and a mapping function ρ is set and used to label row i as $\rho(i)$, where $i \in [1, m]$. In the LSSS structure, when the secret ς in the column vector $\vec{v} = (\varsigma, t_2, t_3, \ldots, t_n)$ is shared, where $t_2, \ldots, t_n \in \mathbb{Z}_p$, then the vector of m shares of the secret ς can be denoted as $\mathbb{M}\vec{v}$, where the party $\rho(i)$ includes the share $(\mathbb{M}\vec{v})_i$.

E. Outline of FP−ME Scheme

***Definition** 3 (FP−ME scheme):* Our FP−ME scheme is constituted by the following algorithms:

- $\mathsf{Setup}(1^\lambda, \ell) \rightarrow (\mathsf{PK,MSK})$: TA accepts the maximum number of tags per ciphertext ℓ, the security parameter λ and returns a system public key PK and a master secret key MSK.
- $\mathsf{EKGen}(\mathsf{MSK}, \mathcal{S}) \rightarrow \mathsf{EK}$: TA generates an encryption key EK based on the sender's attribute set $\mathcal{S}$ and MSK.
- $\mathsf{DKGen}(\mathsf{MSK}, \mathbb{R}) \rightarrow (\mathsf{DK}, \mathsf{KP}_0)$: TA accepts the receiver's access control $\mathbb{R}$, MSK and returns a decryption key DK and an initially punctured key KP_0.
- $\mathsf{Puncture}(\mathsf{KP}_{i-1}, \mathsf{PK}, t) \rightarrow \mathsf{KP}_i$: Data receiver outputs a new punctured key KP_i according to the existing key KP_{i-1}, PK and a tag t.
- $\mathsf{Encrypt}(\mathsf{EK}, \mathcal{R}, \mathcal{S}', \{t_1, \ldots, t_\ell\}, M) \rightarrow \mathsf{CT}$: Data sender generates a ciphertext CT based on his/her own encryption key EK, the receiver's attribute set $\mathcal{R}$, the sender's attribute set $\mathcal{S}'$, a list of tags $\{t_1, \ldots, t_\ell\}$, and a message M.
- $\mathsf{Verify}(\mathbb{S}, \mathsf{CT}) \rightarrow 0$ or 1: Edge/cloud server returns 1 if and only if $\mathcal{S} \models \mathbb{S}$; otherwise it returns 0 based on a sender's policy $\mathbb{S}$ and a ciphertext CT associated with the sender's attribute set $\mathcal{S}$.
- $\mathsf{Decrypt}(\mathsf{DK}, \mathsf{CT'}, \mathsf{KP}_i) \rightarrow (M, \perp)$: Data receiver retrieves the original message M or returns a symbol $\perp$ indicating a decryption failure according to DK, CT' and the current punctured key KP_i.

F. Security Models

This section formalizes two security games to prove that FP−ME can not only achieve the indistinguishability under chosen-plaintext attacks (IND-CPA), but also be existentially unforgeable under chosen message attacks (EUF-CMA) in the random oracle model.

***Definition** 4:* If all polynomial time adversaries have negligible advantages in winning the following IND-CPA game, then FP−ME is secure in the oracle random model. $\mathcal{A}$'s advantage is defined as $Adv = |\Pr[\zeta = \zeta'] - \frac{1}{2}|$.

- **Init**: A challenge attribute set $\mathcal{R}^*$ and a set of tags $t_1^*, \ldots, t_\ell^*$ are picked and declared by adversary $\mathcal{A}$.
- **Setup**: Two empty sets P, C and a counter $(n = 0)$ are initialized by the challenger $\mathcal{B}$. Then, $\mathcal{B}$ runs **Setup** to generate the public parameter PK and the master secret key MSK. Finally, $\mathcal{B}$ returns PK to $\mathcal{A}$.
- **Phases 1 & 2**: $\mathcal{A}$ is allowed to make secret key and current punctured key queries repeatedly. As a response, $\mathcal{B}$ replies to the following queries:
  - $\mathcal{A}$ makes the queries of secret key for an access policy $\mathbb{R}$, where $\mathbb{R} \not\models \mathcal{R}^*$. $\mathcal{B}$ conducts $\mathsf{DKGen}(\mathsf{MSK}, \mathsf{PK}, \mathbb{R}) \rightarrow (\mathsf{DK}, \mathsf{KP}_0)$.

- $\mathcal{A}$ makes current punctured key queries for a tag t where $t \notin \{t_1^*, \ldots, t_\ell^*\}$. $\mathcal{B}$ first increments n, conducts $\mathsf{Puncture}(\mathsf{KP}_0, \mathsf{PK}, t) \rightarrow \mathsf{KP}_n$ and adds t to the set P.
- **Corrupt**() is called for the first time when $\mathcal{A}$ makes requests for the punctured key query. After that, as a response, $\mathcal{B}$ gives the latest punctured key KP_n to it and sets $P \rightarrow C$. Note that all the subsequent queries give $\perp$. Here one restrictive condition is that **Corrupt**() outputs $\perp$ if $(t_1^*, \ldots, t_\ell^*) \cup P = \emptyset$.

- **Challenge**: Two equal-length plaintext messages M_0 and M_1 are picked and submitted to $\mathcal{B}$. A coin ζ is then flipped by $\mathcal{B}$ to choose a random message used for producing the challenge ciphertext. After that, $\mathcal{B}$ conducts $\mathsf{Encrypt}(\mathcal{R}^*, \mathcal{S}', \{t_1^*, \ldots t_\ell^*\}, M_\zeta) \rightarrow \mathsf{CT}^*$.
- **Guess**: A guess $\zeta' \in \{0,1\}$ is responded by $\mathcal{A}$. If $\zeta = \zeta'$, $\mathcal{B}$ will return"1" as its response. Otherwise, $\mathcal{B}$ returns "0".

***Definition** 5:* A polynomial time forger $\mathcal{F}$ in FP-ME scheme can break the proposed FP-ME scheme if the advantage in winning the following EUF-CMA game is non-negligible.

- **Init**: A challenge attribute set $\mathcal{S}'^*$ picked by a forger $\mathcal{F}$ is sent to the challenger $\mathcal{C}$. It is worth noting that the forgery signature contains the embedded $\mathcal{S}'^*$.
- **Setup**: The **Setup** algorithm is then performed by $\mathcal{C}$ to produce PK and MSK after receiving $\mathcal{S}'^*$. After that, $\mathcal{F}$ obtains the PK sent by $\mathcal{C}$.
- **Query Phase**: $\mathcal{F}$ makes encryption queries on $(m, \mathcal{S}')$ and $\mathcal{S}$. For these queries, the encryption key EK is produced and given to $\mathcal{F}$.
- **Forgery**: A signature σ^* for the m^* and $\mathcal{S}'^*$ is created.

The forger $\mathcal{F}$ wins the EUF-CMA game if the signature σ^* is a valid signature. Noting that in this game, the queries $(m^*, \mathcal{S}'^*)$ have not been made and no attribute set $\mathcal{S}^*$ satisfying $\mathcal{S}^* \models \mathcal{S}'^*$ has been sent to $\mathcal{C}$.

V. Our FP-ME-based Online Car-Hailing Service System

In this section, we first propose a novel practical primitive named fine-grained puncturable matchmaking encryption (FP-ME), which enables data source authenticity, bilateral strategy matching and forward secure messaging. Then, we use our proposed FP-ME as a building block to construct a fog-based privacy-preserving online car-hailing service system.

