Solution Probing Attack against Coin Mixing based Privacy-Preserving Crowdsourcing Platforms

Yunlong Mao, *Member, IEEE,* Ziqin Dang, *Student Member, IEEE,* Heng Wang, Yuan Zhang *Member, IEEE,* and Sheng Zhong, *Senior Member, IEEE*

***Abstract*—Conventional crowdsourcing platforms primarily rely on a central server as the broker for information exchange. Although many efforts have been made, centralized platforms are still vulnerable to underlying security issues, such as an untrusted central server and single-point failure. Fortunately, blockchain has emerged as an alternative infrastructure for building crowdsourcing platforms. Many excellent designs of blockchain-based decentralized crowdsourcing (BDCS) solutions have been proposed. Benefiting from blockchain, BDCS can provide fascinating features, like tampering resistance and anonymity. However, a new attack surface appears in BDCS. Recently, a new attack against BDCS named solution probing attack has been identified. The solution-probing adversary can take advantage of the anonymity of BDCS to probe valid solutions using a generative model. Due to the transparency of blockchain transactions, the probing attack is effective even if solutions are encrypted. Nevertheless, we find transaction-mixing techniques effective in defending against probing attacks. In this paper, we introduce the solution probing attack and an improved variant, which can attack coin mixing-based BDCS. We evaluate probing attacks on large-scale crowdsourcing tasks. Experimental results show that the adversary is capable of deceiving BDCS with a limited number of probing, even if the BDCS is protected by solution encryption and coin mixing techniques.**

***Index Terms*—crowdsourcing security, probing attacks, coin mixing, decentralized platform**

I. Introduction

Crowdsourcing has contributed tremendously to traditional industries. Crowdsourcing applications like question-answering [2], fraud detection [3], and ride-sharing [4] have changed the way people live. Data collection is a pretty popular application of crowdsourcing, which labels large amounts of data for industrial or academic purposes, such as machine learning and big data analysis. However, the solution coming from a single worker is commonly unreliable due to individual bias or lack of expertise. Therefore, a common practice for data requesters is to collect aggregated solutions from multiple workers to tackle the problem. Then, a high-quality solution will be extracted from those aggregated solutions with truth discovery algorithms [5]–[8].

As for the crowdsourcing platform, it is conventionally designed in a centralized fashion. A broker with global information can allocate crowdsourcing jobs to proper workers efficiently [9]. Given a specific set of constraints, such as computing resources and job requirements, a centralized crowdsourcing platform can provide the optimal (or approximately optimal) task allocations. However, a centralized crowdsourcing platform requires a trusted third party to act as the broker, which in practice, is hardly available and vulnerable to single-point failure, DDoS attacks, and Sybil attacks [10]. Consequently, redesigning crowdsourcing platforms in a decentralized manner becomes a natural demand. Fortunately, the emergence of blockchain offers a promising infrastructure for building blockchain-based decentralized crowdsourcing (BDCS). Recent work has proposed excellent BDCS designs [11]–[15] atop well-developed blockchains, such as Ethereum.

BDCS, on the one hand, solves the problems of centralized platforms. But on the other hand, it brings new security challenges. Since blockchain uses distributed peers as a public ledger, task solutions submitted by workers will be publicly accessible, resulting in confidential data leakage. Several studies, such as [15], have recommended encrypting solutions before submission. However, encrypted solutions are unsuitable for real-time truth estimation and fair rewarding proof. To tackle the problem, some studies have combined solution encryption with zero-knowledge proof [12]. Unfortunately, security issues still exist in BDCS. Although solutions can be recorded in ciphertext, rewarding transactions between the requester and workers cannot be concealed in the ledger. A solution probing attacker [1] can disclose the hidden relationship between submitted solutions and rewards by observing rewarding transactions. After collecting sufficient rewarding transactions, the attacker can estimate the job's requested truth using a generative model.

The previous study [1] proposes a primary solution probing attack against the existing BDCS designs. In particular, an adversarial worker can take advantage of the anonymity of blockchain to repeatedly submit generated solutions with the goal of defrauding the requester. We note that solutions are not generated randomly. The amount of a reward is highly relevant to the quality of a solution. Therefore, the adversary can collect rewarding transactions from the public ledger and take them as training data to build a generative model, capturing the hidden relation between rewards and solutions. Due to the public ledger transparency, any legal worker registered in BDCS will be capable of building the model. Thus, the probing attack is a pervasive issue in BDCS designs. By utilizing anonymity and transparency simultaneously, an adversarial worker, which may even be not qualified for a crowdsourcing job, can forge

Yunlong Mao, Ziqin Dang, Yuan Zhang, and Sheng Zhong (corresponding author) are with the State Key Laboratory for Novel Software Technology, Nanjing University, China, 210023.

Part of this work was done by Heng Wang when he was a graduate student at Nanjing University. He is now a software engineer at Pony.ai.

A preliminary version [1] has been published in the proceedings of the 19th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS) 2022.

valid solutions for rewards without doing the actual work. For example, when a requester wants to collect real-time weather data in Ushuaia, a solution probing attacker can forge valid solutions even if located in New York City.

However, as indicated by [1], the solution probing attack can be mitigated by obfuscating the rewards. To this end, a mix-and-match defensive solution is proposed in [1]. In this paper, we take this idea a step further, introducing coin mixing [16]–[18] into BDCS for defense against solution probing attacks. We first investigate how coin mixing based BDCS frustrates solution probing attacks. Then, we propose an improved probing attack aiming at defeating coin mixing based BDCS. Actually, coin mixing techniques are not perfectly implemented in practice. Recent studies [19] have revealed that the relationship between anonymous users can still be inferred by learning transaction graph knowledge. Even though both workers and requesters use pseudo-anonymity and coin mixing for trading, the linkage between workers and requesters will be permanently recorded in the ledger. Moreover, by participating in BDCS jobs to be mixed with a target job, the attacker can mount a differential attack and find out possible combinations of solution-reward pairs with the help of transaction linkage analysis.

We evaluate the performance of solution probing attacks against the original BDCS design and a coin mixing based BDCS design. Three classical truth discovery algorithms are used for evaluating solution quality, i.e., Gaussian truth model (GTM) [7], conflict resolution on heterogeneous data (CRH) [6], and PACE [5], which are widely used in crowdsourcing studies for synthetic and real-world data. To evaluate attack performance in different crowdsourcing tasks, we use four different datasets. One is synthetic, using a normal distribution, while the other three are real-world data commonly used in crowdsourcing studies for evaluation. Experimental results have proved that the improved solution probing attack is highly effective against coin mixing based BDCS. Overall, the main contributions made in the paper are three-fold:

- We report a new type of attack against BDCS designs named solution probing attack, defrauding requesters for rewards without finishing the job honestly.
- We introduce a coin mixing based BDCS design and propose an improved solution probing attack defeating it by combining a generative model with transaction analysis methods.
- We implement two solution probing attacks and evaluate their performance against the original BDCS design and a coin mixing based BDCS design. The evaluation results confirm the effectiveness of attacks in different crowdsourcing tasks using different truth discovery strategies.

II. Related Work

A. Blockchain-based Decentralized Crowdsourcing

The emergence of blockchain infrastructure offers a promising solution for BDCS designs. Generally, BDCS works with permissionless blockchain systems, such as Bitcoin [20] and Ethereum [21]. Some recent studies have proposed excellent designs [11], [12], [14], [15] atop well-developed blockchain. In two typical BDCS designs, CrowdBC [15] and ZebraLancer [12], coordination of workers and requesters, such as task match, worker selection, and rewarding transactions is accomplished by self-executing smart contracts. [22] further introduces requester clustering and selection in a mobile environment. [14], smart contracts, along with ciphertext policy attribute-based encryption, are leveraged to build BDCS with fine-grained authorization for data trading. NF-Crowd [11] proposes a protocol reducing the lower bound of transaction fees of the underlying blockchain to $\mathcal{O}(1)$ regards to the number of job participants.

B. Crowdsourcing Quality Evaluation

A quality-aware crowdsourcing platform prefers to reward workers according to solution quality. This strategy is beneficial for the platform in the way that it increases its reliability; it is also beneficial for requesters, who obtain high-quality results from the task; and it is also beneficial to workers, who obtain more rewards by providing high-quality data [5], [8]. Therefore, it is necessary for the platform to utilize truth discovery algorithms to estimate the truth from the aggregated data and evaluate their corresponding quality. In [5], [6], the truth estimation problem is modeled as an optimization problem, in which the truth is the value that minimizes the distance from all data. The quality of each data is then measured by its distance from the estimated truth because the higher the quality, the closer it is to the truth. Besides, the Bayesian probabilistic model is also leveraged for estimating the truth and qualities of workers, in which the expectation-maximization algorithm is utilized to update the estimated truth and qualities [7]. We note that the attack proposed in this paper is effective both for the optimization and the probabilistic model of truth and quality estimation algorithms.

C. Attack against BDCS

In BDCS, attacks can be mounted by a requester, such as *false-reporting attack*, *clogging attack*, or by workers, such as *free-riding attack*, *data poisoning attack* and *transaction analysis attack* [12], [15], [27]–[33]. In the false-reporting attack, requesters misreport the quality of the solutions in order to reduce the reward owed, or claim they have not received the solutions [15], [28]. In BDCS, smart contracts can be utilized to automatically reward workers from deposits provided by the requester according to pre-defined reward policies [12], [15]. In the clogging attack, requesters publish fake tasks to drain the resources of the workers, especially in mobile crowdsourcing [27]. In the free-riding attack, workers obtain rewards by exerting little or no effort in the task, e.g., by submitting random noise. In the data poisoning attack, the adversary intentionally forges data to deviate from the estimated truth [29], [30]. Transaction analysis attack intends to deanonymize users' identities and information by analyzing the public transactions in BDCS. Typically, [31] constructs the transaction graph to link Bitcoin public keys with real identities. [32], [33] further focus on network information analysis like IP addresses and cookies. We note that the

TABLE I
COMPARSION OF DEFENSE SOLUTIONS FOR BDCS.

