PSFQ: A Blockchain-based Privacy-preserving and Verifiable Student Feedback Questionnaire Platform

Wangze Ni
HKUST
Hong Kong SAR, China
wniab@cse.ust.hk

Pengze Chen
HKUST
Hong Kong SAR, China
pchenax@cse.ust.hk

Lei Chen
HKUST, HKUST(GZ)
Hong Kong SAR, China
leichen@cse.ust.hk

ABSTRACT

Recently, more and more higher education institutions have been using student feedback questionnaires (SFQ) to evaluate teaching. However, existing SFQ systems have two shortcomings. The first is that the respondent of an SFQ is not anonymous. The second is that the statistical report of SFQs can be manipulated. To tackle these two shortcomings, we develop a novel SFQ system, namely PSFQ. In PSFQ, the respondent of an SFQ is mixed with multiple users by a ring signature. PSFQ uses an advanced ring signature approach to minimize the size of a ring signature when anonymity satisfies the requirements. Thus, the first shortcoming has been overcome. Moreover, all answers are encrypted by homomorphic encryption and stored on the blockchain, enabling users to verify the correctness of the statistical reports. Our demonstration will showcase how PSFQ provides confidential SFQ responses while ensuring the correctness of statistical reports.

PVLDB Reference Format:
Wangze Ni, Pengze Chen, and Lei Chen. PSFQ: A Blockchain-based Privacy-preserving and Verifiable Student Feedback Questionnaire Platform. PVLDB, 16(12): 3918 - 3921, 2023.
doi:10.14778/3611540.3611585

1 INTRODUCTION

Student feedback questionnaires (SFQs) are widely used in higher education institutions to evaluate teaching effectiveness [8]. Both academia [5] and industry [3] have developed various SFQ systems.

In SFQ systems, institutions/lecturers release SFQs to students and ask them to score some items, such as the courses' quality [2]. After collecting answers, systems generate statistical reports [1]. To prevent Sybil attacks, students are required to verify their identities before answering an SFQ. In Sybil attacks [6], adversaries send a large number of answers to systems to manipulate statistical reports or to crash the system by exhausting its computing resources.

However, existing SFQ systems have two shortcomings. *First*, the identity verification process against Sybil attacks reveals who the respondents are, which can deter students from providing honest feedback [10]. *Second*, the lack of transparency in the generation of statistical reports can lead to manipulation of results by adversaries or the SFQ system itself. This lack of transparency makes it difficult for students and lecturers to detect manipulation, raising doubts about the accuracy of results [12].

To overcome these two shortcomings, this paper proposes a novel SFQ system, namely PSFQ. PSFQ is built on a permissioned blockchain managed by a committee consisting of both lecturers and student representatives. Only the committee is authorized to record data on the blockchain, but anyone can access the data. When SFQs are released to students, PSFQ generates one-time *survey codes* (SC) for students. No one can infer a student by a SC. A student generates a *qualification certificate* (QC) using her SC and multiple others' SCs. A QC is a ring signature (RS) that proves that the respondent owns one of the SCs without revealing the respondent [9]. Some SCs in a QC can be eliminated by analyzing the related QCs. We use advanced approaches proposed in our prior work [9] to select SCs to minimize the size of a QC while meeting the privacy requirement. The privacy requirement of a student is the threshold of the number of un-eliminated SCs in a QC. Thus, PSFQ prevents Sybil attacks by SCs and ensures that respondents' identities remain confidential, *addressing the first shortcoming*. In addition, PSFQ stores each answer on the blockchain. The scores in answers are confused by noise and encrypted by a *Homomorphic encryption function* $H(x)$ [13] (i.e., $H(x) + H(y) = H(x + y)$). The committee releases the average score and the sum of noise in the statistical report, which can be verified by students and lecturers using the encrypted raw scores. Thus, *PSFQ solves the second shortcoming*. In summary, our work contributes in two aspects:

- The system provides people with a platform to answer SFQs confidentially and verify the correctness of statistical reports.
- The system is equipped with an advanced approach, TokenMagic, proposed in our prior work [9]. TokenMagic minimizes the size of a QC while meeting the privacy requirement.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 presents an overview of PSFQ. Section 4 demonstrates five scenarios for using PSFQ. Section 5 concludes our paper.