A. Concrete Construction of FP-ME

- Setup: It chooses a bilinear group $\mathbb{B} = (\mathbb{G}_0, \mathbb{G}_1, p, e, g)$. Then, it selects $\alpha, \beta, \tau \in \mathbb{Z}_p$ at random, and chooses four different hash functions H_1, H_2, H_3, H_4, where $H_1 : \Psi_{\mathcal{S}} \rightarrow \mathbb{G}_0$, $H_2 : \Psi_{\mathcal{R}} \rightarrow \mathbb{G}_0$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_0$ and $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. Next, it selects $\tau \in \mathbb{Z}_p$, $h_1, \ldots, h_\ell \in \mathbb{G}_0$ and samples, computes a degree-ℓ Lagrange polynomial $q(i)$, where $i \in [1, \ell]$ and subject to the constraint that $q(0) = \tau$. Finally, it defines $U(x) = g^{q(x)}$ and t_0 denotes a distinguish tag not used during normal operations, returns PK and MSK as below:
$$\mathsf{PK} = (\mathbb{B}, \{H_i\}_{i\in[1,4]}, \{U(i), h_i\}_{i\in[1,\ell]}, \{e(g,g)^j\}_{j\in\{\alpha,\beta\}}),$$
$$\mathsf{MSK} = (g^\alpha, g^\beta, \tau).$$
- EKGen: It parses the attribute set of the sender as $\mathcal{S} = (att_{\mathcal{S},1}, att_{\mathcal{S},2}, \ldots, att_{\mathcal{S},d})$ and selects $s_i \in \mathbb{Z}_p$ at random. Then, it produces the encryption key $\mathsf{EK} = (ek_{1,i}, ek_{2,i})$ as below: $ek_{1,i} = \{g^\alpha H_1(att_{\mathcal{S},i})^{s_i}\}_{i\in d}$, $ek_{2,i} = \{g^{s_i}\}_{i\in d}$.
- DKGen: It parses the access control of the receiver as $\mathbb{R} = (\mathbb{M}, \pi)$, where $\mathbb{M} \in \mathbb{Z}_p^{n_\mathbb{M} \times m_\mathbb{M}}$ is a matrix and $\pi : [n\mathbb{M}] \rightarrow \Psi_{\mathcal{R}}$ is a mapping function. Next, it selects $r, r_\sigma, r_1, ..., r_{n_\mathbb{M}} \in \mathbb{Z}_p$ at random such that $\overrightarrow{x} = (\beta + \tau r_\sigma, x_2, ..., x_{m_\mathbb{M}})^\perp \in \mathbb{Z}_p^{m_\mathbb{M} \times 1}$, and calculates $\overrightarrow{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_{n_\mathbb{M}}) = \mathbb{M}\overrightarrow{x}$. Finally, it returns $\mathsf{DK} = ((\mathbb{M}, \pi), \{dk_{1,i}, dk_{2,i}\}_{i\in n_\mathbb{M}})$ and an initial punctured key $\mathsf{KP}_0 = (\mathsf{KP}_{0,1}, \mathsf{KP}_{0,2}, \mathsf{KP}_{0,3}, \mathsf{KP}_{0,4})$:
$$dk_{1,i} = g^{\lambda_i} H_2(\pi(i))^{r_i}, \quad dk_{2,i} = g^{r_i}, \mathsf{KP}_{0,1} = (g^\tau)^{r+r_\sigma},$$
$$\mathsf{KP}_{0,2} = U(H_4(t_0))^r, \mathsf{KP}_{0,3} = g^r, \mathsf{KP}_{0,4} = t_0.$$
- $\mathsf{Puncture}$: On input an existing key KP_{i-1} as $\{\mathsf{KP}_0, \mathsf{KP}_1, \ldots, \mathsf{KP}_{i-1}\}$, it randomly chooses $\lambda'_1, \ldots, \lambda'_i, \lambda', v_0, v_1$ from $\mathbb{Z}_p$, such that $\lambda' = (\lambda'_1 + \lambda'_2 + \ldots + \lambda'_i)/i$. Then, it outputs a new punctured key $\mathsf{KP} = (\mathsf{KP}'_0, \mathsf{KP}_1, \ldots, \mathsf{KP}_{i-1}, \mathsf{KP}_i)$ as follows:
$$\mathsf{KP}'_{01} = \mathsf{KP}_{0,1} \cdot g^{\tau(v_0 - \lambda')} = (g^\tau)^{r + r_\sigma + v_0 - \lambda'},$$
$$\mathsf{KP}_{i,1} = (g^\tau)^{(v_1 + \lambda'_i)/i},$$
$$\mathsf{KP}'_{0,2} = \mathsf{KP}_{0,2} \cdot U(H_4(t_0))^{v_0} = U(H_4(t_0))^{r+v_0},$$
$$\mathsf{KP}_{i,2} = V(H_4(t))^{v_1/i}, \mathsf{KP}'_{0,3} = \mathsf{KP}_{0,3} \cdot g^{v_0} = g^{r+v_0},$$
$$\mathsf{KP}_{i,3} = g^{v_1/i}, \mathsf{KP}'_{0,4} = t_0, \mathsf{KP}_{i,4} = t.$$
- $\mathsf{Encrypt}$: It first parses the attribute set of the receiver as $\mathcal{R} = (att_{\mathcal{R},1}, att_{\mathcal{R},2}, \ldots, att_{\mathcal{R},f})$ and the attribute set of the sender as $\mathcal{S}' = (att_{\mathcal{S},1}, att_{\mathcal{S},2}, \ldots, att_{\mathcal{S},d'})$, where $\mathcal{S}' \subseteq \mathcal{S}$. Then, it selects $s, s'_i, r'' \in \mathbb{Z}_p$, a collection of tags $t_1, ..., t_\ell \in \{0,1\}^* \backslash \{t_0\}$ at random and computes as follows, where these tags $t_1, ..., t_\ell$ are public information and their spaces can be considered as the unique identifier or timestamps supplementing the identity of the sender [18], [20]:
$$c_0 = m \cdot e(g,g)^{\beta s}, c_1 = g^s, c_{2,i} = H_2(att_{\mathcal{R},i})^s, c_{3,j} = U(H_4(t_j)), c_{4,i} = ek_{2,i} \cdot g^{s'_i} = g^{s_i + s'_i}, c_5 = g^{r''}.$$
A binary string c_{1-5} is denoted as $c_{1-5} = c_0||c_1||c_{2,i}||c_{3,j}||c_{4,i}||c_5$. For $i' \in [1, d']$, j exits due to $\mathcal{S}' \subseteq \mathcal{S}$ and $att_{\mathcal{S},i'} = att_{\mathcal{S},j}$, then it calculates $c_{6,i'}$ as follows:
$$ek_{1,i'} = ek_{1,j} \cdot H_1(att_{\mathcal{S},i'})^{s'_i} = g^\alpha H_1(att_{\mathcal{S},i'})^{s_i + s'_i}, c_{6,i'} = ek_{1,i'} \cdot H_3(c_{1-5})^{r''} = g^\alpha H_1(att_{\mathcal{S},i'})^{s_i + s'_i} H_3(c_{1-5})^{r''}.$$
Finally, it generates a ciphertext $\mathsf{CT} = ((\mathcal{S}, \mathcal{R}'), c_0, c_1, \{c_{2,i}\}_{i\in[1,f]}, \{c_{3,j}\}_{j\in[1,\ell]}, c_5, \{c_{4,i}, c_{6,i}\}_{i\in[1,d']})$.
- Verify: It first parses the access control of the sender as $\mathbb{S} = (\mathbb{W}, \omega)$, where $\mathbb{W} \in \mathbb{Z}_p^{n_\mathbb{W} \times m_\mathbb{W}}$ is a matrix and

$\omega : [n_{\mathbb{W}}] \rightarrow \Psi_{\mathcal{S}}$ is a mapping function. Next, it selects $\overrightarrow{z} = (1, z_2, ..., z_{m_{\mathbb{W}}})^{\perp} \in \mathbb{Z}_p^{m_{\mathbb{W}} \times 1}$ at random and then calculates $\overrightarrow{\xi} = (\xi_1, \xi_2, ..., \xi_{n_{\mathbb{W}}}) = \mathbb{W}\overrightarrow{z}$. At last, it picks $\{\rho_i\}_{i \in I}$, where I represents the sender's attribute set as $I = \{i | i \in [n_{\mathbb{W}}], \omega(i) = \mathcal{S}\}$ and calculates $\sum_{i \in I} \mathbb{W}\rho_i = (1, 0, ..., 0)$.

$$\prod_{i \in I} \left(\frac{e(c_{6,i}, g)}{e(H_1(att_{\mathcal{S},i}), c_{4,i}) \cdot e(H_3(c_{1-5}), c_5)} \right)^{\xi_i \rho_i} = e(g, g)^{\alpha}.$$

If the above equality holds, it returns 1. Otherwise, it aborts and returns 0.

- Decrypt: It takes DK associated with a policy of the receiver, $\mathbb{R} \in \mathcal{P}_{\mathcal{R}}$, KP, CT and tags $\{t_1, ...t_\ell\}$ as input. Next, let L represent the receiver's attribute, where $L = \{i | i \in [n_{\mathbb{M}}], \pi(i) = \mathcal{R}\}$. It next calculates the corresponding set of reconstruction constants $\{i, v_i\}_{i \in L}$ which satisfies the linear reconstruction property: $\Sigma_{i \in L} v_i \lambda_i = \beta + \tau r_\sigma$ and it then performs the calculation as below:

$$\begin{aligned} \mathsf{A} &= \prod_{i=1}^{L} \left(\frac{e(dk_{1,i}, c_1)}{e(dk_{2,i}, c_{2,i})} \right)^{v_i} \\ &= \prod_{i=1}^{L} \left(\frac{e(g^{\lambda_i} H_2(\pi(i))^{r_i}, g^s)}{e(g^{r_i}, H_2(att_{\mathcal{R},i})^s)} \right)^{v_i} \\ &= \prod_{i=1}^{L} e(g, g)^{\lambda_i v_i} = e(g, g)^{(\beta + \tau r_\sigma)s}. \end{aligned}$$

For $j = 0, ..., i$, it parses KP_i as $(\mathsf{KP}_{i,1}, \mathsf{KP}_{i,2}, \mathsf{KP}_{i,3}, \mathsf{KP}_{i,4})$. It next computes a set of coefficients $w_1, ..., w_\ell, w^*$ as below:

$$w^* \cdot q(H_4(\mathsf{KP}_{i,4})) + \sum_{k=1}^{\ell} (w_k \cdot q(H_4(t_k))) = q(0) = \tau.$$