Scheme	Resistance to Attacks						Features		
	Sybil Attack	False Reporting	Free riding	Plain-SPA	Cipher-SPA	DDoS Attack	Fully on-chain	Minium Mixing Time	Privacy Preserving
CrowdBC [15]	✓	✓	✓	✗	✗	-	-	-	✗
BPCM [22]	✓	✓	✗	✗	✗	-	-	-	✓
ZebraLance [12]	✓	✓	✓	✗	✗	-	-	-	✓
CoinJoin [23]	✗	✓	✗	✓	✗	✓	✓	1 block	✗
CoinShuffle [16], [24]	✗	✓	✗	✓	✗	✓	✓	1 block	✓
CoinParty [25]	✓	✓	✓	✓	✗	partial	✗	2 blocks	✓
TumbleBit [26]	✓	✓	✓	✓	✗	✓	✗	2 blocks	✓
CrowdMix	✓	✓	✓	✓	✓	✓	✓	2 blocks	✓

SPA means the solution probing attack.
CrowdMix is the defense solution constructed in the paper by adopting a coin mixing technique.

proposed solution probing attack is different from the free-riding attack since the solution probing attack aims at quality-aware crowdsourcing tasks, which are tricky for free-riding attackers. On the other side, we note that the proposed probing attack tries to affect the estimated truth as little as possible, which is totally different from the data poisoning attacks.

D. Defense solutions for BDCS

Accordingly, various defense solutions for BDCS have been studied. In [28], reputation mechanisms are employed for managing workers and requesters to defend against the false-reporting attack. To tackle the free-riding attack, workers are selected and rewarded according to their reputations or quality [28], or they are required to deposit to smart contracts before participating in a task in BDCS, then retrieve the deposition if they submit high-quality solutions [12], [15]. A common-prefix-linkable scheme is proposed in [12] to detect malicious workers who repeatedly submit solutions to the same task, but it requires a trusted third party for authentication. To defend against data poisoning attack, median-of-weighted-average, and maximum influence estimation are leveraged to mitigate the influence of the forged data [29], [30]. Aiming at defending against transaction analysis attacks, a recent study proposes that the requester can delay payment and divide account addresses in transactions [19].

In particular, coin mixing is a classic idea for enhancing the anonymity of blockchain transactions, aiming to eliminate the link information between payers, payees, and the transaction itself. In coin mixing schemes, a group of payers exchange their coins and re-generate transactions, effectively hiding the relations between cash and owners. CoinJoin [23] is one of the first proposed schemes for mixing coins. Due to the characteristic of blockchain that does not constrain the number of inputs and outputs in each transaction, CoinJoin mixes one-to-one transactions from various payers and generates a joint transaction in a random permutation to hide the relation. This mixing approach eliminates external links but does not consider internal credibility between payers at all. CoinShuffle [16], [24] designs P2P mixing protocol by sequential encryption and DiceMix to ensure security. CoinParty [25] and TumbleBit [26] set up one or more agents for coin mixing while collecting payers' coins by off-chain transactions with more security assumptions.

TABLE I makes a detailed comparison of related defense solutions for protecting BDCS, listing the resistance to common attacks with other solution features. Compared with the typical BDCS designs, coin mixing based BDCS can defend against the plaintext solution probing attack. Furthermore, CoinJoin and Coinshuffle have a lower overhead in coin mixing but can not resist the Sybil attack and free riding. CoinParty and TumbleBit have better resistance to all the attacks while lacking transparency due to off-chain payment. In particular, none of them can resist ciphertext solution probing attacks. Please note that the original CoinJoin and CoinShuffle methods have not taken into account encrypted task solutions. Therefore, we have adapted these methods into BDCS with encryption. The experimental result shows that these methods cannot defend against the cipher solution probing attack. However, if we construct a defense solution by combining the coin mixing and encryption techniques subtly, like CrowdMix introduced in this work, the solution probing attack can be defeated. That is the motivation for our extension work, improving the solution probing attack against coin mixing based BDCS.

III. SYSTEM AND THREAT MODEL

A. Blockchain-based Decentralized Crowdsourcing (BDCS)

A BDCS system provides crowdsourcing users with a peer-to-peer transaction platform. Since the decentralized crowdsourcing system studied in the paper is built upon blockchain, we will introduce the system model in a composite form. As shown in Figure 1, a benign BDCS system consists of blockchain peers (also known as miners), crowdsourcing requesters, and workers. Distributed peers construct the infrastructure of blockchain via a peer-to-peer network, upon which the bargain of crowd knowledge is facilitated just as in a centralized crowdsourcing platform. The requester and

worker using a BDCS system can acquire attractive features provided by the blockchain, such as anonymity, transparency, and temper proof. The routine of each role is listed as follows.

- *Requester*. A requester is commonly an individual or an organization who hires the crowd to finish a predefined task. The task usually requires some specific conditions or human intelligence to be accomplished, such as environment sensing or data entry annotation. To motivate the crowd to join the task, the requester should provide a proper incentive. Each requester in BDCS can publish multiple tasks.
- *Worker*. A worker can be any individual with the necessary requirements. If a worker is interested in a published task, the worker can try to earn the reward by finishing the predefined task according to the specified requirements in the job description. By submitting a valid solution to the requester, the worker becomes a legal candidate for reward. Each work in BDCS can undertake multiple tasks.
- *Miner*. A miner (or peer, interchangeable) is responsible for maintaining the underlying blockchain by generating blocks to store transactions and validating them in the network. The miner will receive monetary or service rewards by maintaining the blockchain infrastructure.
- *Blockchain*. The blockchain used in a BDCS system is commonly a permissionless blockchain, which consists of peers mining for blocks in an open registering manner. We note that it is not necessary for any requester or worker to participate in the blockchain as a peer. Instead, a BDCS can be constructed using blockchain as a service (BaaS) or off-the-shelf public blockchain, such as Bitcoin [20] and Ethereum [21]. All messages transmitted between requesters and workers are recorded in blockchain transactions.

B. Truth Estimation for Crowdsourcing

Prior knowledge of crowdsourcing tasks may be unavailable for the requester. Meanwhile, crowdsourced workers may submit biased solutions. Therefore, a truth estimation algorithm is commonly used in crowdsourcing for the discovery of true solutions [34]. Three classic truth discovery algorithms are considered here, i.e., GTM, CRH, and PACE [5]–[7]. The GTM algorithm is based on the probabilistic model with the goal of maximizing Equation (1), in which $\mathcal{M}$ is the estimated truth value of each data, Σ is the variance of each data, $\mathcal{S}$ denotes the participating workers, $\mathcal{C}_e$ is the dataset of e-th dimension, o_c is the normalized data, μ_0 and σ_0 is the prior knowledge of the task, α and β are hyperparameters of the model. The variance of each data is negatively correlated with data quality. While in the CRH algorithm, the goal is to minimize Equation (2), in which w_k is the quality of k-th data, v_{im}^k is the k-th data for i-th task of m-th dimension, v_{im}^* is the estimated truth value of the data, $d(\cdot,\cdot)$ denotes the distance function. The constraint of Equation (2) is to ensure the practicality of the solutions. In the PACE algorithm, the truth is estimated as the centroid χ of all the data, then the deviation of each data from the centroid is computed as $\theta_i = |d(\chi, d_i)|$, in which d_i denotes the i-th data. Finally, the quality is calculated as in Equation (3).

$$\max_{\mathcal{M},\Sigma} f(\mathcal{M},\Sigma) = -\sum_{s\in\mathcal{S}}\left(2(\alpha+1)\log\sigma_s + \frac{\beta}{\sigma_s^2}\right) - \sum_{e\in\mathcal{E}}\frac{(\mu_e-\mu_0)^2}{2\sigma_0^2} - \sum_{e\in\mathcal{E}}\sum_{c\in\mathcal{C}_e}\left(\log\sigma_{s_c} + \frac{(o_c-\mu_e)^2}{2\sigma_{s_c}^2}\right). \tag{1}$$

$$\min_{\mathcal{X}^*,\mathcal{W}} f\left(\mathcal{X}^*,\mathcal{W}\right) = \sum_{k=1}^{K} w_k \sum_{i=1}^{N}\sum_{m=1}^{M} d_m\left(v_{im}^*, v_{im}^k\right), \quad s.t. \quad \sum_{k=1}^{K} exp\left(-w_k\right) = 1. \tag{2}$$

$$q_i = \frac{\frac{1}{\theta_i+\epsilon}}{\sum_i \frac{1}{\theta_i+\epsilon}}. \tag{3}$$

C. Threat Model

Since a BDCS system uses blockchain as a public and transparent ledger, historical transactions of crowdsourcing tasks are publicly accessible, and any legal BDCS user can view solutions submitted regarding the task. In order to tackle the access control issue, workers are demanded to encrypt their solutions before submission by defensive solutions [12], [15]. However, a recent study indicates that private information leakage still exists, and a solution probing attack [1] can be mounted under the encryption circumstance. The underlying reason is that although solutions are in ciphertext, the BDCS system records rewarding transactions in plaintext, which can be exploited by the adversary to infer the distribution of correct solutions. It has also been verified that privacy-preserving transaction techniques like coin mixing can mitigate the adversarial effect of solution probing attacks, which can be a promising defense solution against the probing attack. As demonstrated in Figure 1, we have introduced a feasible solution probing attack against BDCS with solution encryption schemes. However, this attack can be defeated by combining coin mixing and encryption techniques. In this paper, we will investigate an effective solution probing attack against BDCS that is constructed using coin mixing and solution encryption techniques.

In the BDCS system, a requester is regarded as honest to workers but curious to other requesters. When having received solutions, a requester will just estimate the quality and pay for them without any information interception. However, a requester may store other requesters' addresses or real identities during the joint task publishing or transaction mixing. Each worker in BDCS, also regarded as semi-honest, follows the crowdsourcing protocol normally but tries to deceive requesters for rewards without actual work. This can be achieved by probing legal solutions using fake identities or additional accounts. Malicious behaviors like poisoning are out of the discussion.