2 PRELIMINARIES

Permissioned blockchain. In permissioned blockchains [4], common users only have read access to the transactions on the blockchain. The writing access is restricted to a committee of authorized members who do not trust each other. In a permissioned blockchain, a user first sends a transaction to the committee. Then, the committee reaches a consensus with a consensus protocol. Next, the committee packages the transaction into a block b and appends b to the blockchain. A block b' contains a hash of transactions in b' and is linked with its previous block b by the hash of b. *Thus, users can verify whether the transactions in a block b have been tampered*

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611585

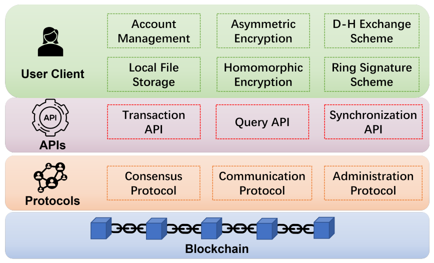


Figure 1: The System Architecture of PSFQ.

with by checking that the hash generated by the transactions in b equals the hash contained in b's next block.

Asymmetric Encryption Scheme. In an asymmetric encryption scheme, a public key (PK) pk is matched with a unique secret key (SK) sk, where $pk = sk \cdot G$, and G is a base point [11]. A PK is published, while a SK is kept secret. Data encrypted with a PK can only be decrypted with its corresponding SK.

Ring Signature Scheme. An RS scheme [9] enables an entity to prove that it belongs to a specific set of entities without revealing which one it is. For instance, if a user wants to prove that she knows the SK of pk_1, she makes an RS rs as follows: 1) She selects a set of other PKs $\overline{pk}$ and inserts pk_1 into $\overline{pk}$, denoted by $\overline{pk}'$; 2) She runs an RS generation algorithm $gen()$ to generate an RS rs over $\overline{pk}'$ using a set of random numbers RN and the SK sk_1. An RS contains the set $\overline{pk}'$ of PKs and a hash image hi_1 of sk_1. The hash image of a SK is unique, and no one can infer the SK by a hash image. With rs, verifiers can prove that rs is made by a user who knows the SK of one in $\overline{pk}'$, but they cannot tell which one it is.

Diffie-Hellman Exchange Scheme. A Diffie-Hellman (D-H) exchange scheme [7] is used to generate a PK and a SK for a user without interaction. We use an example to present a D-H key exchange scheme. Suppose a PK of Bob is pk_B, and the SK is sk_B. Suppose $\mathcal{F}_s()$ is a cryptographic hash function. Alice first randomly selects a number ω and generates $pk'_B = \mathcal{F}_s(\omega \cdot pk_B) \cdot G$. Then, Alice generates $\Omega = \omega \cdot G$ and publishes pk'_B as well as Ω. Once Bob observes Ω, he calculates $sk'_B = \mathcal{F}_s(\Omega \cdot sk_B)$. Since $sk'_B \cdot G = pk'_B$, sk'_B is the SK of pk'_B.

3 SYSTEM

3.1 System Architecture

PSFQ is built on a permissioned blockchain and involves two roles: Client and Committee Member (CM). Clients, who are typically students or lecturers, only have read access to data on the blockchain. In contrast, CMs have both read and write access, store all data on the blockchain, and are online all the time to process transactions. CMs are elected regularly. To be convincing, a committee includes officials of the institution and representatives of students/lecturers. A committee has a public-secret key pair, which is used to encrypt and decrypt students' answers. Once a new committee c is elected, c generates a new public-secret key pair and publishes the PK on PSFQ. Thus, a CM can only read data during her tenure.

Figure 1 depicts the architecture of PSFQ. There are many cryptographic tools in the *User Client* layer. A client uses cryptographic tools to encrypt her answers, generate QCs, and verify the correctness of reports. A CM uses cryptographic tools to verify the validness of QCs, decrypt answers, and generate reports. A client stores some files in her device, e.g., her SK and answers. A CM stores all transactions on the blockchain in her device. PSFQ provides *APIs* for users to propose transactions, query data, and synchronize the databases on their devices with the blockchain. Moreover, PSFQ implements some *protocols*. For example, CMs use consensus protocols to decide the update of the blockchain. Clients and CMs use communication protocols to send and receive messages. By administration protocols, students/lecturers register accounts or participate in elections of committee elections.

3.2 Workflow

Step 1. Release SFQs. Institutions or lecturers (simplified as publishers) first generate SFQs. Then, publishers make a one-time SC for each student. An SC is a PK generated by the student's PK using the D-H exchange scheme. An SC can only be used to propose an answer. Publishers publish SCs and send SFQs to students.