Then, it calculates:

$$\begin{aligned} \mathsf{B} =& \prod_{j=0}^{i} \left(\frac{e(\mathsf{KP}_{j,1}, c_1)}{e\left(\mathsf{KP}_{j,3}, \prod_{k=1}^{\ell} c_{3,k}^{w_k}\right) \cdot e(\mathsf{KP}_{j,2}, c_1)^{w^*}} \right) \\ =& \frac{e\left((g^\tau)^{r + r_\sigma + v_0 - \lambda'}, g^s\right)}{e\left(g^{r+v_0}, \prod_{k=1}^{\ell} U(H_4(t_k))^{w_k}\right) e\left(U(H_4(t_0))^{r+v_0}, g^{s w_k}\right)} \\ & \cdots \frac{e\left((g^\tau)^{(v_1 + \lambda'_i)/i}, g^s\right)}{e\left(g^{v_1/i}, \prod_{k=1}^{\ell} U(H_4(t_k))^{w_k}\right) \cdot e\left(U(H_4(t_0))^{v_1/i}, g^s\right)^{w_k}} \\ =& \frac{e\left(g^{\tau\left(r + r_\sigma + v_0 - \lambda'\right)}, g^s\right)}{e(g^{r+v_0}, g^{\tau s})} \cdots \frac{e\left(g^{\tau\left(\lambda'_i + v_1\right)/i}, g^s\right)}{e\left(g^{v_1/i}, g^{\tau s}\right)} \\ =& e(g, g)^{\tau\left(r_\sigma - \lambda'\right)s} \cdot e(g, g)^{\tau \lambda'_1 s/i} \cdot \ldots \cdot e(g, g)^{\tau \lambda'_i s/i} \\ =& e(g, g)^{\tau r_\sigma s}. \end{aligned}$$

Finally, it recovers the plaintext by computing $M = c_0 \cdot \frac{\mathsf{B}}{\mathsf{A}}$.

B. FP−ME Deployed in Online Car-hailing Service Systems

In this section, the novel primitive FP−ME is exploited as the main building block to construct an online car-hailing service system, which involves five various entities shown in Fig. 2: trusted authority, passengers, drivers, fog node and cloud. In our designed system, trusted authority is mainly responsible to establish system public parameters for the other entities, create encryption keys for passengers, and produce decryption keys and punctured keys for drivers. Passengers are considered as data senders who encrypt their orders and deliver the requests to the fog nodes to find the suitable vehicles. After obtaining the order requests, fog nodes either forward the requests to some drivers close to the pick up location, or directly schedule close-by drivers to take the orders after order verification passes. Besides, fog nodes commonly transmit all received orders to clouds for the long-term storage. After getting the ciphertext order matched or assigned by fog nodes, drivers take the order and recover some information related to the passengers such as pick up location, destination and phone number. Here the fine-grained puncturable matchmaking encryption is denoted as *FP−ME=(FP-ME.Setup, FP-ME.EKGen, FP-ME.DKGen, FP-ME.Puncture, FP-ME.Encrypt, FP-ME.Verify, FP-ME.Decrypt)*. Below we introduce how to integrate FP−ME into an online car-hailing service system within the following four modules:

1) *System initialization*: Before system initialization, system users (i.e., passengers and drivers) require to send their attribute sets to the trusted authority. As a response, trusted authority performs certificate verification to ensure the system users legitimacy. It first conducts the *FP-ME.Setup* algorithm to produce public parameter *PK* for all system participants (see step. ①), invokes the *FP-ME.EKGen* algorithm to generate an encryption key *EK* for a passenger (see step. ②), and runs the *FP-ME.DKGen & FP-ME.Puncture* algorithms to correspondingly create a decryption key *EK* and a punctured key *KP* for a driver (see step. ③).
2) *Order generation and uploading*: A passenger picks an access strategy and a set of time tags indicating his/her requirements and timeliness for drivers and then encrypts his/her order information (such as phone number, the pick-up location and destination) by running the *FP-ME.Encrypt* to generate the ciphertext order (see step. ④). Subsequently, the generated *CT* is delivered to fog nodes for sharing and matching with suitable drivers. Here, we need to illustrate that the order needs to be taken by a driver within a stipulated time. Once the order is not taken when the time is exceeded, it will be invalid. The system supports a passenger to create a new one, which is pretty fitting for the actual applications. In practice, to confirm that the transmitted ciphertext order has been indeed captured, the fog node often gives an acknowledge message to the passenger as its response. Note that with the passengers transport layer security (TLS) handshake [36], the mutual authentication can be realized, which guarantees that the order delivered by the passenger has been forwarded to the target fog node.

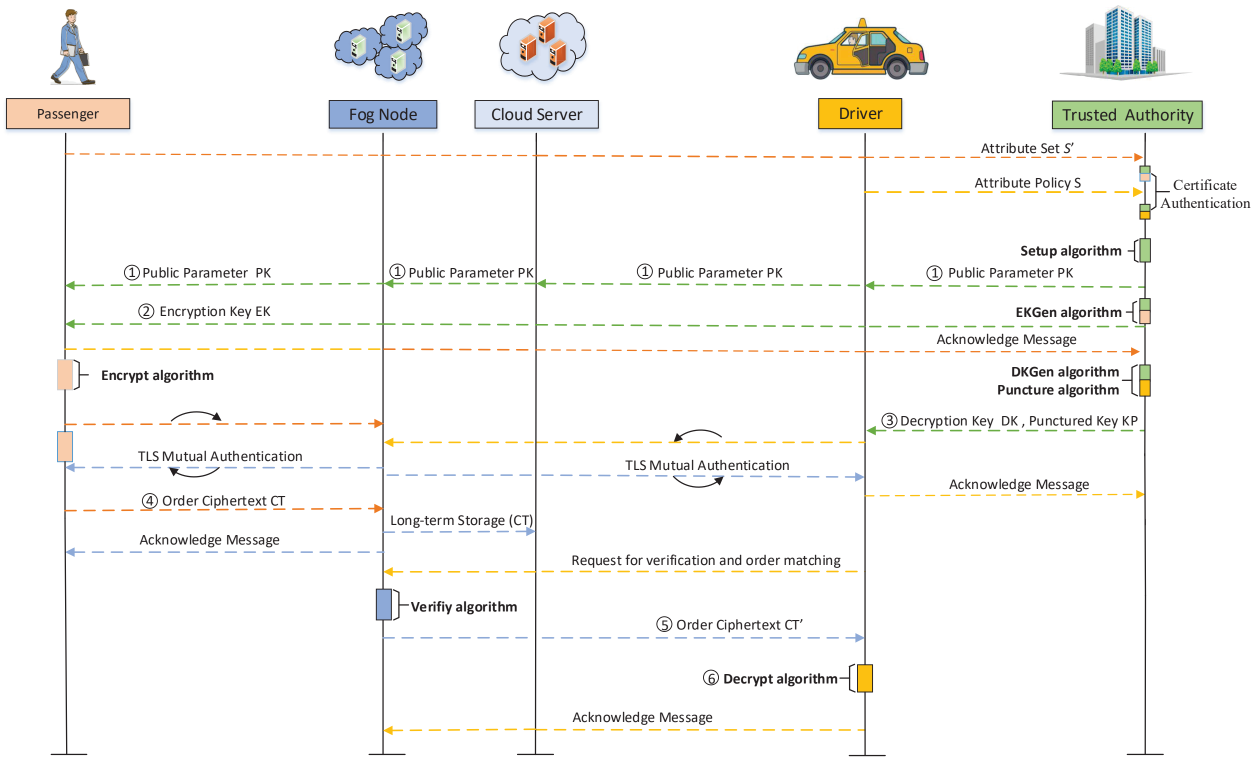


Fig. 2: The whole interaction process of protocol modules in the online car-hailing service system

3) *Order matching and forwarding*: The fog node conducts the authenticity verification task by running *FP-ME.Verify*, finds the suitable driver who matches with the order's requirements specified by the access strategy and sends the resulting cipheretext order *CT'* $= (c_0, c_1, c_{2,i}, c_{3,i})$ to the driver (see step. ⑤). Note that during the matching process, the driver enables specifying his/her access strategy to filter out his/her desirable passengers. Besides, to realize long-term storage of the orders, fog nodes commonly send them to the cloud server for long-time backups. In our FP-ME, the cloud server can theoretically run the *FP-ME.Verify* algorithm for the cross-fog nodes service. The reason is that the verification algorithm only requires the ciphertext and the sender policy as input, and can complete the ciphertext verification, where the ciphertext can be extracted from the fog node and the senders policy is sent by a sender. For the cloud server, which stores all the ciphertexts in each fog node, it is also able to perform the *FP-ME.Verify* algorithm for the cross-fog nodes service after getting the senders policy.
4) *Order recovering and taking*: Once receiving the order ciphertext from the fog node, the driver first recovers the encrypted order to obtain the raw order information (see step. ⑥) and then sends an acknowledge message to the fog node to confirm the order distributed has been taken. Here, TLS handshake protocol is also required to be implemented between drivers and fog nodes for ease of ensuring that the message has been given to the counterpart.