*1) **Adversarial Goal**:* The goal of the adversary is to maximize the rewards obtained from the BDCS by submitting forged solutions. More specifically, the adversarial goal is to deceive the requester that the forged solutions are as good as solutions submitted by benign workers with actual labor.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3355453

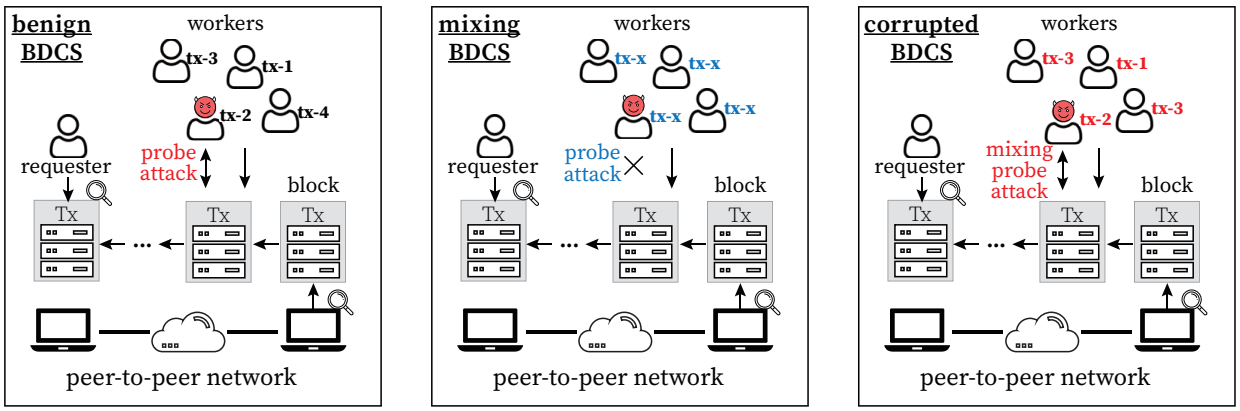


Fig. 1. Blockchain-based decentralized crowdsourcing system and two different probing attacks.

As per the task specification, the adversary should not be a qualified worker. For example, the adversary is not located in a specified area for a sensing task. For the adversary, the cost of computing resources is much lower than becoming qualified. Since the probing attack can be mounted repeatedly in any task, the adversary's cost can be neglected when compared with the profit earned by forging solutions. Moreover, although it is not the adversary's original intention, the fact that the adversary obtains high rewards without actual efforts undermines the BDCS fairness, thereby reducing the reliability and reputation of the BDCS system.

*2) **Adversarial Capability**:* To achieve the adversarial goal, essential capabilities are needed. Considering practical applications, capability limitations should also be made explicitly. First of all, the adversary is semi-honest, having the same capability as a benign worker, e.g., accessing public information on the blockchain, submitting solutions to the requester, and being involved in transactions. Besides, the adversary can use pseudo-anonymous identities on the BDCS to submit multiple solutions for the same crowdsourcing task. Since rewarding transactions are recorded on a public ledger, the adversary can guess the solution by observing historically rewarding transactions, which is the basic idea of the previous probing attack [1]. Considering possible defenses, accessible information on the BDCS varies in different threat models.

- *A0*. This is the basic threat model without any defensive solutions. Solutions and rewards are recorded in the BDCS in plaintext;
- *A1*. On the basis of *A0*, solutions are encrypted by workers. Rewarding transactions are recorded in the BDCS in plaintext;
- *A2*. On the basis of *A1*, rewarding transactions are protected using a coin mixing technique.

Please note that we limit the adversarial capability to probing. Malicious behaviors like data poisoning attack [30] is out of the discussion. It is another research topic of crowdsourcing security and should be studied separately.

IV. Solution Probing Attack

In this section, we introduce the solution probing attack against a BDCS system under *A0* and *A1* assumptions, deceiving the requester through generating solutions with plaintext probing and ciphertext probing, respectively.

A. Attack with Plaintext Solutions

Benefiting from the anonymity of the underlying blockchain, the BDCS system offers anonymous submissions for workers. Although the anonymity of BDCS can apparently preserve the identity of a worker or a requester, the linkage between solutions and rewards still exists due to the transparency of the blockchain public ledger. Therefore, an adversary $\mathcal{A}$ can generate the required solutions of a task based on the linkage between the probed solutions and rewards to deceive the requester. We first give a concise interpretation of the probing attack by demonstrating how to forge solutions under *A0* threat model since it is straightforward.

Informally, for a crowdsourcing task T, the solution probing attack contains two main procedures, i.e., *probing* and *submitting*. In the probing procedure, the adversary $\mathcal{A}$ can probe at most N solutions and obtain the corresponding rewards, where N is the minimum number of legal solutions required by T. By investigating the hidden relationship between solutions and the corresponding rewards from the observed solution-reward pairs, $\mathcal{A}$ can train a generative model $\mathcal{G}$ capturing the hidden relationship. Then, in the submitting procedure, $\mathcal{A}$ can generate new solutions with $\mathcal{G}$ and submit solutions using new identifiers in BDCS. It should be noted that under *A0* threat model, $\mathcal{A}$ can collect solution-reward pairs corresponding to other workers in plaintext from the public ledger directly instead of literally probing. For brevity, we say $\mathcal{A}$ probes at most N solution-reward pairs (including collection from the public ledger) regarding a specific task.

We also note that it is possible to train the generative model with a small proportion of solution-reward pairs when N is quite large, or the hidden relationship is simple. After collecting enough solution-reward pairs, $\mathcal{A}$ constructs a training dataset $\mathbb{D}$. Then, $\mathcal{A}$ trains $\mathcal{G}$ to generate M solutions based on the hidden relationship between solutions and rewards. For the task T, if there exists a solution generated by $\mathcal{A}$ within the maximum requested solutions is evaluated as valid by the requester, we say that the probing attack is successful.

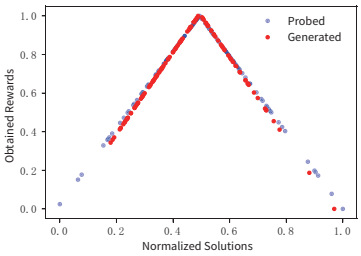


Fig. 2. The probed and generated solutions and their corresponding rewards.

Therefore, the goal of $\mathcal{A}$ is to maximize the rewards from the BDCS system, which can be formulated as

$$\underset{\mathbb{D}_g \leftarrow \mathcal{G}}{\text{maximize}} \ \frac{1}{M} \sum_{d \in \mathbb{D}_g} \mathcal{R}_T(d), \tag{4}$$

where $\mathbb{D}_g$ denotes solutions generated by $\mathcal{G}$, $\mathcal{R}_T$ is the reward policy of the crowdsourcing task T. Particularly, we assume that the $\mathcal{A}$ utilizes a typical GAN [35] as the generative model $\mathcal{G}$, which involves two deep neural networks, i.e., a generative model and a discriminative model competing against each other. Please note that our attack can be implemented using other generative models. Briefly, the generative model $\mathcal{G}$ learns to generate solution-reward pairs, while the discriminative model $\mathcal{D}$ learns to differentiate generated solutions from the probing ones:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\boldsymbol{x} \sim p_{\mathbb{D}}(\boldsymbol{x})} \log \mathcal{D}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}(\log(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{z})))), \tag{5}$$

where $\boldsymbol{z}$ is a noise variable following a prior distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$, such as a normal distribution. By solving the above-mentioned min-max optimization problem, $\mathcal{A}$ can construct a generated solution set $\mathbb{D}_g$ using $\boldsymbol{z}$ as input.

To intuitively interpret the feasibility of our solution probing attack, we demonstrate the attack using a synthetic dataset for a single crowdsourcing task requiring 256 numeric solutions. $\mathcal{A}$ probes half of the solutions and then generates and submits another half. For proof of concept, the probed solutions are generated randomly following a normal distribution. Valid solutions are specified by the requester, ranging from 0.2 to 0.8. Meanwhile, 0.5 is predefined as the optimal solution, being rewarded with a 1.0 value. Otherwise, valid solutions are rewarded with values decreased linearly. During the probing procedure, $\mathcal{A}$ probes 128 solution-reward pairs as subsequent training data and trains a GAN with the probed data sufficiently. As for the submitting procedure, $\mathcal{A}$ generates 128 candidate solutions. Fig. 2 shows both the probed and the generated solutions with the corresponding rewards. As we can see from the figure, most of the generated solutions obtain expected rewards, indicating that the distribution of valid solutions and their hidden relationship to the rewarding strategy can be correctly captured by $\mathcal{A}$.

Noting that solutions under *A0* assumption are recorded in plaintext, the adversary is free to collect solution-reward pairs on the BDCS instead of probing all the time. Actually, we find that the adversary can generate more precise solutions after observing solution-reward pairs of honest workers. To fully exploit the transparent public ledger of the BDCS, we recommend conducting the probing and submitting by turns. More specifically, $\mathcal{A}$ trains $\mathcal{G}$ using the observed pairs of solutions and rewards and then generates self-assured solutions for probing. After the requester feeds back the rewards according to $\mathcal{R}_T$, $\mathcal{A}$ trains $\mathcal{G}$ using the feedback and more observed solution-reward pairs for calibration. We call this procedure a round of probing attack. By training $\mathcal{G}$ iteratively in this way, $\mathcal{A}$ can get expected rewards with fewer efforts. Let $((d_1, cash_1), (d_2, cash_2), \ldots, (d_n, cash_n))$ represent solution-reward pairs probed by the adversary $\mathcal{A}$ in each round, $n \leq N$. We can summarize the workflow of the solution probing attack with plaintext solutions in Algorithm 1.

Algorithm 1 Solution probing attack

Require: public ledger PL of BDCS, the minimal solution number N and the generated solution number M for task T, prior noise $\boldsymbol{z}$, probing round t.
Ensure: generated solutions $\mathbb{D}_g$.

```
set probing size n ← ⌊(N − M)/t⌋
set generating size m ← ⌊M/t⌋
set 𝔻 ← ∅, 𝔻_g ← ∅
for i = 0 to t do
    D ← collect ⌊n⌋ solution-reward pairs ((d_1, cash_1), (d_2, cash_2), ..., (d_n, cash_n)) from PL
    𝔻 ← 𝔻 ∪ D
    train 𝒢 with 𝔻 using Equation 5
    D_g ← generate m solutions (d_1, d_2, ..., d_m) by 𝒢(z)
    submit D_g to PL for task T and get rewards (cash_1, cash_2, ..., cash_m)
    D ← ((d_1, cash_1), (d_2, cash_2), ..., (d_m, cash_m))
    𝔻 ← 𝔻 ∪ D, 𝔻_g ← 𝔻_g ∪ D_g
end for
return 𝔻_g
```

B. Attack with Ciphertext Solutions

The probing attack is effective when plaintext solutions are accessible. However, it is not clear whether the probing attack works under *A1* assumption, where solutions are protected by encryption as suggested in secure BDCS studies [12], [15]. In this circumstance, the adversary cannot collect solution-reward pairs freely from the public ledger of BDCS. In other words, it is more challenging to implement the solution probing attack with ciphertext solutions since the adversary has no external information about the relationship between solutions and rewards. Therefore, the adversary has to discover the hidden relationship in a tricky way. Fortunately, although workers' solutions are encrypted, rewarding transactions are still accessible on the public ledger of BDCS. Given this advantage, the adversary can enhance the original attack by making adequate modifications to the probing phase. Noticing that rewards are correlated with solutions honestly, it is still feasible for the adversary to infer solutions intuitively.