Step 2. Answer SFQs. A student u_i writes the answers. For each score $s_{i,j}$ over an item χ_j, u_i selects a *student noise* $sn_{i,j}$. Then, u_i identifies her survey codes (SC) sc and generates its secret key (SK) sk, namely the *survey key*, using the D-H exchange scheme. Next, u_i selects a set SC' of SCs released with the SFQ and inserts sc into SC' as SC. The number of SCs in SC represents u_i's *privacy requirement*. Then, u_i generates a qualification certificate (QC) qc_i, which is an ring signature (RS) over SC' using sk. Moreover, u_i makes a *mixing key* mk_i for the committee by the committee's public key (PK) using the D-H exchange scheme. u_i packages the answers, the student noises, and the QC as a transaction tx_i. u_i encrypts tx_i using mk_i and sends the encrypted transaction with mk_i to the committee. Meanwhile, u_i stores tx_i and mk_i in her device.

Step 3. Verify an answer. Once a CM cm receives a transaction tx_i, cm first checks if the hash image hi_i in qc_i has been recorded. Since the hash image of an SC is unique, if hi_i has been recorded, the student has proposed an answer. For fairness, a student can only propose an answer, and the transaction tx_i whose hi_i has been recorded will be rejected. Next, cm checks if qc_i is generated by the owner of one in the SC set using the RA scheme. If qc_i cannot be verified, tx_i will be rejected.

Step 4. Append an answer to the blockchain. cm uses the D-H exchange scheme to calculate a *decode key* dk_i, which is the SK of the mixing key used in tx_i. cm decrypts tx_i using dk_i. For each score $s_{i,j}$ in tx_i, cm selects two random numbers, namely the *committee noise* $cn_{i,j}$ and the *double noise* $dn_{i,j}$. Then, cm sends tx_i, committee noises, and double noises to other committee members. After reaching a consensus with other committee members, cm appends the encrypted tx_i to the blockchain, attached with $H(cn_{i,j})$, $H(dn_{i,j})$, $H(s_{i,j} + cn_{i,j} + sn_{i,j})$, and $H(dn_{i,j} + sn_{i,j})$.

Step 5. Release a statistical report. After the expiration of an SFQ, cm generates a statistical report. For each item χ_j, cm publishes the number N_j of students scoring χ_j and the average scores $v_j = \frac{\sum_i s_{i,j}}{N_j}$. Meanwhile, for each χ_j, cm publishes the sum of students noises $SN_j = \sum_i sn_{i,j}$, the sum of committee noises $CN_j = \sum_i cn_{i,j}$, and the sum of double noises $DN_j = \sum_i dn_{i,j}$.

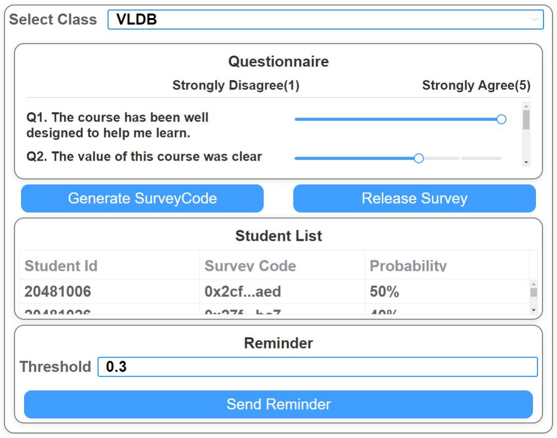


Figure 2: The Release Interface

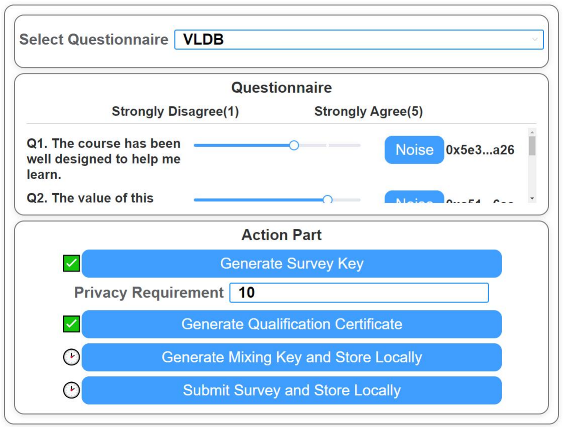


Figure 3: The Answer Interface.