Discussion: Note that if the geographical information is encrypted, the driver could get lots of encrypted orders which may policy-fine but not appropriate because of the distance issue. In our car-hailing service system, the geographical information is naturally sensitive and should be encrypted prior to sending to fog/cloud server. We acknowledge that our solution could lead to additional encrypted orders which may be policy-fine but not appropriate for a driver. However, it is inevitable since our solution aims to realize semantic security for passengers geographical information. Moreover, this issue is somewhat independent of our motivation in this paper, since our approach primarily focuses on how to ensure the authenticity, bilateral matching and timeliness of the passengers order rather than solving the redundant order issue for drivers. To address the problem of filtering out the redundant orders from various passengers, a potential solution is to employ the privacy-preserving computation approach, such as multi-party computation (MPC) [37], [38], garbled circuits (GC) [39], [40] and homomorphic encryption (HE) [41], [42], to securely compute the distance of potential passengers without leaking the geographical information of the participants, thus filtering out the order of the passengers who are far from the drivers distance.

VI. Security Proofs and Analysis

In this section, we present the security proofs to indicate the CPA & EU-CMA security of our proposed `FP-ME` scheme. Next, we also give the security analysis of our constructed online car-hailing service system to prove its resistance to various attacks, which states its practicality and feasibility.

A. Security Proofs of Our FP-ME Primitive

***Theorem** 1:* If the CDH assumption holds, then there does not exist a probabilistic polynomial time forger $\mathcal{F}$ that can break the EU-CMA security of the FP-ME.

Proof: Assuming that an adversary $\mathcal{F}$ attacks our FP-ME successfully with non-negligible advantage, then an algorithm $\mathcal{A}$ could be established to utilize $\mathcal{F}$ to solve the CDH problem. $\mathcal{A}$ is delivered $(A = g^a, B = g^b)$ and asked to return g^{ab}.

Suppose that $\mathcal{F}$ makes queries at most q_{H_1} times to H_1-oracle and q_{H_3} times to H_3-oracle. $\mathcal{A}$ preserves the lists $\mathcal{L}_1$ and $\mathcal{L}_3$ to retain the answers of the oracles H_1 and H_3. Besides, $\mathcal{A}$ picks a random $\sigma \in [1, q_{H_3}]$. If i is delivered for the query of H_1, $\mathcal{A}$ checks the $\mathcal{L}_1$ list and does the following actions below:

- If an entry for the query is checked in $\mathcal{L}_1$, then the same answer is returned to $\mathcal{F}$.
- Otherwise, $\mathcal{A}$ simulates as follows:
  - If $i \in \mathcal{S}^*$, it picks a random $\beta_i \in \mathbb{Z}_p$ and responds the answer $H_1(i) = g^{\beta_i}$.
  - If $i \notin \mathcal{S}^*$, a random $\beta_i \in \mathbb{Z}_p$ is chosen and the answer $H_1(i) = A^{-\beta_i} = g^{-a\beta_i}$ is responded.

If m_i is delivered for the query of H_3, where m_i denotes the content of signing, $\mathcal{A}$ checks the $\mathcal{L}_3$ list and does the following actions below below:

- If an entry for the query is checked in $\mathcal{L}_3$, then the same answer is returned to $\mathcal{F}$.
- Otherwise, $\mathcal{A}$ simulates as follows:
  - If $i \neq \sigma$, it picks random values $\alpha_i, \beta'_i \in \mathbb{Z}_p$ and responds the answer $H_3(i) = A^{\alpha_i} g^{\beta'_i} = (g^a)^{\alpha_i} g^{\beta'_i}$.
  - If $i = \sigma$, a random $\beta'_i \in \mathbb{Z}_p$ is chosen and the answer $H_3(i) = g^{\beta'_i}$ is responded.

Suppose that $\mathcal{F}$ makes queries at most q_e encryption key extraction requests. The encryption key on attribute set $\mathcal{S}$ can be queried in this phase. In the following, how to simulate an encryption key on S is presented. Here, two sets $\mathcal{T}$ and $\mathcal{T}'$ are formalized, where $\mathcal{T} = \mathcal{S} \cap \mathcal{S}'$ and $\mathcal{T} \subseteq \mathcal{T}'$. Next, it simulates the encryption key components $\mathsf{EK} = (ek_{1,i}, ek_{2,i})$ as below.

- For $i \in \mathcal{T}'$, $\mathsf{EK} = (ek_{1,i} = B^{\tau}(H_1(i))^{s_i}, ek_{2,i} = g^{s_i})$, where τ, s_i are randomly chosen from $\mathbb{Z}_p$.
- For $i \notin \mathcal{T}'$, EK can be simulated as $\mathsf{EK} = (g^{ab}(A^{-\beta_i})^{s'_i + \frac{1}{\beta_i}b}, g^{\frac{1}{\beta_i}b + s'_i})$. The key is correctly simulated because $s_i = \frac{1}{\beta_i}b + s'_i$ and $g^{\alpha} H_1(i)^{s_i} = g^{-a\beta_i s'_i} = g^{ab}(g^{-a\beta_i})^{s'_i + \frac{1}{\beta_i}b}$ hold, where $g^{\alpha} = g^{ab}$.

$\mathcal{F}$ also makes requests for the signature query on the context m for an attribute set $\mathcal{S}$. If $H_3(m) \neq g^{\beta_i}$, the signature could be simulated by $\mathcal{A}$ in the following, where m is the content of signing. In order to simulate $(c_{6,i}, c_5, c_{4,i}) = (g^{\alpha} H_1(att_{\mathcal{S},i'})^{s_i + s'_i} H_3(m)^{r''}, g^{r''}, g^{s_i + s'_i})$, it chooses $s'_i, r' \in \mathbb{Z}_p$ and lets $r'' = -\frac{1}{\alpha_{i_d}} b + r'$. Then, it is easy to set $g^{\alpha} H_1(att_{\mathcal{S},i'})^{s_i + s'_i} H_3(m)^{r''} = H_1(i)^{s'_i + s_i} \cdot B^{-\frac{\beta_i}{\alpha_{i_d}}} \cdot (A^{\alpha_{i_d}} g^{\beta_i})^{r'' + \frac{1}{\alpha_{i_d}} b} = H_1(i)^{s'_i + s_i} \cdot (g^b)^{-\frac{\beta_i}{\alpha_{i_d}}} \cdot ((g^a)^{\alpha_{i_d}} g^{\beta_i})^{r'' + \frac{1}{\alpha_{i_d}} b}$, $g^{r''} = (B^{-\frac{1}{\alpha_{i_d}}} g^{r'}) = g^{-\frac{1}{\alpha_{i_d}} b + r'}$ when $H_3(m) = A^{\alpha_{i_d}} g^{\beta_i}$.

Eventually, $\mathcal{A}$ outputs a forged signature σ^* on message m^* for attribute sets $\mathcal{S}^*$. If $H_3(m) \neq g^{\beta_\sigma}$, then $\mathcal{A}$ aborts. Otherwise, it meets the verification equation, which indicates that $\sigma^* = (\sigma^*_{6,i}, \sigma^*_5, \sigma^*_{4,i})_{i \in \mathcal{S}^*} = (c^*_{6,i}, c^*_5, c^*_{4,i})_{i \in \mathcal{S}^*} = (g^{\alpha} H_1(att_{\mathcal{S},i'})^{s_i + s'_i} (H_3(m))^{r''}, g^{r''}, g^{s_i + s'_i})_{i \in \mathcal{S}^*}$.

Thus, $\mathcal{A}$ can then perform the calculation $g^{ab} = \prod_{i \in \mathcal{S}^*} ((\sigma^*_{6,i} / ((\sigma^*_5)^{\beta'_\sigma} (\sigma^*_{4,i})^{\beta_i}))^{\xi_i \rho_i} = (g^{ab} \prod_{i \in \mathcal{S}^*} H_1(i)^{s_i + s'_i} (H_3(m))^{r''} / (\prod_{i \in \mathcal{S}^*} (g^{s_i + s'_i})^{\beta_i} (g^{r''})^{\beta'_\sigma})$ because $H_1(i) = g^{\beta_i}$ and $H_3(m^*) = g^{\beta'_\sigma}$. For $\mathcal{A}$'s success, we need to ensure the forgery signature on content m^* such that $H_3(m^*) = g^{\beta'_\sigma}$.

***Theorem** 2:* If the DBDH assumption satisfies, then there does not exist an adversary $\mathcal{A}$ that breaks the CPA security of the FP-ME with a negligible advantage.

Proof: Assuming that an existing adversary $\mathcal{A}$ can successfully breach our CPA security game with non-negligible advantage, it is easy to build another algorithm $\mathcal{B}$ that can solve the DBDH via the assistance of adversary $\mathcal{A}$. For the tuple $(g, A = g^x, B = g^y, C = g^z)$ given to $\mathcal{B}$, it is asked to determine whether $\Phi = e(g, g)^{xyz}$ or Φ is a random of $\mathbb{G}_1$.