Since solution-related information is reduced in the adversary's view under *A1* assumption, the first question for

the adversary is to discover the rewarding strategy used by the target task. The underlying challenge is how to find out the hidden relationship using fewer solution-reward pairs. To tackle the problem, the adversary uses a bootstrapping technique [36] to boost the quality of the generated solutions, which utilizes a re-sampling method to estimate the distribution of valid solutions rapidly. In short, the bootstrapping process independently samples from the probed data with replacement and then estimates the distribution based on all data obtained. Borrowing from bootstrapping the re-sampling idea, the adversary augments solution-reward pairs according to a straightforward strategy, i.e., the higher the reward of a solution, the more often this solution is sampled. After bootstrapping, the adversary has a better chance to learn the relationship between solutions and rewards.

Since honest workers' solutions are encrypted, only the solution-reward pairs relevant to the adversary are available for supervision in the generative model training. To accelerate model training and improve the generated solution quality, we train the generative model using solutions and rewards together instead of training with solutions only, which means that the generative model yields solutions and rewards concurrently. Although the generated rewards are not used for transactions on the BDCS, they still provide supervising information by comparing them with real rewards obtained from the adversary's probing. Meanwhile, the generated reward can be seen as a prediction of the evaluating result of a generated solution. When the generative model is well-trained, the adversary can generate enough solution-reward pairs and sort them according to the rewards in descending order. Solutions at the top will be submitted to the BDCS for evaluation. Algorithm 2 summarizes the attack with ciphertext solutions.

Algorithm 2 Solution probing attack with ciphertext solutions

Require: public ledger PL of BDCS, the minimal solution number N and the generated solution number M for task T, prior noise $\boldsymbol{z}$, maxinum probing round t_m, bootstrapping steps b.
Ensure: generated solutions $\mathbb{D}_g$.
1: set generating size $m \leftarrow \lfloor M/t \rfloor$
2: set $\mathbb{D} \leftarrow \emptyset$, $\mathbb{D}_g \leftarrow \emptyset$
3: initialize parameters of $\mathcal{G}$ randomly
4: **for** $t \leq t_m$ **do**
5: generate M solution-reward pairs $((d_1, cash_1)$, $(d_2, cash_2)$, …, $(d_M, cash_M))$ by $\mathcal{G}(\boldsymbol{z})$
6: sort d_i by $cash_i$ in descending order, $i \in [1, M]$
7: $D_g \leftarrow$ select solutions $(d_1, d_2, \ldots, d_m)$ with m top-ranked rewards
8: $\mathbb{D} \leftarrow \mathbb{D} \cup D_g$, $\mathbb{D}_g \leftarrow \mathbb{D}_g \cup D_g$
9: submit D_g to PL for task T and get real rewards $(cash'_1, cash'_2, \ldots, cash'_m)$
10: $\boldsymbol{e} \leftarrow$ calculate estimating error by $|(cash_1, cash_2, \ldots, cash_m) - (cash'_1, cash'_2, \ldots, cash'_m)|$
11: **for** $j = 0$ to b **do**
12: $d_j \leftarrow$ re-sample d_i proportional to e_i^{-1} from D_g
13: $\mathbb{D} \leftarrow \mathbb{D} \cup \{d_i\}$
14: **end for**
15: train $\mathcal{G}$ with $\mathbb{D}$ using Equation 5
16: **end for**
17: **return** $\mathbb{D}_g$

C. Mix-and-Match Defenses

Although encrypting solutions makes it harder for the adversary to probe solutions, it is still possible to implement the attack using Algorithm 2. Thus, it is natural to discuss possible defenses against the probing attack. By observing the attack with ciphertext solutions, we can easily conclude that the breach is rewarding information. Then, the intuition of a possible defense is to protect the solutions with encrypting and the rewards with mixing.

To prevent BDCS from the solution probing attack, a *mix-and-match* strategy-based defense is designed in [1]. The *mix* phase requires the BDCS platform to protect rewarding information by mixing up k rewards for different solutions, while the *match* phase requires each worker to participate in at least k crowdsourcing tasks for rewards. In the mix phase, rewarding values in transactions to be mixed together will be masked by a zero-sum mask. Therefore, the additive mask can be removed when k tasks are finished. Please note that it is practical to require workers to complete at least k tasks since workers concurrently participate in multiple tasks of the same platform in most real-world cases [29], [37].

Assuming that the task set assigned to a worker W_i using well-developed schemes like [38] is $\boldsymbol{t}_{W_i} = \{T_1, T_2, \ldots, T_m\}$, $i, m \in \mathbb{N}^+$. For security concerns, the BDCS matches each task with $k-1$ companion tasks in $\boldsymbol{t}_{W_i}$, $1 < k < m$. Since $\boldsymbol{t}_{W_i}$ may be updated as more tasks are published, companion tasks can always be available for any task in $\boldsymbol{t}_{W_i}$. Assuming that we have companion task set $\boldsymbol{t}_{W_i}^{T_j}$ for task T_j by randomly selection in $\boldsymbol{t}_{W_i}$, then W_i is required to accomplish all tasks in $\boldsymbol{t}_{W_i}^{T_j}$ before T_j's corresponding reward $cash_{W_i,T_j}$ can be collected.

Specifically, when a worker W_i chooses to participate in a task T, the BDCS will match another $k-1$ tasks available to W_i for T, $1 < k < m$. On W_i's task registration, requesters R_j owning these k tasks should generate random numbers r_{R_j,W_i}, $j \in [1, k]$ using a verifiable random source, which is available on many off-the-shelf blockchain implementations like [39]. By exchanging random numbers with each other secretly, requesters of k tasks can agree on a mask $s_{T,W_i}^{R_j}$ regarding T for R_j, satisfying $\sum_{l \in [1,k], l \neq j} s_{T,W_i}^{R_l} = -r_{R_j,W_i}$. When solutions submitted by W_i are evaluated, R_j sends a masked reward $cash_{W_i,T} + r_{R_j,W_i}$ to the BDCS, which is usually implemented using a transaction to a smart contract. In order to be verified by miners in the blockchain, a proof $\Pi(T, R_j, W_i, d_i)$ should also be sent to the platform. If the proof is correct, then W_i will accept R_j's commitment to the reward and continue to finish the other $k-1$ tasks. If W_i's solutions to other tasks are also valid, then each requester R_l of a companion task will send a masked reward $cash_{W_i,T_l} + s_{T,W_i}^{R_l}$ along with the proof $\Pi(T_l, R_l, W_i, d_i)$ to the BDCS. Once all masked rewards are collected via transactions, a total reward $cash_i = cash_{W_i,T} + r_{R_j,W_i} + \sum_{T_l \in \boldsymbol{t}_{W_i}^T} cash_{W_i,T_l} + s_{T,W_i}^{R_l}$ will

be calculated for W_i. Since $r_{R_j,W_k} + \sum_{l\in[1,k],l\neq j} s^{R_l}_{T,W_i} = 0$, the total reward will be a sum of rewards of T and other $k-1$ companion tasks.

V. Solution Probing Attack against Coin Mixing based BDCS

The key to defending against solution probing attacks is to conceal the relation between solutions and rewards. However, transactions on the blockchain require rewards to be published in plaintext, which adds a large limitation in defenses. The mix-and-match defense introduced in [1] is a primary attempt to obfuscate rewarding information. Inspired by the mix-and-match defense, we further utilize coin mixing methods [16], [17] to preserve rewards from the adversary. Coin mixing is a popular technique that mixes the cryptocurrency funds from different users to obscure the source and destination of transactions. To this end, various mixing methods have been studied in previous work, such as [16], [23], [24]. Using different mixing methods can result in different defense solutions. Unfortunately, through our study of coin mixing based BDCS, none of the implementations can be entirely secure under the improved solution probing attack. Although different coin mixing methods may vary in implementation details, they share the same core idea of coin mixing, which is the basis of BDCS with coin mixing. From this perspective, we can use a relatively strong mixing implementation for the defense construction, named CrowdMix. For a comprehensive study of the effect of mixing implementations, we will further implement defenses using other mixing methods [16], [23], [24] and evaluate them fairly. Thus, let us focus on CrowdMix first. As long as CrowdMix can preserve transaction privacy, it is effective in defeating the original solution probing attack [1].

A. CrowdMix defense solution for BDCS

As demonstrated by the probing attack with ciphertext solutions, the adversary $\mathcal{A}$ still can pair the solutions and rewards when acting as a semi-honest worker. The encryption of solutions can only make a limited difference, but the value of rewards in transactions must be plaintext on a public ledger. To this end, we introduce a defense solution for BDCS, named CrowdMix, based on coin mixing, preserving reward transactions. CrowdMix adopts a centralized coin mixing scheme. The original transactions will be mixed and delivered by an appointed *Mixer*. In this way, the identification information of all parties involved in the transactions will be hidden. To achieve this goal, the Mixer itself should have multiple identifications for transaction preservation.

Generally, the payer of a transaction transfers the coins to the mixer first. Then, the mixer mixes them with other coins and transfers the remaining deducted service fee to the appointed remittee. The mixer is also supposed to provide the payer with valid proof of transactions to the remittee. Furthermore, in the BDCS scenario, the mixer sums up the value of transactions for the same remittee individual and transfers a coin of the total value to the remittee in a single transaction. It can confuse the real value of rewards and hide the identities of payers simultaneously. Specifically in BDCS, when having received the i-th solution from a particular worker in account pk_R^{TK}, the requester will publish a transaction of $cash_i$ from account pk_R^{TX} to the mixer on the blockchain. The corresponding task information is encrypted by the mixer's public key and packed in this transaction. After receiving all K tasks from the worker, the mixer will mix the rewards and publish a joint transaction to the worker. All the tasks will be declared in this transaction in plaintext. Now, the reward for a worker having done K tasks is

$$R = (1-\omega)\sum_{i=0}^{K} cash_i, \tag{6}$$

where ω denotes the service fee ratio for coin mixing. To demonstrate how the blockchain works in CrowdMix and how each role interacts with others, we use Fig. 3 to show the workflow of task submission and transactions generation in CrowdMix.