Step 6. Verify a statistical report. A user u_i first checks if her transaction tx_i was appended to the blockchain without tampering. Then, for each χ_j, u_i checks whether $H(v_j \cdot N_j) + H(SN_j) + \sum_i H(cn_{i,j})$ equals $\sum_i H(s_{i,j} + cn_{i,j} + sn_{i,j})$ and whether $H(SN_j) + \sum_i H(dn_{i,j})$ equals $\sum_i H(dn_{i,j} + sn_{i,j})$. If two conditions are satisfied, the statistical report is correct.

3.3 Core Technique

The resistance to Sybil Attacks. Since each student is assigned a SC that can only be used once, adversaries cannot propose a large number of answers to PSFQ. Thus, PSFQ prevents Sybil attacks.

The generation of QCs. QCs mix the respondents of answers. A QC's anonymity is determined by its SC set. For example, there are 4 SC sets, $SC_1 = \{sc_1, sc_2\}$, $SC_2 = \{sc_1, sc_2\}$, $SC_3 = \{sc_1, sc_2, sc_3\}$, and $SC_4 = \{sc_1, sc_4, sc_5\}$. Each SC can only be used once, and an SC set includes an SC used for the answer. Thus, we can deduce that sc_1 and sc_2 are used in SC_1 and SC_2. Thus, sc_1 and sc_2 can be eliminated from SC_3, and the anonymity of SC_3 is 0. Since sc_1 cannot be used in SC_4, SC_1 and SC_4 have the same anonymity (i.e., the probability that adversaries successfully find the SC that was used by a random guess), but SC_4 includes more SCs than SC_1. The size of a QC is related to the number of SCs. Thus, the QC using SC_4 causes higher communication and storage costs. PSFQ employs an RA scheme, TokenMagic, proposed in our prior work [9]. TokenMagic combines several existing SC sets to generate a new SC set such that the SCs in the new SC set will not be eliminated. Since different SC sets contains different numbers of SCs, combining different SC sets leads to a different size of a new SC set. TokenMagic intelligently selects existing SC sets to minimize the size of a QC while meeting the anonymity requirement set by the client.

The privacy protection. The respondent of an answer is confused by a QC. Each score $s_{i,j}$ in an answer is confused by a student noise $sn_{i,j}$. Even if adversaries know the function $H()$ and the value $H(s_{i,j} + sn_{i,j})$, they cannot infer $s_{i,j}$. Moreover, an answer is encrypted by a mixing key, whose SK is only known to the committee. Thus, adversaries cannot decrypt the answer.

The correctness of an answer on the blockchain. A user verifies if her answer is correctly appended to the blockchain by checking if the answer on the blockchain equals the answer stored in her device. Besides, she checks if $H(s_{i,j} + cn_{i,j} + sn_{i,j})$ equals $H(cn_{i,j}) + H(s_{i,j} + sn_{i,j})$. She also checks if $H(dn_{i,j}) + H(sn_{i,j})$ equals $H(dn_{i,j} + sn_{i,j})$. If all conditions are satisfied, the answer is correctly appended.

The correctness of a report on the blockchain. If no student declares that her answer was incorrectly recorded on the blockchain, $H(cn_{i,j})$, $H(dn_{i,j})$, $H(s_{i,j}+cn_{i,j}+sn_{i,j})$, and $H(dn_{i,j}+sn_{i,j})$ are correct. Thus, if $\sum_i H(dn_{i,j})+H(SN_j)$ equals $H(dn_{i,j}+sn_{i,j})$, SN_j is correct. Then, if $H(v_j \cdot N_j) + H(SN_j) + \sum_i H(cn_{i,j})$ equals $\sum_i H(s_{i,j} + cn_{i,j} + sn_{i,j})$, v_j is correct.