- **Init**: $\mathcal{A}$ sends to $\mathcal{B}$ the challenge attribute set $\mathcal{R}^* = (attr^*_{\mathcal{R},1}, \ldots, attr^*_{\mathcal{R},f})$. $\mathcal{A}$ also gives $\mathcal{B}$ the target tag set $(t_1, t_2, \ldots, t_\ell)$ that it plans to attack.
- **Setup**: $\mathcal{B}$ randomly picks $\beta' \in \mathbb{Z}_p$ and implicitly lets $\beta = xy + \beta'$ by setting $e(g, g)^{\beta} = e(g, g)^{\beta'} e(A, B)$. $\mathcal{B}$ also randomly chooses $\theta_{t_0}, \theta_{t_1}, \ldots, \theta_{t_\ell} \in \mathbb{Z}_p$, in which θ_{t_0} is a special value that will not be used for the normal simulations. Then, $\mathcal{B}$ implicitly lets $q(0) = x$ while $U(t_i) = g^{\theta_{t_i}}$. Next, random values $z_1, z_2, \ldots, z_\ell \in \mathbb{Z}_p$ are picked and $\mathcal{B}$ sets $\{h_i = g^{z_i}\}_{i \in [1, \ell]}$. Finally, the public parameter is set as $\mathsf{PK} = (g, g, U_i, h_i, t_0, e(g, g)^{\beta})$.
- **Phases 1 & 2**: **Case 1:** $\mathbb{R} \vdash \mathcal{R}^*$: $\mathcal{A}$ makes queries for the LSSS access controls and tags. Assuming that the access control $\mathbb{R} = (\mathbb{M}, \Pi)$ and tag t are added to the empty set P, there must exist a vector $\vec{x} = (x_1, \ldots, x_{m_{\mathbb{M}}})$ such that $x_1 = \beta + \tau r_\sigma$, where $r'_\sigma \in \mathbb{Z}_p$, $\tau = x$. For all $\Pi^*(i) \in \vec{x}$, it meets that $\mathbb{R}\vec{x} = \vec{0}$.
  $\mathcal{B}$ chooses $r'_1, \ldots, r'_\ell \in \mathbb{Z}_p$ and calculates $dk_{2,j} = g^{r'_j} B^{x_j} = g^{(r'_j + yx_j)}$, which implicitly sets $r_j = r'_j + y \cdot x_j$. Then, $\mathcal{B}$ simulates $dk_{1,j}$ as $dk_{1,j} = g^{\beta'} (g^x)^{r'_\sigma} (g^{r'_j + yx_j})^{z_j} = g^{xy + \beta' + x(r'_\sigma - y)} (g^{r'_j + yx_j})^{z_j} = g^{\beta} (g^x)^{r_\sigma} H_2(\Pi(j))^{r_j}$, which also implicitly sets $\tau = x$.
  Then, the punctured key is set as $\mathsf{KP}_0 = (\mathsf{KP}_{0,1}, \mathsf{KP}_{0,2}, \mathsf{KP}_{0,3}, \mathsf{KP}_{0,4})$:
  $\mathsf{KP}_{0,1} = (g^x)^{y + r'_\sigma - y} = (g^{\tau})^{r + r_\sigma}$, $\mathsf{KP}_{0,2} = (g^y)^{\theta_{t_0}}$, $\mathsf{KP}_{0,3} = g^y$, $\mathsf{KP}_{0,4} = t_0$. Next, $\mathcal{B}$ increments n and calculates $\mathsf{KP}_n = \textbf{Puncture}(\mathsf{KP}_{n-1}, t)$, and adds t to set P. In the case $\mathcal{R} \models \mathbb{R}$, the following is considered:
  - For **Corrupt**() query and $(t^*_1, \ldots, t^*_d) \cap C = \emptyset$, where C is initialized as an empty set, $\mathcal{B}$ picks $v'_0, v'_1, \lambda'_1, \ldots, \lambda'_i \in \mathbb{Z}_p$ such that $\lambda' = (\lambda'_1 + \ldots + \lambda'_i)/i$ and sets $v_0 = \lambda' + v'_0$ and $v_1 = -\lambda' + v'_1$ implicitly. Thus, it works as follows:

$$\mathsf{KP}'_{0,1} = \mathsf{KP}_{0,1} \cdot g^{x(v_0 - \lambda')} = (g^{\tau})^{r + r_\sigma + v_0 - \lambda'},$$

$$\mathsf{KP}'_{i,1} = (g^x)^{(v_1+\lambda'_i)/i} = (g^\tau)^{(v_1+\lambda'_i)/i},$$
$$\mathsf{KP}'_{0,2} = \mathsf{KP}_{0,2} \cdot (U(H_4(t_0)))^{v_0} = g^{\theta_{t_0}(v_0+r)},$$
$$\mathsf{KP}'_{i,2} = (U(H_4(t_j)))^{v_1/i} = (U(H_4(t_j)))^{(-\lambda'+v'_1)/i},$$
$$\mathsf{KP}'_{0,3} = \mathsf{KP}_{0,3} \cdot g^{v_0} = g^{r+v_0},$$
$$\mathsf{KP}'_{i,3} = g^{v_1/i}, \mathsf{KP}'_{0,4} = t_0, \mathsf{KP}'_{i,4} = t.$$

- **Corrupt**() is called for the first time when $\mathcal{A}$ makes this query. After that, $\mathcal{B}$ gives the latest punctured key KP_n as its response and sets $P \to C$. Note that all the subsequent queries give $\perp$.

Case 2: $\mathbb{R} \not\models \mathcal{R}^*$: Similarly, $\mathsf{DK} = (dk_{1,i}, dk_{2,i})$ can be simulated as that in **case 1**. Besides, the punctured key is set as $\mathsf{KP}_0 = (\mathsf{KP}_{0,1}, \mathsf{KP}_{0,2}, \mathsf{KP}_{0,3}, \mathsf{KP}_{0,4})$: $\mathsf{KP}_{0,1} = (g^x)^{y+r'_\sigma-y} = (g^\tau)^{r+r_\sigma}$, $\mathsf{KP}_{0,2} = (g^y)^{\theta_{t_0}}$, $\mathsf{KP}_{0,3} = g^y$, $\mathsf{KP}_{0,4} = t_0$. Next, $\mathcal{B}$ also increments n and calculates $\mathsf{KP}_n = \textbf{Puncture}(\mathsf{KP}_{n-1}, t)$, and adds t to set P. In the case $\mathcal{R} \not\models \mathbb{R}$, the following is considered:

- **Corrupt**() is called in the first time as that in **case 1** when $\mathcal{A}$ makes requests for the query. After that, as a response, $\mathcal{B}$ gives the latest punctured key KP_n to it and sets $P \to C$. Note that all the subsequent queries give $\perp$.
- **Corrupt**() does not query or $(t_1^*, \ldots, t_d^*) \cap C \neq \emptyset$, where C is initialized as an empty set. $\mathcal{B}$ picks $v_0, v_1, \lambda'_1, \ldots, \lambda'_i \in \mathbb{Z}_p$ such that $\lambda' = \lambda'_1 + \ldots + \lambda'_i$. Thus, it works as follows:

$$\mathsf{KP}'_{0,1} = \mathsf{KP}_{0,1} \cdot g^{x(v_0-\lambda')} = (g^\tau)^{r+r_\sigma+v_0-\lambda'},$$
$$\mathsf{KP}'_{i,1} = (g^x)^{(v_1+\lambda'_i)/i} = (g^\tau)^{(v_1+\lambda'_i)/i},$$
$$\mathsf{KP}'_{0,2} = \mathsf{KP}_{02} \cdot (U(H_4(t_0)))^{v_0} = g^{\theta_{t_0}(v_0+r)},$$
$$\mathsf{KP}'_{i,2} = (U(H_4(t_j)))^{v_1/i},$$
$$\mathsf{KP}'_{0,3} = \mathsf{KP}_{0,3} \cdot g^{v_0} = g^{r+v_0},$$
$$\mathsf{KP}'_{i,3} = g^{v_1/i}, \mathsf{KP}'_{0,4} = t_0, \mathsf{KP}'_{i,4} = t.$$

- **Challenge:** Two messages M_0 and M_1 with the equal length picked by $\mathcal{A}$ are delivered to $\mathcal{B}$. A coin ζ is flipped by $\mathcal{B}$ to choose a random message used for producing the challenge ciphertext, where $\zeta \in \{0,1\}$. After that, $\mathcal{B}$ produces the challenge ciphertext by setting $c_0 = M_\zeta \cdot \Phi, c_1 = g^z, c_{2,j} = g^{zz_j}$, where $s = z$ is set implicitly.
- **Guess**: $\zeta' \in \{0,1\}$ is outputted by $\mathcal{A}$. If $\zeta = \zeta'$, $\mathcal{B}$ will return"1" as its response. Otherwise, $\mathcal{B}$ returns "0".

B. Security Analysis of Our Online Car-hailing Service System

In this part, we analyze the possible attacks in our online car-hailing service system. We mainly consider the following kinds of common attacks: collusion attacks, eavesdropping attacks and impersonation attack, which may undermine the constructed system or lead to the privacy leakage of order ciphertext.