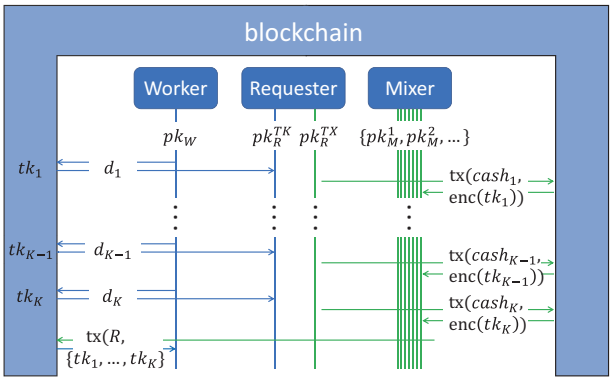


Fig. 3. Workflow of CrowdMix.

After coin mixing, only the transaction containing R is directly rewarding information relevant to a worker, which is irrelevant to the requester's real identity and the worker's solutions. So, it is impossible for an adversary to pair a solution and the corresponding reward from the public ledger and build the probed dataset $\mathbb{D}$ like before. Although BDCS with CrowdMix has fair resistance to the solution probing attack, there still exists a risk of information leakage in the mixed transactions. To disclose the underlying relationship between encrypted solutions and mixed transactions, the adversary can submit more solutions as a worker, which means that the adversary trades more workload for the task knowledge.

We investigate the vulnerability of CrowdMix by devising a new solution probing attack. The newly proposed attack aims to generate the solution-reward pairs set $\mathbb{P}$ for probing attacks under different assumptions, i.e., whether the K tasks come from a single requester or multiple requesters.

B. Attack against Single-Requester CrowdMix

When all K tasks undertaken by a worker are from a single requester, it means that the requester publishes tasks using different pseudonyms. So as rewarding. In this case,

the advantage for an adversary is that the rewarding strategy is fixed for a single requester. Although the requester may confuse the public by using pseudonyms and coin mixing, the adversary can link these pseudonyms, having the same task requirement and using the same rewarding strategy. Inspired by the linkage attack proposed in [33], we give a transaction linkage analysis procedure in Algorithm 3, identifying a single requester with different pseudonyms on the public ledger of BDCS.

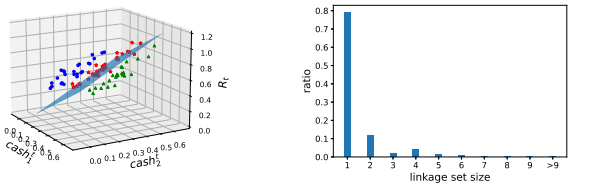
Algorithm 3 Linkage analysis for single-requester mixing.

Require: public ledger PL of BDCS, the minimal solution number N and the generated solution number M for task T, maxinum probing round t_m.
Ensure: solution-reward pairs set $\mathbb{P}$.

initialize $U \leftarrow \emptyset, H \leftarrow \emptyset$
for $t \leq t_m$ **do**
  scan the transactions pool $\mathbb{T}$ and sender set S from PL
  initialize $U^t \leftarrow \{(u_j, R_j) | u_j = \emptyset, R_j = 0, j <= |S|\}$
  for tx in $\mathbb{T}$ **do**
    if sender of tx is S_j **then**
      $U^t_{S_j} \leftarrow (u_j \cup \{cash_{\text{tx}}\}, R_j + cash_{\text{tx}})$
    end if
  end for
  delete (u_j, R_j) in U_t when $|u_j| \neq K$
  if $|U_t| == 1$ **then**
    append (pk_R^{TK}, pk_R^{TX}) of S_1 to H
    append $\{(d_i^t, cash_i^t)\}|_{i=1}^K$ to $\mathbb{D}$
  end if
  $U \leftarrow U \cup U_t$
end for
make clustering with U, H and update $\mathbb{P}$
return $\mathbb{P}$

Algorithm 3 shows the process of linkage analysis for single-requester mixing. Assuming that the adversary has submitted K solutions and received the total rewards R_t as a worker in the t-th round, now the adversary will scan all the transactions $\mathbb{T}$ generated by different senders in S during the working interval t_m, which is quantified as from the submitted time of d_1^t to the received time of R_t. Then, transactions from the same sender will be picked and classified into different groups u. Only groups of size K are retained, while smaller or larger groups are filtered out. Ideally, there is only one group in U_t in one round. Then, the adversary can directly pair the rewards in that group with his own solutions and add K pairs into the solution-reward pairs set $\mathbb{P}$. Meanwhile, the common sender account of this group is bounded with the submitting account as (pk_R^{TK}, pk_R^{TX}) of one requester, recorded in a hash table H_t. The above linkage knowledge U and H_t will be updated and used for the next detection heuristically.

But in practice, the linkage set U_t usually has multiple groups mistakenly linked with R_t, causing massive transactions to happen at the same time. Thus, a detailed analysis is needed. We use a feature vector v_j to represent the group u_j, composed of the difference of the total rewards from R_t, the timestamp of each transaction, the account of senders, and many other features. These vectors are divided into 3 clusters



(a) Clustering result. (b) Distribution of linkage set size.

Fig. 4. A demonstration of linkage analysis of transaction mixing ($K = 2$).

by clustering algorithms, which represent the significantly lower, higher, and closely near total rewards than R_t. So, the groups containing the real solutions will probably fall into the last cluster. Experiments prove that clustering can effectively shrink the linkage set. Fig. 4(a) shows the clustering result when $K = 2$. The plane shows the linear additive relationship between $(cash_1^t, ..., cash_K^t)$ and R_t. It is shown that the groups in the linkage set reduce obviously and are mostly distributed close to the plane after clustering. And the real-linked groups are all in the linkage set. Fig. 4(b) shows the distribution of linkage set size. The ratio of 1 comes to 78%, indicating the validity of the linkage attack.

C. Attack against Multiple-Requester CrowdMix

The solution analysis will be more complicated when there are multiple requesters publishing tasks. Generally, when K tasks are published by different requesters, the linkage attack is not able to detect the solution groups. In this case, we introduce a differential analysis method to probe for a valid solution-reward dataset. The adversary continuously samples the value of a solution $\hat{d}$ from an equal-interval differential section $\{df_1, ..., df_{q_m}\}$ and keeps the remaining $K-1$ solutions constant for q_m times. Then, the adversary submits these q_m new solution groups and get the rewards $\{R_q\}|_{q=1}^{q_m}$. Since the other $K-1$ solutions are fixed, the pair $(\hat{d}, R_q)$ can replace $(d_i, cash_i)$ in solution-reward pairs set $\mathbb{P}$ of solution probing attack after normalization. In this way, the real reward sent to the adversary from the target requester can be separated from other requesters' rewards. The workflow of the differential analysis method is given in Algorithm 4.

The differential method can generate the pairs $(\hat{d}, R_q)$ of any size via negligible computation overheads. However, plenty of solution probes are needed when K is quite large. Meanwhile, if too many similar solutions are submitted, the adversary may catch the requesters' attention and precautions. Thus, we try to reduce the submits when the generative model tends to converge. A common way to do this is by generating the pairs locally. Detailedly, we can add zero-sum noise to the other $K-1 (K \geq 3)$ solutions except $\hat{d}$ and suppose the total rewards remain R without real submitting. Then we use the $\{d'_1, ..., \hat{d}, ..., d'_K, R\}$ to train a new generative model to generate valid pairs $(\hat{d}, R)$. The following experiments prove that the local generating can keep the quality at the end of the differential method.

Given the transaction analysis and differential analysis procedures, we can now introduce the whole process of the

Algorithm 4 Differential analysis for multiple-requester mixing.

Require: public ledger PL of BDCS;The maximum attack rounds t_m;the generated solutions $D_g = \{d_i^t, ..., d_K^t\}|_{t=1}^{t_m}$.
Ensure: solution-reward pairs set $\mathbb{P}$, generated solutions $\hat{D}_g$.

initialize differential section $\{df_1, ..., df_{q_m}\}$, $\hat{D}_g \leftarrow \emptyset$
for $t \leq t_m$ **do**
  pick $\hat{d}$ from $\{d_i^t, ..., d_K^t\}$ randomly
  $\hat{D}_g \leftarrow \hat{D}_g \cup \{\hat{d}\}$
  for $q \leq q_m$ **do**
    replace $\hat{d}$ with df_q and submit to PL
    get real rewards R_q
    append $\{\hat{d}, R_q\}$ to $\mathbb{P}$
  end for
end for
return $\mathbb{P}$, $\hat{D}_g$

improved solution probing attack against coin mixing based BDCS. As shown in Algorithm 5, the adversary can choose different strategies depending on whether tasks are assigned by a single requester or multiple requesters. If all K tasks are assigned by a single requester, the adversary can recover solution-reward pairs using the transaction linkage analysis method. If tasks are assigned by different requesters, the adversary needs to re-generate solutions and recover true solution-reward pairs using the differential analysis method. In either case, the adversary will be capable of constructing true solution-reward pairs. Then, the generative model can be trained in the same way as the original probing attack.

VI. Evaluation

To demonstrate the effectiveness of the solution probing attack, we evaluate it under both threshold and quality-related reward policies in plain and privacy-preserving BDCS with varying probing and attack sizes. For the threshold reward policy, once the quality of a solution satisfies the pre-defined conditions, workers will obtain the same amount of rewards, while for the quality-related policy, the higher the quality, the more the reward. Besides, three state-of-the-art truth discovery algorithms, GTM, CRH, and PACE, are utilized for calculating the quality of the solutions.

The proposed solution probing attacks are evaluated separately. The original probing attack [1] will be evaluated in plain BDCS, encryption based BDCS, and mix-and-match based BDCS. The improved probing attack will be evaluated in coin mixing based BDCS since the original attack cannot work in this case. Please note that we evaluate both attacks using different datasets and quality evaluation algorithms. Moreover, the improved solution probing attack will be evaluated in single-requester and multiple-requester cases when coin mixing is applied in BDCS.

A. Experiment Setup

1) Datasets: To demonstrate the effectiveness of the proposed solution probing attack, we have conducted experiments

Algorithm 5 Solution probing attack against coin mixing based BDCS.