3.4 Differences from the Existing Systems

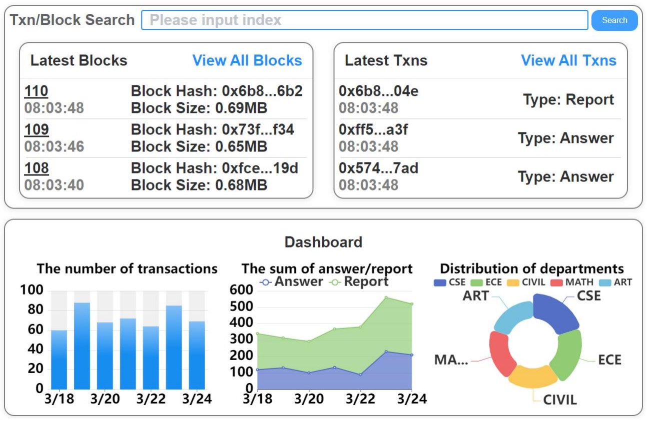
In existing systems [3, 5], the respondent of an answer is clear. However, in PSFQ, the respondent of an answer is confused by a QC. Thus, students dare to give honest feedback [10] in PSFQ. In some systems [3], only the system knows answers, preventing students/lecturers from verifying the correctness of reports. Although some systems [5] enable verification, they reveal answers to the public. In contrast, PSFQ employs homomorphic encryption and noise to obscure answers before publishing, allowing for verification without compromising confidentiality. As a result, students/lecturers can verify report while maintaining privacy.

4 DEMONSTRATION

In our demonstration, attendees, acting as lectures and students, interactively experience using PSFQ to release SFQs, answer SFQs and verify statistical reports. Suppose an attendee Alice is a lecturer, and Bob is a student attendee enrolled in Alice's class.

Scenario 1. Design and release SFQs. Alice designs her SFQ in the *Release* interface, shown in Figure 2. She first selects the class for which the SFQ is intended. She then adds questions in the *Questionnaire* panel. Once the SFQ is ready, she generates one-time SCs for students, which are displayed in the *Student List* panel. Finally, Alice releases the SFQ.

Scenario 2. Answer an SFQ. Once Alice releases her SFQ, Bob can answer the SFQ in the *Answer* interface, shown in Figure 3. Bob first selects Alice's SFQ in the selection bar. Then, he answers the SFQ. For each score over an item, Bob clicks the *Noise* button to generate a student noise, displayed to the right of the button. Bob then clicks the *Generate a Survey Key* button, which displays a waiting symbol while the survey key is being generated. Once the mixing key has been generated, a green tick replaces the wait


Txn/Block Search

Latest Blocks	
110 08:03:48	Block Hash: 0x6b8...6b2 Block Size: 0.69MB
109 08:03:46	Block Hash: 0x73f...f34 Block Size: 0.65MB
108 08:03:40	Block Hash: 0xfce...19d Block Size: 0.68MB

Latest Txns	
0x6b8...04e 08:03:48	Type: Report
0xff5...a3f 08:03:48	Type: Answer
0x574...7ad 08:03:48	Type: Answer

Figure 4: The Query Interface

symbol. Then, Bob inputs a privacy requirement and generates the QC. Next, Bob clicks the buttons to generate a mixing key, encrypt the answer, and submit the answer.

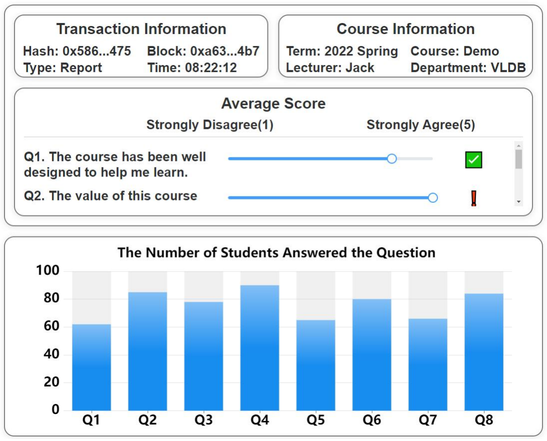
Scenario 3. Reminder Students to Answer. Although the respondent of an answer is mixed by a QC, PSFQ still can calculate the *answer probability* (AP) that a student has answered a SFQ. For example, there are two QCs whose SC sets are $SC_1 = \{sc_1, sc_2\}$ and $SC_2 = \{sc_2, sc_3\}$. Suppose sc_1 is Jack's SC, sc_2 is Bob's SC, and sc_3 is Tom's SC. Then, the AP of Jack is $\frac{1}{3}$, the AP of Bob is $\frac{2}{3}$, and the AP of Tom is $\frac{1}{3}$. *Alice appreciates the privacy-preserving effect of PSFQ by observing APs.* Then, Alice sets a threshold $\overline{p}$ of APs and sends reminders to students whose APs are lower than $\overline{p}$.