- *Collusion attacks.* Collusion attacks refer to that several malicious parties can intentionally collaborate with each other to derive some clear-text privacy from ciphertext or to breach the system. In our system, there are two types of collusion attacks: one is that malicious passengers may conspire with other passengers to build valid orders without legal encryption keys. Another is that different malicious drivers without decryption keys collude with other drivers to recover the encrypted orders. Since every encryption key is created with a unique randomness, the combination of multiple encryption keys cannot produce a new valid encryption key. Hence, the collusion attacks cannot lead to valid ciphertext generation. Also, since the decryption key is also generated with a unique randomness, any combination of multiple decryption keys cannot create a legal decryption key. Therefore, the collusion attacks lead to the failure of decrypting non-authorized ciphertext order.
- *Eavesdropping attacks.* Eavesdropping attacks refer to that any malicious entities except the order owner (passenger) and legitimate order recipients (drivers) attempt to snoop private data from ciphertext order. In our system, there are also two types of attacks: one is outsider attacks and another is insider attacks. For outsiders who do not own valid decryption keys, they can get the ciphertext order but learn nothing from it. For insiders who are considered as fog nodes, cloud or drivers with invalid decryption key, since fog nodes and cloud also do not have the decryption key, they cannot snoop any private data from the ciphertext order. Besides, although malicious drivers have invalid decryption keys, they attempt to use them to produce a new legitimate decryption key. As a result of the fact that each decryption key is generated with a unique randomness, any combination of multiple decryption keys cannot create a legal decryption key. Hence, the privacy of ciphertext order cannot be leaked resulting by this type of eavesdropping attack.
- *Impersonation attacks.* Impersonation attacks refer to that any malicious entities who have strong motivations to act as legitimate passengers produce valid encryption keys for forging valid ciphertext orders. Similar to the reasons given in the eavesdropping attacks, since the entities including the outsiders, cloud, and fog nodes, have no valid encryption keys in their hands, hence they cannot forge valid encryption keys, thus producing invalid ciphertext order. Besides, malicious drivers also cannot forge valid encryption keys since each encryption key is produced with a unique randomness and an encryption key with at least two random seeds cannot pass the signature verification.

VII. Performance Analysis and Evaluation

In this section, we present comprehensive comparisons between our `FP-ME` and other related works. We first consider the functionality comparisons. Then we theoretically conclude the computation and space complexity. Finally, we give the empirical evaluations.

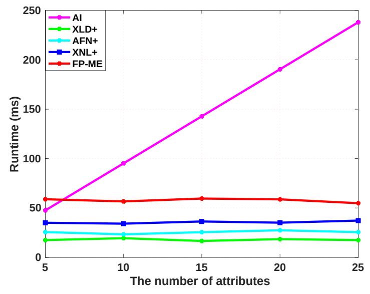
A. Theoretical Performance Analysis

TABLE III presents the comparisons of computation and space complexity among related proposals [10]–[15], [20]–[22], [26]–[28], [32] and our `FP-ME`. Here, we separate the

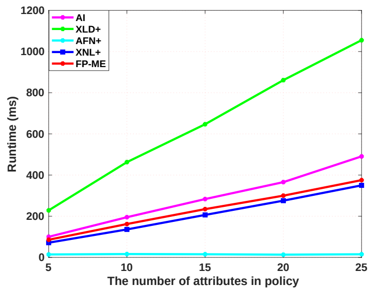
TABLE III: Computation and Space Complexity Comparisons among Existing Approaches.

Proposal	**Computation Complexity**							**Space Complexity**				
	Setup	EKGen	DKGen	Puncture	Encrypt	Verify	Decryption	PK	EK	DK	KP	CT
AI [10]	$\mathcal{O}(\Omega_s+\Omega_r)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	$\mathcal{O}(\Omega_s+\Omega_r)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$
CGW [11]	$\mathcal{O}(\Omega_s+\Omega_r)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	$\mathcal{O}(\Omega_s+\Omega_r)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$
XLD+ [12]	$\mathcal{O}(1)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	$\mathcal{O}(1)$	N/A	$\mathcal{O}(\mathcal{S}+\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$
AFN+ [22]	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{S}\cdot\mathcal{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathbb{R})$	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$
XNL+ [23]	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{S}+\mathcal{R})$	$\mathcal{O}(\mathbb{S})$	$\mathcal{O}(\mathbb{S}+\mathbb{R})$	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathbb{S}+\mathcal{R})$
Our `FP-ME`	$\boldsymbol{\mathcal{O}(\ell)}$	$\boldsymbol{\mathcal{O}(\mathcal{S})}$	$\boldsymbol{\mathcal{O}(\mathbb{R})}$	$\boldsymbol{\mathcal{O}(\mathbb{R})}$	$\boldsymbol{\mathcal{O}(\mathcal{S}+\mathcal{R})}$	$\boldsymbol{\mathcal{O}(\mathbb{S})}$	$\boldsymbol{\mathcal{O}(\mathbb{S}+\mathbb{R})}$	$\boldsymbol{\mathcal{O}(\ell)}$	$\boldsymbol{\mathcal{O}(\mathcal{S})}$	$\boldsymbol{\mathcal{O}(\mathbb{R})}$	$\boldsymbol{\mathcal{O}(\mathbb{R})}$	$\boldsymbol{\mathcal{O}(\mathbb{S}+\mathcal{R})}$
GNS+ [13]	$\mathcal{O}(L)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{S})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(L)$	$\mathcal{O}(\mathbb{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{R})$
YLW+ [14]	$\mathcal{O}(L)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{S})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(L)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{R})$
HZL+ [15]	$\mathcal{O}(L)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{S})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(L)$	$\mathcal{O}(\mathcal{S})$	$\mathcal{O}(\mathbb{R})$	N/A	$\mathcal{O}(\mathcal{R})$
PNX+ [20]	$\mathcal{O}(\ell)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{S})$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\ell)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{R})$
XNH+ [21]	$\mathcal{O}(1)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{S})$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(1)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{R})$
TKN+ [32]	$\mathcal{O}(\ell)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{S})$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(L)$	N/A	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathbb{R})$	$\mathcal{O}(\mathcal{R})$

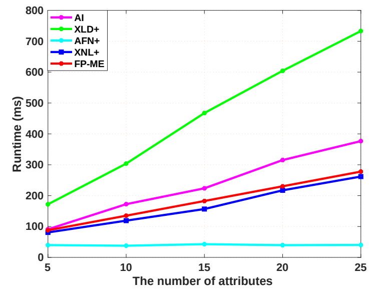
Note: PK: public parameter; EK: encryption key; DK: decryption key; KP: punctured key; CT: ciphertext. ℓ: the number of time tags; L: the number of maximum attributes; N/A: no this function; $\mathcal{S}$: attribute set of data sender; $\mathbb{R}$: receiver's access policy; $\mathcal{R}$: attribute set of data receiver; $\mathbb{S}$: sender's access policy;



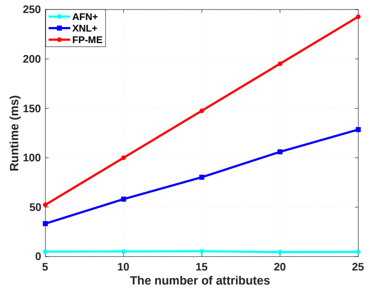
(a) Setup algorithm



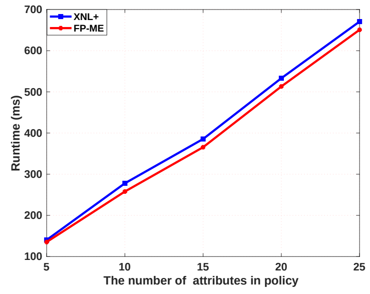
(b) DKGen algorithm



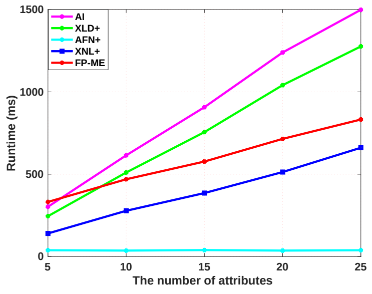
(c) Encrypt algorithm



(d) EKGen algorithm



(e) Verify algorithm

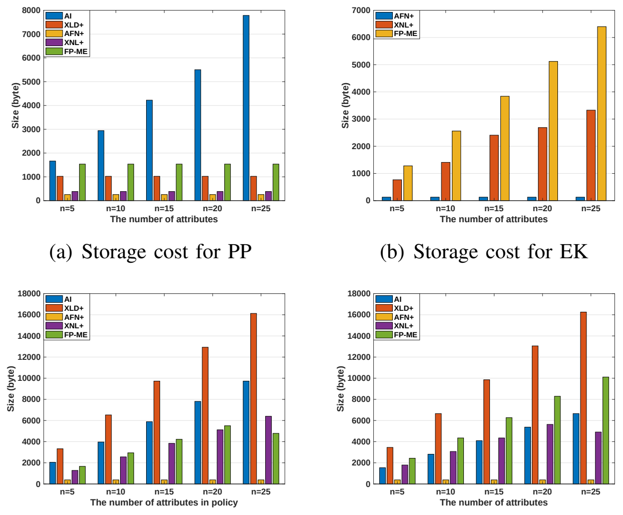


(f) Decrypt algorithm

Fig. 3: Experimental Simulations for algorithm running time

comparisons into two parts: one part is the computation and space complexity comparison among the solutions related to the bilateral strategy matching, another part is the computation and space complexity comparison among the schemes related to the data authenticity and forward security. For the comparisons related to the bilateral strategy matching, it's easy to learn that our `FP-ME` is almost comparable to the proposals [22], [23] in terms of the computation complexity of the EKGen, DKGen, Encrypt, Verify and Decryption algorithms. Besides, our `FP-ME` has a lower computation complexity in the DKGen algorithm than the works [10]–[12] and has a higher computation complexity in the Setup algorithm than [12], [22], [23]. We can also conclude that our `FP-ME` requires almost the same space complexity as [12], [22], [23] in terms of storing EK, DK, CT.