Require: public ledger PL of BDCS, the minimal solution number N and the generated solution number M for task T, prior noise $\boldsymbol{z}$, task set size K, maximum probing round t_m, bootstrapping steps b.
Ensure: generated solutions $\mathbb{D}_g$.

1: set generating size $m \leftarrow \lfloor M/t \rfloor$
2: set $\mathbb{D} \leftarrow \emptyset$, $\mathbb{D}_g \leftarrow \emptyset$
3: initialize parameters of $\mathcal{G}$ randomly
4: **for** $t \leq t_m$ **do**
5: generate M solution-reward pairs $((d_1, cash_1), (d_2, cash_2), \ldots, (d_K, cash_M))$ by $\mathcal{G}(\boldsymbol{z})$
6: sort d_i by $cash_i$ in descending order, $i \in [1, K]$
7: $D_g \leftarrow$ select solutions $(d_1, d_2, \ldots, d_K)$ with K top-ranked rewards
8: **if** tasks from single requester **then**
9: perform a round of Algorithm 3 and get $\mathbb{P}$
10: get $(cash'_1, cash'_2, \ldots, cash'_K)$ from $\mathbb{P}$
11: $\boldsymbol{e} \leftarrow$ calculate estimating error by $|(cash_1, cash_2, \ldots, cash_m) - (cash'_1, cash'_2, \ldots, cash'_K)|$
12: **end if**
13: **if** tasks from multiple requesters **then**
14: perform a round of Algorithm 4 and get $\mathbb{P}, \hat{D}_g$
15: $D_g \leftarrow \hat{D}_g$, get $(R_1, \ldots, R_{q_m})$ from $\mathbb{P}$
16: generate q_m new solution-reward pairs $(\hat{d}_1,\hat{R}_1),\ldots,(\hat{d}_{q_m},\hat{R}_{q_m})$ by $\mathcal{G}(\boldsymbol{z})$
17: $\boldsymbol{e} \leftarrow$ calculate estimating error by $|(\hat{R}_1, \ldots, \hat{R}_{q_m}) - (R_1, \ldots, R_{q_m})|$
18: **end if**
19: $\mathbb{D} \leftarrow \mathbb{D} \cup D_g$, $\mathbb{D}_g \leftarrow \mathbb{D}_g \cup D_g$
20: **for** $j = 0$ to b **do**
21: $d_j \leftarrow$ re-sample d_i proportional to e_i^{-1} from D_g
22: $\mathbb{D} \leftarrow \mathbb{D} \cup \{d_i\}$
23: **end for**
24: train $\mathcal{G}$ with $\mathbb{D}$ using Equation 5
25: **end for**
26: **return** $\mathbb{D}_g$

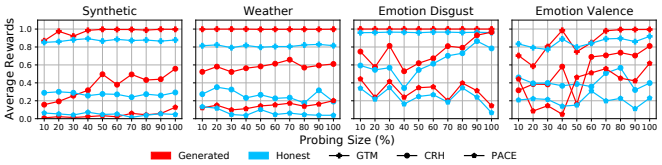


Fig. 5. Solution probing attack against ciphertext BDCS with different probing sizes under a quality-related policy.

on both synthetic and real-world *continues numeric values* datasets, which contain different ranges of numeric values and skewness to simulate the variety of crowdsourcing tasks. The synthetic dataset is first harnessed to demonstrate the effectiveness of the solution probing attack, which contains 256 values $d \sim N(\mu, \sigma^2)$ where μ sampled uniformly from $[10, 30)$ and σ sampled uniformly from $[0, 30)$ correspondingly. The Gaussian distribution is chosen for its wide occurrence in practice, but it

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3355453

TABLE II
REWARDS OF THE GENERATED AND HONEST SOLUTIONS IN CIPHERTEXT BDCS USING A THRESHOLD REWARD POLICY

Dataset	Generated, Honest Solutions		
	max	min	average
Synthetic	**0.95**, 0.72	**0.02**, 0.62	**0.53**, 0.67
	0.78, 0.67	**0.65**, 0.63	**0.72**, 0.66
Weather	**1**, 0.73	**0.67**, 0.65	**0.94**, 0.69
	1, 0.75	**0.83**, 0.59	**0.96**, 0.67
Emotion Disgust	**1**, 0.95	**0**, 0.89	**0.63**, 0.92
	1, 0.92	**0.67**, 0.89	**0.80**, 0.91
Emotion Valence	**1**, 0.93	**0**, 0.77	**0.33**, 0.86
	0.57, 0.90	**0**, 0.70	**0.21**, 0.80

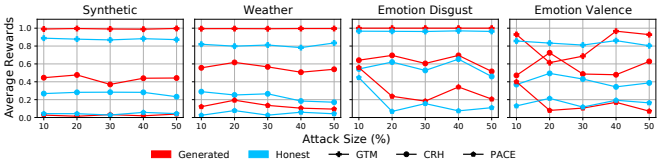


Fig. 6. Solution probing attack against ciphertext BDCS with different attack sizes under a quality-related policy.

TABLE III
REWARDS OF THE GENERATED AND HONEST SOLUTIONS IN BDCS USING MIX-AND-MATCH DEFENSE UNDER A THRESHOLD REWARD POLICY.

Dataset	Generated, Honest Solutions		
	max	min	average
Synthetic	**0.94**, 0.72	**0.32**, 0.60	**0.65**, 0.67
	0.92, 0.66	**0.69**, 0.61	**0.83**, 0.63
Weather	**1**, 0.75	**0.48**, 0.65	**0.79**, 0.69
	0.98, 0.76	**0.33**, 0.63	**0.71**, 0.72
Emotion Disgust	**0.90**, 0.95	**0**, 0.90	**0.16**, 0.93
	0.13, 0.96	**0**, 0.92	**0.03**, 0.94
Emotion Valence	**0.33**, 0.95	**0**, 0.77	**0.10**, 0.84
	0.60, 0.95	**0**, 0.53	**0.12**, 0.81

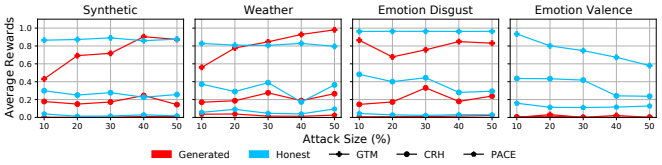


Fig. 9. Solution probing attack against BDCS using mix-and-match defense with different attack sizes under a quality-related reward policy.

is easy for the adversary to find out the pattern once it probes partial data due to the synthesizing strategy. Two benchmark real-world datasets are explored in our experiments. The first dataset is *Weather* [40], which contains weather information of 30 major USA cities from 18 sources. The second dataset is *Emotion* [37], containing numeric assignments from workers to describe their feelings towards some text documents. For the Weather dataset, we choose data from *San Jose* on a given day and for the Emotion dataset, we choose data from *Disgust* and *Valence*, which contain values ranging from $[0, 100]$ and $[-100, 100]$ correspondingly. Those data are explored because they contain the most data and are representative among datasets in value range and skewness. The amount of data in each dataset is 256, 284, 580, and 40 for Synthetic, Weather, Emotion Disgust, and Emotion Valence, respectively. Each dataset is evenly split into two subtasks as tasks of BDCS.

2) GAN and quality rewarding: We implemented the GAN in `Python 3.8.10` with `Pytorch 1.8.0`. The generative model is a four-layer sequential model constructed in linear layers. The discriminative model is also a four-layer sequential model built from linear layers. The regression model the adversary exploits is a three-layer neural network, each of which is constructed in linear layers. The activate function applied for all models is the *rectifier* except for the output of the last layer, which is sigmoid for the generative model and the discriminative model to convert the output to interval $[0, 1]$.

During experiments, μ_0 of the GTM is set to be the average of the submitted solutions, and σ_0 is 1.0 both for the task and the workers. The distance function is chosen to be square distance, and the initial weights of all solutions equal to $\log |\mathcal{D}|$ in the CRH algorithm, in which $|\mathcal{D}|$ denotes the size of the dataset and log is the natural logarithm. Unless explicitly specified, data are normalized to $[0, 1]$ using min-max normalization before training or during the representation.

In the threshold reward policy, the platform first calculates the standard deviation σ of the collected solutions and estimates the truth using three truth discovery algorithms: GTM, CRH, and PACE. It then averages that estimated truth as the final truth τ. A worker is rewarded if and only if its solution d satisfies:

$$|d - \tau| < \sigma \tag{7}$$

3) Evaluation metrics: To quantify the effectiveness of the solution probing attack, we consider two metrics: average rewards of the normal and the generated solutions. Two metrics together quantify the effectiveness of the solution probing attack.

4) Hyperparameter configuration: During the experiments, we evaluate solution probing attack with two different variables: the number of solution-reward pairs the adversary probed and the number of generated solutions the adversary submitted, which are called probing size N and attack size $\mathcal{M}$ respectively. In plain BDCS, N equals to $\mathcal{N}$ due to the public ledger feature of blockchain and $\mathcal{M}$ ranges from 10% to 50%

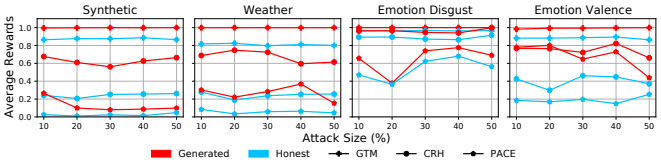


Fig. 7. Solution probing attack against a plain BDCS system under the quality-related policy.

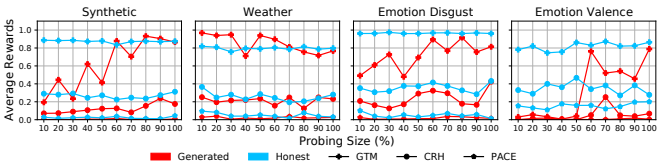


Fig. 8. Solution probing attack against BDCS using mix-and-match defense with different probing sizes under a quality-related reward policy.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3355453

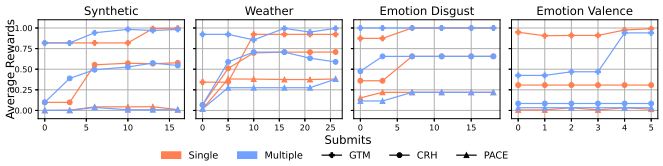


Fig. 10. Attacks against coin mixing based BDCS with different submits.