Scenario 4. Query data. Alice and Bob can query data on the blockchain in the *Query* interface, shown in Figure 4. Alice and Bob first appreciate the throughput of PSFQ by the dashboard at the bottom of the page. The dashboard shows the number of transactions per day, the sum of answers/reports on the blockchain, and the distribution of the departments of the SFQs. Since the transaction including Bob's answer is not listed in the recent transaction list, Bob clicks the *View All Blocks/Transactions* button to view all blocks/transactions. Alice wants to see the transactions in the latest block. Thus, she clicks the index of the latest block to see the details.

Scenario 5. Verify a report. After a while, Alice's SFQ expires and a statistical report is generated. Alice and Bob verifies the report in the *Verify* interface, shown in Figure 5. They first enter the hash of the transaction containing the report in the search box in the Query interface. After clicking the transaction, the Verify interface for the report is invoked. Then, the transaction information (e.g., hash, time) is displayed in the upper left corner of the page. The course information (e.g., term, lecturer) is displayed in the upper right corner of the page. The details of the report are displayed below the information panel. Alice and Bob verify the correctness of each average score by clicking the *Check* buttons. If the committee manipulates an average score, a red exclamation symbol appears. If the average score is correct, a green tick symbol appears.

5 CONCLUSION

Our demo introduces a new SFQ system called PSFQ. PSFQ enables users to answer SFQs confidentially and verify the correctness of statistical reports. To achieve anonymity while minimizing costs, we have employed a novel technique called TokenMagic [9]. Our demo allows the audience to interactively explore how PSFQ protects students' privacy and guarantees the correctness of reports.


Transaction Information
Hash: 0x586...475 Block: 0xa63...4b7
Type: Report Time: 08:22:12

Course Information
Term: 2022 Spring Course: Demo
Lecturer: Jack Department: VLDB

Average Score
Strongly Disagree(1) Strongly Agree(5)
Q1. The course has been well designed to help me learn.
Q2. The value of this course

Figure 5: The Verify Interface

6 ACKNOWLEDGMENT

Lei Chen's work is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant and HKUST-Webank joint research lab grants. Corresponding author: Lei Chen.

REFERENCES

[1] [n.d.]. [Online] Statistics Reports. https://oidr.hkust.edu.hk/4e.htm, Last accessed on 2023-07-02.
[2] [n.d.]. [Online] Student Feedback Questionnaires. https://student-survey.hkust.edu.hk/web/4a1.htm, Last accessed on 2023-07-02.
[3] [n.d.]. [Online] SuFo. https://play.google.com/store/apps/details?id=my.edu.uitm.sufo&hl=zh_HK&gl=US&pli=1, Last accessed on 2023-07-02.
[4] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. 2019. Caper: a cross-application permissioned blockchain. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1385–1398.
[5] Atharv Chandratre and Shubham Garg. 2019. Blockchain Based Course Feedback System. *Available at SSRN 3762332* (2019).
[6] John R Douceur. 2002. The sybil attack. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1.* Springer, 251–260.
[7] Nan Li. 2010. Research on Diffie-Hellman key exchange protocol. In *2010 2nd International Conference on Computer Engineering and Technology*, Vol. 4. IEEE, V4–634.
[8] Luke Mandouit. 2018. Using student feedback to improve teaching. *Educational action research* 26, 5 (2018), 755–769.
[9] Wangze Ni, Peng Cheng, Lei Chen, and Xuemin Lin. 2021. When the recursive diversity anonymity meets the ring signature. In *Proceedings of the 2021 International Conference on Management of Data*. 1359–1371.
[10] Jennifer Rowley. 2003. Designing student feedback questionnaires. *Quality assurance in education* 11, 3 (2003), 142–149.
[11] Moumita Roy, Nabamita Deb, and Amar Jyoti Kumar. 2014. Point generation and base point selection in ECC: An overview. *International Journal of Advanced Research in Computer and Communication Engineering* 3, 5 (2014), 6711–6713.
[12] Don F Westerheijden, Veerle Hulpiau, and Kim Waeytens. 2007. From design and implementation to impact of quality assurance: an overview of some studies into what impacts improvement. *Tertiary Education and Management* 13 (2007), 295–312.
[13] Xun Yi, Russell Paulet, Elisa Bertino, Xun Yi, Russell Paulet, and Elisa Bertino. 2014. *Homomorphic encryption.* Springer.