Besides, our `FP-ME` needs a lower space complexity than [10]–[12] in terms of storing DK and CT and requires a higher space complexity than [12], [22], [23] for storing PK. For computation and space complexity comparisons among the works related to the data authenticity and forward security, our `FP-ME` has the same computation complexity of the EKGen, DKGen and Verify algorithms as those in [13]–[15], a lower computation complexity in the Setup phase than [13]–[15], and a higher computation complexity in the Encrypt and Decrypt



(a) Storage cost for PP
(b) Storage cost for EK
(c) Storage cost for DK
(d) Storage cost for CT

Fig. 4: Experimental simulations about the storage cost with the number of attributes n.

phases than [13]–[15], [20], [21], [32]. We also observe that our FP−ME almost has the same space complexity for storing EK and DK as those in [13]–[15], a higher space complexity for storing CT than that in [13]–[15], [20], [21], [32].

B. Experimental Results

In our experimental simulations, the works [10], [12], [22], [23] are chosen to make comparisons with our FP−ME since these experimental works are all based on bilateral access control and aim to realize the most similar functionalities as our FP−ME. We utilize the version of Intellij IDEA-2018.2.5 and Java 8 to evaluate our approach. Besides, we also install the latest JPBC library [35] for the underlying cryptographic operations. In our experiment, a "fog node" is simulated with a Lenovo server, which has a 512SSD, 1TB storage space of hard disk and is deployed on Windows 10 operating system under Intel (R) 8 Core(TM) i7-7820HK CPU @2.9 GHz and 16GB RAM. We replace a user (acting as a passenger or driver) with a Huawei nova3 phone with 6GB RAM, a four-core 2.36GHz Cortex A73 processor and four-core Cortex A53 1.8GHz processor. It is worth noting that fog computing is a special distributed cloud paradigm that extends computation, communication, and storage facilities toward the edge of a network. Compared to traditional clouds, fog nodes can support delay-sensitive service requests from end-users with reduced energy consumption and low traffic congestion. For each fog node in fog-cloud architectures, it is also recognized to have almost the same computing power and enough resources as the cloud. Hence, to better simulate the order encryption and decryption process for end-users (*i.e.*, passengers and drivers), we try our best to allocate all the computing and storage resources of the cloud server to one fog node in our experimental evaluations. For the accuracy simulation, there are two extreme cases with these cryptographic solutions: 1) for the driver who meets all the requirements of a passenger, he/she can take the passengers order with 100% accuracy if he/she is willing to take the order; 2) for the driver who does not match the requirements of a passenger, it is completely impossible to take the order of the passenger (*i.e.*, 0% accuracy). The basic reason leading to the extreme cases is that only the driver matching the requirements of the passengers order can decode-then-take the ciphertext order while the driver who does not satisfy those requirements is filtered out.

For ease of the comparisons of experimental simulations about computation and storage costs, we let the number of attributes (or in policy) vary from 5 to 25. Since only our FP−ME can provide the forward security among the bilateral matching works, here we only simulate the comparisons of all common algorithms except the Puncture algorithm. Besides, due to the fact that the computation and storage cost of our FP−ME proposal is associated with the number of timestamps and attributes (or in policy) while that of other proposals are only related to the number of attributes (or in policy), we set the number of timestamp as 5. In our simulations, Fig. 3 depicts the comparisons of experimental simulations for the running time of each algorithm. Specifically, Fig.3(a)-3(f) successively present the running time comparisons of Setup, DKGen, Encrypt, EKGen, Verify and Decrypt algorithms. Fig. 4 depicts the comparisons of experimental simulations about storage cost. Specifically, Fig.4(a)-4(d) correspondingly show the storage cost comparisons for storing PK, EK, DK and CT.

For the running time comparisons, from Fig.3(a), we can clearly observe that for the Setup algorithm only the running time of the AI [10] is incrementally linear with the number of the attributes while that of other proposals almost remains stable. Besides, we can learn that the running time in FP−ME is much less than that in [10] and slightly more than that in [12], [22], [23]. From Fig.3(b), we easily conclude that except the work AFN+ [10], the running time for decryption key generation in other works including AI [10], XLD+ [12], XNL+ [23] and FP−ME is incrementally linear with the number of attributes of the access policy. We also depict that the running time in FP−ME is a bit higher than that in XNL+ [23] and AFN+ [22] and is much lower than that in AI [10], XLD+ [12]. Besides, as seen from Fig.3(c), it is easy to observe that the running time for ciphertext generation in AI [10], XLD+ [12], XNL+ [23] and FP−ME is linearly incremental to the number of attributes. Besides, we can learn that the running time in FP−ME is higher than that in XNL+ [23], AFN+ [22] and is lower than that in AI [10], XLD+ [12]. For the running time comparison of encryption key generation for signing, we only make comparisons of the time-cost in XNL+ [23], AFN+ [22] and FP−ME since AI [10] and XLD+ [12] are incapable of supporting signing function. From Fig.3(d), It is easy to obtain that the running time for encryption key generation in these three proposals grows linearly with the incremental of the amount of attributes, and the running time of encryption key generation algorithm in FP−ME is indeed higher than that in XNL+ [23], AFN+ [22]. Since only XNL+ [23] and FP−ME can provide outsourced verification, here we only compare the running time of these proposals. From Fig.3(e), we can obviously find that the running time in these two works basically follows an incrementally-linear relationship with the number of attributes of the access policy. We also find that the running time in our FP−ME is slightly lower than that in XNL+ [23]. From Fig.3(f), it is straightforward to see that only AFN+ [22] has a stable running time for conducting the decryption algorithm while the running time in the rest works is all linear with the number of attributes. In addition, FP−ME requires lower running time to implement decryption algorithm than AI [10], XLD+ [12] and needs higher running time to conduct it than XNL+ [23], AFN+ [22].

For the storage cost comparisons, from Fig.4(a), we can clearly see that the storage cost for storing PK in the setup phase in [10] increases with the number of attributes while that in others [12], [22], [23] and our FP−ME basically remains stable. Besides, we can find that our FP−ME has a higher storage cost than [12], [22], [23]. From Fig.4(b), we can conclude that the storage cost for EK generated in the EKGen phase in XNL+ [23] and our FP−ME is incremental to the number of attributes while that in AFN+ [22] keeps stable. We also see that our storage cost is higher than the other schemes [12], [23]. From Fig.4(c), we can find that the storage cost for DK generated in DKGen phase in works [10], [12], [23] and FP−ME increases with the number of attributes in policy while that in AFN+ [22] is always stable. We also see that our storage cost for DK is much lower than [10], [12], [23]. From Fig.4(d), we can also derive that the storage cost for storing CT in the Encrypt phase in works [10], [12], [23] and FP−ME follows an incrementally linear relationship with the number of attributes in the access policy while that in AFN+ [22] almost keeps stable. In addition, we can find that our storage cost is slightly higher than [10], [23] and much lower than [12].

In summary, our FP−ME has a relatively lower running time than AI [10], XLD+ [12] and a slightly higher running time than XNL+ [23] and AFN+ [22]. We also summarize that our FP−ME has a lower storage cost for storing decryption key than [10], [12], [23] and has a slightly higher storage cost for storing ciphertext than [10], [23].

VIII. Conclusion

In this paper, we devised a fog-based privacy-preserving online car-hailing service system via our designed primitive referred as a fine-grained puncturable matchmaking encryption (FP−ME), which elegantly handles several security and privacy issues in this practical use, such as the fine-grained bilateral order matching between passengers and drivers, the authenticity of passengers' orders in ciphertext and the timeliness of passengers' ciphertext orders. The elaborated strict security proofs and analysis indicated that the FP−ME primitive can be applied to construct a practical fog-based privacy-preserving online car-hailing service system. We also studied performance of the system via simulations to demonstrate the practicability and effectiveness of the online car-hailing service system. In the future, we would utilize the bloom filter and the privacy-preserving policy technologies on the basis of our FP−ME to achieve the privacy of the attribute label with its content.

Acknowledgments

This work was supported in part by Nanyang Technological University (NTU)-DESAY SV Research Program under Grant 2018-0980.