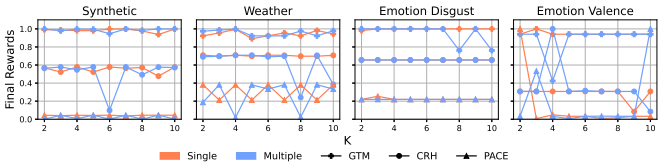


Fig. 12. Rewards obtained by the adversary when attacking on coin mixing based BDCS with different K.

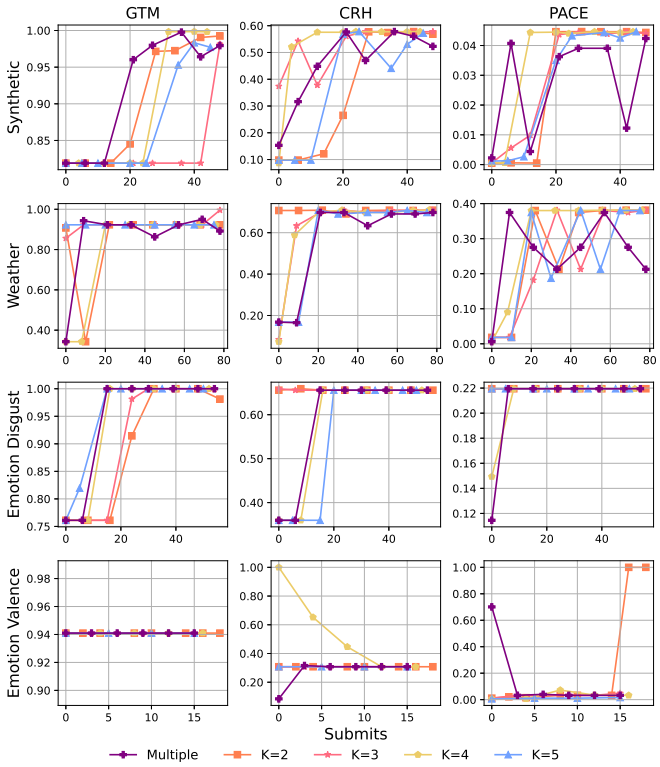


Fig. 11. Attacks against coin mixing based BDCS with different K values.

with step 10% of $\mathcal{N}$. While in privacy-preserving BDCS, when evaluating N related to the attack effectiveness, it ranges from 10% to 100% with step 10% of $\mathcal{N}$ and $\mathcal{M}$ is 30% of $\mathcal{N}$; when evaluating $\mathcal{M}$ related to attack effectiveness, it ranges from 10% to 50% with step 10% of $\mathcal{N}$ and N is 50% of $\mathcal{N}$. For the GAN, the dropout rate between each layer is 30%. Probing and attack sizes are the ratios of the datasets to offset the impact of different data sizes. The base Bootstrapping parameter B is 100 with each training item augmented qB times where q is the normalized solution quality and epochs for training is 600. The loss function of the GAN is chosen to be *BCELoss*. The adversary generates $M = \mathcal{M}|\mathcal{D}|$ solution-reward pairs where $|\mathcal{D}|$ denotes the size of the dataset, then it submits $\mathcal{M}$ solutions with the highest indicated rewards to the platform.

B. Attack against Plain BDCS

In a plain BDCS system with a threshold reward policy, due to the simplicity, all generated solutions are able to reap the rewards from BDCS in all attack sizes in all datasets, while in the honest workers, the average rewards range from 0.67 to 0.90, lower than the generated solutions. Fig. 7 shows the result in plain BDCS under a quality-related reward policy, which presents that the average rewards of honest solutions are higher than their honest counterparts in all quality evaluation algorithms and datasets, proving the effectiveness of the solution probing attack. The figure also shows that the average rewards of the generated solutions are overall not very high. This is caused by the property of quality evaluation algorithms. Notice that with the increasing number of generated solutions, the average rewards have reduced in some datasets, which is caused by regression toward the mean.

C. Attack against Encryption based BDCS

TABLE II presents the experiment results of different probing and attack sizes in each dataset in ciphertext BDCS under threshold reward policy. From the table, we can learn that the average rewards of the adversary are higher than the honest workers when it comes to the maximum obtainable rewards except in Emotion Valence in attack sizes, which is due to the fact that the adversary can only derive solutions from 10 solution-reward pairs. However, the adversary's average rewards are unstable, consistently lower than honest workers, except in Weather, leading to lower overall average rewards for the adversary. The results demonstrate that solutions encryption protects BDCS to some extent, but the adversary is able to carefully choose its strategies in order to reap the rewards from BDCS.

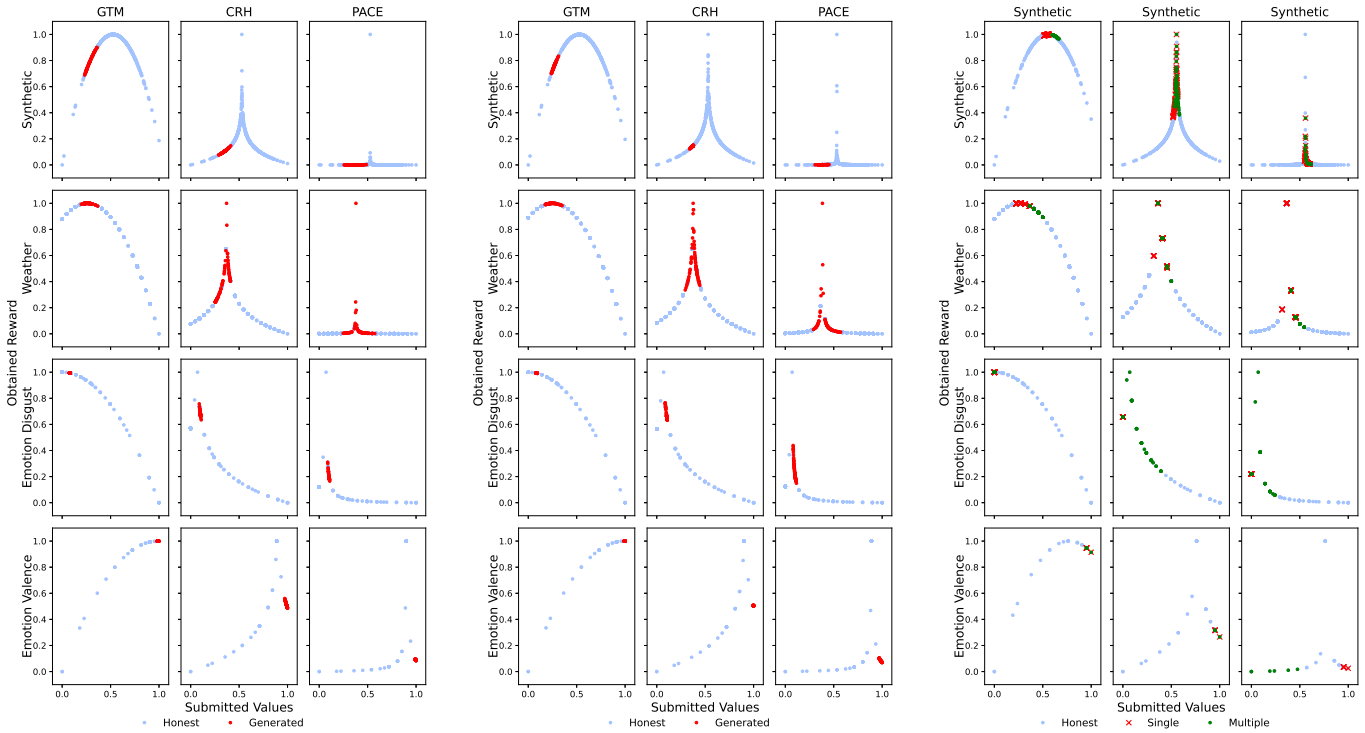
Fig. 5 shows that under quality-related reward policy, the average rewards of the adversary are higher than the honest worker in most cases in all datasets and quality evaluation algorithms, except when the amount of solution-reward pairs probed are small, rendering the regression model unable to derive the desired solutions from the rewards. Fig. 6 demonstrates that probes half of the solution-reward pairs are able to effectively generate high-quality solutions. As more solutions are generated, the average rewards of the adversary are higher than the honest workers, which also demonstrates the effectiveness of the optimizations.

The results of TABLE II, Fig. 5 and Fig. 6 demonstrate that even though the solutions are encrypted, the attacker is still able to launch attacks to generate high-quality solutions, especially in quality-related reward policy, which is widely adopted in practice.

D. Attack against Mix-and-Match based BDCS

The mix-and-match defense experiments are conducted both for the threshold reward policy and the quality-related reward policy, in which the masks of companion tasks are offset under

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3355453



(a) attack to CoinJoin (b) attack to CoinShuffle (c) attack to CrowdMix

Fig. 13. Distribution of solutions generated by solution probing attacks in different defense implementations. Since CoinJoin and CoinShuffle are not explicitly designed for multiple requesters, only CrowdMix is evaluated in the multiple-requester case.

modulo $N = 2^{63} - 1$. TABLE III shows the experiment results of different probing and attack sizes in each dataset of mix-and-match BDCS under threshold reward policy. From the table, we observe that the adversary obtains lower average rewards compared with their privacy-preserving counterparts when it comes to the maximum except in Synthetic. The reason is that the Synthetic dataset follows Gaussian distribution, which means if the generated solutions are random, they will converge to the interval of rewardable threshold, as demonstrated in the average of the generated solutions. However, we can observe that in different attack sizes in Synthetic, the adversary obtains rewards higher than those in privacy-preserving BDCS, which proves the effectiveness of the optimization of the solution probing attacks. The mix-and-match defense destabilizes and lowers the rewards of the adversary, as shown in the min and average columns of the table.

Fig. 8 shows the average rewards of the adversary and the honest workers in mix-and-match BDCS, in which the adversary has varied probing sizes. It shows that except in partial Weather and Synthetic datasets with GTM algorithms, the average rewards of the adversary are lower than honest workers, demonstrating the effectiveness of the mix-and-match defense. However, the average rewards of the adversary decreased compared with ciphertext counterparts in Weather and Synthetic datasets. Fig. 9 shows that when the adversary generates more solutions, it cannot obtain higher average rewards than honest workers except in the Weather dataset with GTM algorithms, which is caused by the fact that when more solutions are generated, it will affect the final result of the estimated quality in GTM algorithms. However, when compared with the ciphertext counterparts, the mix-and-match defense effectively decreases the rewards of the adversary.

Based on the results of TABLE III, Fig. 8 and Fig. 9, we conclude that the mix-and-match defense can effectively decrease the rewards obtained by the adversary, which means the probing attack considered in [1] can be mitigated by transaction mixing techniques.

E. Attack against Coin Mixing based BDCS

The two attacks in coin mixing based BDCS are evaluated under the quality-related policy. Unless otherwise specified, the mixing size K is set to 3, and the service fee ratio ω is set to 0.05 as default. The average rewards of solution probing attack in coin mixing BDCS showed in Fig. 10, decreases obviously in total than plain and privacy-preserving BDCS. It proves the defense validity of coin mixing. Meanwhile, the two attacks make it back on track thanks to the generated high-quality solutions with the increase of submits. And the differential attack stabilizes slower due to its more submits. Considering the influence of mixing size, we also evaluate the average rewards with different submits under the impact of K in Fig. 11 and Fig. 12. It is noted that K is fixed at 3 in differential attacks because K makes no difference in the sampling. The two figures show that further mixing has no remarkable effect on the quality of solutions. On the contrary, the higher K accelerates the attacks due to more submits.

CrowdMix is a specific coin mixing based BDCS implementation introduced in this paper. However, we note that there are various coin mixing methods that can be used for different implementations. As long as transactions on the blockchain can be preserved, the mixing method is suitable for the defense implementation. Although coin mixing or transaction preservation techniques are orthogonal to our work, we try to offer comprehensive evaluation results by adapting different mixing methods into BDCS. In particular, CoinJoin [23] and CoinShuffle [16], [24] are also used for coin mixing based BDCS implementations. Please note that CoinJoin and CoinShuffle are not explicitly designed for multiple requesters. Only CrowdMix will be evaluated in the multiple-requester setting. Fig. 13 shows the evaluation result of the improved attack against three different implementations of coin mixing based BDCS. Through the comparison between the submitted values and the obtained rewards under attacks, we can confirm that the improved solution probing attack is still effective when other mixing methods are used. It is noted that the attack results of CoinJoin and CoinShuffle are similar because a single requester can be identified precisely by the attack. This result implies that transaction preservation or mixing techniques should take into account applications like crowdsourcing, where task assignments may disclose transaction senders.

VII. Conclusion and Limitation

We report a new vulnerability of blockchain based decentralized crowdsourcing systems, which can be leveraged to mount solution probing attacks, ruining data privacy and fair trade in crowdsourcing. We identify that the vulnerability lies in unprotected reward information on a public ledger in a quality-aware crowdsourcing system. However, transaction mixing techniques like coin mixing can significantly mitigate the performance of the probing attack. In our further study, we investigate coin mixing based BDCS systems and introduce transaction analysis methods for disclosing the preserved reward information. In this way, we design a new solution probing attack, corrupting coin mixing based BDCS systems.

We note that our solution probing attack is limited to a semi-honest adversarial model. Malicious behaviors like poisoning attacks are not taken into consideration. An adversary who is capable of poisoning a crowdsourcing task can construct a more threatening solution probing attack since the adversary can affect the truth estimated by the requester. In this way, the adversary can manipulate the truth and defraud the requester for reward, which deserves to be studied in further work.

Acknowledgement

The authors would like to thank the editor and reviewers for the time and efforts they have kindly made in this paper. The attack interpretation and evaluation are significantly improved after revision by following their suggestions. This work was supported in part by the National Natural Science Foundation of China under Grants NSFC-62272222, NSFC-61902176, and NSFC-62272215 and in part by the Natural Science Foundation on Frontier Leading Technology Basic Research Project of Jiangsu under Grant BK20222001.

References

[1] H. Wang, Y. Mao, and S. Zhong, "Solution probing attack against blockchain-based quality-aware crowdsourcing platforms," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2022, pp. 81–89.

[2] M. Miceli, M. Schuessler, and T. Yang, "Between subjectivity and imposition: Power dynamics in data annotation for computer vision," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, pp. 1–25, 2020.

[3] R. Mouawi, I. H. Elhajj, A. Chehab, and A. Kayssi, "Crowdsourcing for click fraud detection," *EURASIP Journal on Information Security*, vol. 2019, pp. 1–18, 2019.

[4] W. Shen, X. He, C. Zhang, Q. Ni, W. Dou, and Y. Wang, "Auxiliary-task based deep reinforcement learning for participant selection problem in mobile crowdsourcing," in *ACM International Conference on Information & Knowledge Management*, 2020, pp. 1355–1364.

[5] B. Zhao, S. Tang, X. Liu, and X. Zhang, "Pace: Privacy-preserving and quality-aware incentive mechanism for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1924–1939, 2020.

[6] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, "Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 1986–1999, 2016.

[7] B. Zhao and J. Han, "A probabilistic model for estimating real-valued truth from conflicting sources," *Proc. of QDB*, vol. 1817, 2012.

[8] X. Gao, H. Huang, C. Liu, F. Wu, and G. Chen, "Quality inference based task assignment in mobile crowdsensing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 10, pp. 3410–3423, 2020.

[9] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudré-Mauroux, "The dynamics of micro-task crowdsourcing: The case of amazon mturk," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 238–247.

[10] J. Zhang, F. Yang, Z. Ma, Z. Wang, X. Liu, and J. Ma, "A decentralized location privacy-preserving spatial crowdsourcing for internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2299–2313, 2020.

[11] C. Li, B. Palanisamy, R. Xu, J. Wang, and J. Liu, "Nf-crowd: Nearly-free blockchain-based crowdsourcing," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2020, pp. 41–50.

[12] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 853–865.

[13] W. Feng and Z. Yan, "Mcs-chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain," *Future Generation Computer Systems*, vol. 95, pp. 649–666, 2019.

[14] H. Ma, E. X. Huang, and K.-Y. Lam, "Blockchain-based mechanism for fine-grained authorization in data crowdsourcing," *Future Generation Computer Systems*, vol. 106, pp. 121–134, 2020.

[15] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "Crowdbc: A blockchain-based decentralized framework for crowdsourcing," *IEEE transactions on parallel and distributed systems*, vol. 30, no. 6, pp. 1251–1266, 2018.

[16] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "P2p mixing and unlinkable bitcoin transactions," *Cryptology ePrint Archive*, 2016.

[17] I. A. Seres, D. A. Nagy, C. Buckland, and P. Burcsi, "Mixeth: efficient, trustless coin mixing service for ethereum," *Cryptology ePrint Archive*, 2019.

[18] N. Lu, Y. Chang, W. Shi, and K. R. Choo, "Coinlayering: An efficient coin mixing scheme for large scale bitcoin transactions," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, 2022.

[19] Z. Zhang, J. Yin, Y. Liu, and J. Liu, "Deanonymization of litecoin through transaction-linkage attacks," in *2020 11th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2020, pp. 059–065.

[20] "Bitcoin is an innovative payment network and a new kind of money." [Online]. Available: https://bitcoin.org/

[21] "Ethereum is the community-run technology powering the cryptocurrency ether (ETH) and thousands of decentralized applications." [Online]. Available: https://ethereum.org/

[22] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407–1419, 2019.

[23] "CoinJoin: Bitcoin privacy for the real world — bitcointalk.org," https://bitcointalk.org/?topic=279249, [Accessed 08-Nov-2022].

[24] T. Ruffing, P. A. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *ESORICS*, 2014.

[25] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "Coinparty: Secure multi-party mixing of bitcoins," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, 2015, pp. 75–86.

[26] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and Distributed System Security Symposium*, 2017.

[27] Y. Zhang, M. Simsek, and B. Kantarci, "Self organizing feature map for fake task attack modelling in mobile crowdsensing," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[28] Y. Zhang and M. Van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2140–2148.

[29] M. Fang, M. Sun, Q. Li, N. Z. Gong, J. Tian, and J. Liu, "Data poisoning attacks and defenses to crowdsourcing systems," in *Proceedings of the web conference 2021*, 2021, pp. 969–980.

[30] Y. Zhao, X. Gong, F. Lin, and X. Chen, "Data poisoning attacks and defenses in dynamic crowdsourcing with online data quality learning," *IEEE Transactions on Mobile Computing*, 2021.

[31] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," *arXiv preprint arXiv:1502.01657*, 2015.

[32] A. Biryukov and S. Tikhomirov, "Deanonymization and linkability of cryptocurrency transactions based on network analysis," in *2019 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2019, pp. 172–184.

[33] S. Goldfeder, H. Kalodner, D. Reisman, and A. Narayanan, "When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies," *arXiv preprint arXiv:1708.04748*, 2017.

[34] G. Wu, L. Zhou, J. Xia, L. Li, X. Bao, and X. Wu, "Crowdsourcing truth inference based on label confidence clustering," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 4, pp. 1–20, 2023.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[36] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 569–593.

[37] R. Snow, B. O'connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks," in *Proceedings of the 2008 conference on empirical methods in natural language processing*, 2008, pp. 254–263.

[38] L. Wang, D. Yang, X. Han, D. Zhang, and X. Ma, "Mobile crowdsourcing task allocation with differential-and-distortion geo-obfuscation," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 967–981, 2019.

[39] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*. Springer, 2018, pp. 66–98.

[40] L. D. Laure Berti-Equille. Data sets for data fusion experiments. [Online]. Available: https://lunadong.com/fusionDataSets.htm

Yunlong Mao received B.S. and Ph.D. degrees in computer science from Nanjing University in 2013 and 2018, respectively. He is currently an associate professor at Nanjing University. His current research interests include security, privacy, machine learning, and blockchain.

Ziqin Dang is a graduate student in the Department of Computer Science at Nanjing University. His research interests include security, privacy, and machine learning.

Heng Wang was a graduate student in the Department of Computer Science at Nanjing University from 2019 to 2022. Now, he is a software engineer at Pony.ai.

Yuan Zhang received the BS degree in automation from Tianjin University, Tianjin, China, in 2005, the MSE degree in software engineering from Tsinghua University, Beijing, China, in 2009, and the PhD degree in computer science from the State University of New York at Buffalo, Buffalo, New York, in 2013. His research interests include security, privacy, and economic incentives.

Sheng Zhong received the B.S. and M.S. degrees from Nanjing University in 1996 and 1999, respectively, and the Ph.D. degree from Yale University in 2004, all in computer science. He is interested in security, privacy, and economic incentives.