References

[1] T. Alexander, W. Mazurczyk, A. Mishra and A. Perotti, "Mobile Communications and Networks", *IEEE Communications Magazine,* vol. 58, no. 3, pp. 54, IEEE, 2020.
[2] M. Shen, B. Ma, L. Zhu, J. Hu, et al., "Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection", *IEEE TIFS,* vol. 13, no.4, pp. 940-953, 2017.
[3] J. Sun, H. Xiong, S. Zhang, et al., "A secure flexible and tampering-resistant data sharing system for vehicular social networks", *IEEE TVT,* vol. 69, no. 11, pp. 12938-12950, 2020.
[4] D. Wang, W. Cao, J. Li, J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks", *2017 IEEE ICDE,* IEEE, pp. 243-254, 2017.
[5] T. Wu, M. Zhang, X. Tian, et al., "Spatial differentiation and network externality in pricing mechanism of online car hailing platform", *International Journal of Production Economics,* vol. 219, pp. 275-283, 2020.
[6] J. Sun, G. Xu, T. Zhang, et al., "Secure Data Sharing With Flexible Cross-Domain Authorization in Autonomous Vehicle Systems", *IEEE TITS*, DOI: 10.1109/TITS.2022.3157309, 2022.
[7] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage", *IEEE TIFS,* vol. 14, no. 2, pp. 331-346, IEEE, 2019.
[8] J. Sun, G. Xu, T. Zhang, et al., "Share your data carefree: An efficient, scalable and privacy-preserving data sharing service in cloud computing", IEEE TCC, DOI: 10.1109/TCC.2021.3117998, 2021.
[9] X. Han, Y. Zhou, K. Chen K, et al., "ADS-Lead: Lifelong Anomaly Detection in Autonomous Driving Systems", *IEEE TITS,* DOI: 10.1109/TITS.2021.3122906, 2022
[10] N. Attrapadung, H. Imai, "Dual-policy attribute based encryption", *ACNS'09,* Springer, Berlin, Heidelberg, LNCS 5536, pp. 168-185, 2009.
[11] J. Chen, R. Gay, H. Wee, "Improved dual system ABE in prime-order groups via predicate encodings", *EUROCRYPT 2015*, Springer, Berlin, Heidelberg, LNCS 9057, pp. 595-624, 2015.
[12] S. Xu, Y. Li, R. H. Deng, et al., "Lightweight and expressive fine-grained access control for healthcare Internet-of-Things", *IEEE TCC,* DOI: 10.1109/TCC.2019.2936481, 2019.
[13] M. Gagne, S. Naraya, R. Safavi-Naini, "Threshold attribute-based signcryption", *International Conference on Security and Cryptography for Networks,* Springer, Berlin, Heidelberg, pp. 154-171, 2010.
[14] J. Yu, S. Liu, S. Wang, et al., "LH-ABSC: A Lightweight Hybrid Attribute-based Signcryption Scheme for Cloud-Fog Assisted IoT", *IEEE IoTJ,* DOI: 10.1109/JIOT.2020.2992288, IEEE, 2020.
[15] C. Hu, N. Zhang, et al. , "Body area network security: a fuzzy attribute-based signcryption scheme", *IEEE JSAC,* vol. 31, no. 9, pp. 37-46, 2013.
[16] R. Canetti, S. Halevi, J. Katz, "A forward-secure public-key encryption scheme", *Journal of Cryptology,* vol. 20, no. 3, pp. 265-294, 2007.
[17] K. Kasamatsu, T. Matsuda, K. Emura, et al., "Time-specific encryption from forward-secure encryption", *SCN'12,* Springer, pp. 184-204, 2012.
[18] M. D. Green, I. Miers, "Forward secure asynchronous messaging from puncturable encryption", *IEEE S & P'15*, IEEE, pp. 305-320, 2015.
[19] T. V. Phuong, W. Susilo, J. Kim J, et al., "Puncturable Proxy Re-Encryption Supporting to Group Messaging Service", *ESORICS'2019,* Springer, Cham, LNCS 11735, pp. 215-233, 2019.
[20] T. V. Phuong, R. Ning, C. Xin C, et al., "Puncturable attribute-based encryption for secure data delivery in internet of things", *IEEE ICC'18*, IEEE, pp. 1511-1519, 2018.
[21] L. Xue, J. Ni, C. Huang, et al., "Forward Secure and Fine-grained Data Sharing for Mobile Crowdsensing", *2019 17th International Conference on Privacy, Security and Trust (PST)*, IEEE, pp. 1-9, 2019.
[22] G. Ateniese, D. Francati, D. Nunez, et al., "Match me if you can: matchmaking encryption and its applications", *Crypto'19* Springer, Cham, LNCS 11693, pp. 701-731, 2019.
[23] S. Xu, J. Ning, Y. Li, et al., "Match in my way: Fine-grained bilateral access control for secure cloud-fog computing", *IEEE TDSC,* DOI: 10.1109/TDSC.2020.3001557, 2020.
[24] B. Chen, T. Xiang, M. Ma, et al., "CL-ME: Efficient Certificateless Matchmaking Encryption for Internet of Things", *IEEE IoTJ,* vol. 8, no. 19, pp. 15010 - 15023, 2021.

[25] V. Goyal, O. Pandey, A. Sahai, et al., "Attribute-based encryption for fine-grained access control of encrypted data", *CCS'06,* pp. 89-98, 2006.
[26] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption", *IEEE S & P'07*, IEEE, pp. 321-334, 2007.
[27] R. Ostrovsky, A. Sahai, B. Waters, "Attribute-based encryption with non-monotonic access structures", *CCS'07,* pp. 195-203, 2007.
[28] S. Agrawal, M. Chase M, "FAME: fast attribute-based message encryption", *CCS'17*, pp. 665-682, 2017.
[29] J. Li, Q. Yu, Y. Zhang, et al., "Key-policy attribute-based encryption against continual auxiliary input leakage", *Inf. Sci,* vol. 470, no. 175-188, 2019.
[30] J. Lai, R. H. Deng, Y. Li, et al., "Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption", *AsiaCCS'14,* pp. 239-248, 2014.
[31] J. Yu, F. Kong, X. Cheng, et al., "Forward-secure identity-based public-key encryption without random oracles", *Fundamenta Informaticae*, vol. 111, no. 2, pp. 241-256, 2011.
[32] T. Kitagawa, H. Kojima, N. Attrapadung N, et al, "Efficient and fully secure forward secure ciphertext-policy attribute-based encryption", *Information Security,* Springer, Cham, pp. 87-99, 2015
[33] J. Nieto, M. Manulis, D. Sun D, "Forward-secure hierarchical predicate encryption", *Pairing'12,* Springer, pp. 83-101, 2012.
[34] J. Wei, X. Chen, J. Wang, et al., "Forward-secure puncturable identity-based encryption for securing cloud emails", *ESORICS'19*, Springer, Cham, pp. 134-150, 2019.
[35] B. Lynn, "The stanford pairing based crypto library", accessed: Feb. 2021, [Online], Available: http://crypto.stanford.edu/pbc/.
[36] P. Li, J. Su, X. Wang, "iTLS: Lightweight Transport-Layer Security Protocol for IoT With Minimal Latency and Perfect Forward Secrecy", *IEEE IoTJ,* vol. 7, no. 8, pp. 6828-6841, 2020.
[37] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation", *ACM CCS*, pp. 1575-1590, 2020.
[38] C. Zhao, S. Zhao, M. Zhao, et al., "Secure multi-party computation: theory, practice and applications", *Information Sciences*, vol. 476, pp. 357-372, 2019.
[39] Y M. Bellare, V. T. Hoang, P. Rogaway, "Foundations of garbled circuits", ACM CCS, pp. 784-796, 2012.
[40] E. M. Songhori, S. U. Hussain, A. R. Sadeghi, et al., Tinygarble: Highly compressed and scalable sequential garbled circuits, IEEE Symposium on Security and Privacy, IEEE, pp. 411-428, 2015.
[41] A. Acar, H. Aksu, A. S. Uluagac, et al., "A survey on homomorphic encryption schemes: Theory and implementation", *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1-35, 2018.
[42] W. Lu, Z. Huang, C. Hong, et al., "PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption", *IEEE S & P*, IEEE, pp.1057-1073, 2021.

Jianfei Sun received his Ph.D. degree from University of Electronic Science and Technology of China (UESTC). He is currently a research fellow at School of Computer Science and Engineering, Nanyang Technological University. His research interests include network security and IoT security. He has published many paper on IEEE TDSC, IEEE TII, IEEE TCC, IEEE TVT, IEEE TITS, IEEE IoTJ, Inf. Sci, IEEE Systems, etc. His research interests include network security and IoT security.

Guowen Xu is currently a research fellow at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has published many paper on IEEE TDSC, IEEE TIFS, IEEE TII, IEEE TVT, ACM ACSAC, ACM AsiaCCS, etc. His research interests include AI security and privacy-preserving issues in Deep Learning.

Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelors degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

Mamoun Alazab (Senior Member, IEEE) is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems. He works closely with government, industry and some top scientists including Prof. Mauro Conti and Prof. Dusit Niyato, etc. He also served on the editorial boards of many international journals in IEEE Transactions on Computational Social Systems, Journal of Information Security and Journal of Cybersecurity and Privacy, etc.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity, Director of the Secure Mobile Centre, and Deputy Dean for Faculty & Research, School of Computing and Information Systems, Singapore Management University (SMU). His research interests are in the areas of data security and privacy, network security, and applied cryptography. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew Fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. He serves/served on the editorial boards of ACM Transactions on Privacy and Security, IEEE Security & Privacy, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, Journal of Computer Science and Technology, and Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security. He is a Fellow of IEEE and Fellow of Academy of Engineering Singapore.