1. Intro

Hi, everyone. I'm Liang Li. It’s really a great pleasure to be here and I am very happy to introduce our paper: **Bilateral Secure and Decentralized Crowdsourcing Task Matching Atop Consortium Blockchain.** The co-authors include Haiqin Wu...

2. What is Crowdsourcing?

First of all, we review what is crowdsourcing. Crowdsourcing relies on numerous people working collaboratively to solve complex problems. For example, a company wanting to collect weather data in Shanghai can recruit individuals to report weather conditions from various districts. This approach is cost-effective, fast, and creative compared to traditional outsourcing.

3. Task Matching in Crowdsourcing

Crowdsourcing involves three entities: the requester is a task owner who publishes tasks to the server for worker recruitment, and the worker is a task executor. The crowdsourcing server conducts the task matching.

Task matching needs to assign tasks to qualified workers. In crowdsourcing, the requester first uploads the task information to CS, such as task content, task requirements, and task type. CS assigns task to a qualified worker according to requester-side task requirements or/and worker-side task preferences.

Task matching should be correct, which means the matched worker actually meets the task requirements and the task meets the worker's preferences. The matching algorithm also should be efficient. Furthermore, as we can see, the matching algorithm needs some sensitive information as input, and it is conducted on a dishonest server. The server may sell the user's private information for profit. So, how to protect user privacy is important.

4. Existing Solutions—Traditional Model

In recent years, a line of work has been presented to solve the privacy leakage problem in task matching. The existing schemes mainly contain three types: auction-based, location-based, and keyword-based.

These models are only based on a single attribute to achieve task matching. These schemes require a cloud server assumed to be semi-honest.

For privacy, commonly used privacy protection techniques include Cryptographic commitment, hash function, encryption algorithm, and so on.

5. Existing Solutions—based on Blockchain

The dishonest centralized crowdsourcing suffers from the notorious single-point-of-failure and collusion attacks. It may lead to unreliable matching and insufficient privacy protection issues.

Therefore, to solve these problems, blockchain-based crowdsourcing systems were proposed. Researchers use the properties of blockchain to build a decentralized crowdsourcing system.

And, it's easy to combine blockchain and crowdsourcing, we just replace the centralized server with smart contracts to achieve reliable task matching like this figure. Right, it sounds simple. However, this will bring new privacy leakage challenges due to the transparency of smart contracts.

6. Limitations of the Existing Solutions

In a word, the existing solutions have the following limitations: The matching model is oversimplified, which means most schemes are based on coarse-grained location or keyword and only consider one-party requirements and privacy. Additionally, although some schemes use the blockchain to replace the centralized server, these schemes are not really decentralized because there still exists a trusted entity for distributing user keys. Finally, the on-chain operation suffers from high overhead.

Therefore, a decentralized crowdsourcing with dual-side privacy-preserving and flexible task matching is required.

7. Contributions

In our paper, we proposed a decentralized crowdsourcing system with no central trust and dual-side privacy-preserving task matching. We also achieved flexible task matching. Our scheme also achieves the correctness and public verifiability of matching results.

Next, I will introduce the technical roadmap of our scheme.

8. ABE

First, I will introduce the main building block of our scheme, attribute-based encryption. ABE extends traditional public-key encryption to enable fine-grained access control to encrypted data. ABE has two variants: ciphertext-policy abe and key-policy abe. As in this PowerPoint, in ciphertext-policy abe, the Encryptor assigns an access policy f in ciphertext, and the decryption is successful if and only if the set of attributes of the Decryptor satisfies the access policy.

9. Design Rationale of Our Scheme

In our scheme, the requester uploads the task ciphertext to the blockchain with task requirements denoted by vector x. The worker retrieves the attribute key for vector v from the authority and generates a search token using the TokenGen algorithm. The smart contract then performs the Search algorithm to match tasks if the inner product of v and x is 0 and the task and worker keywords match. Finally, the worker retrieves and decrypts the task ciphertext from the blockchain.

This ensures that both requester and worker data are protected by encryption and the search token algorithm. It allows flexible task matching based on specific requirements and keywords. The smart contract guarantees reliable and publicly verifiable results.

10. Vector Generation

Now, I will explain how to transform the task requirements and attributes to vectors.

The AND-Gate access structure with wildcard is used in our scheme. In this access structure, each attribute has multiple possible values, the notation '+' denotes this value is positive, '-' denotes the opposite, and '*' denotes the don't care attribute. For example, in this table, Alice is a teacher in the CS department of University A, and Alice satisfies access structure W2 rather than W1.

We compute the vectors x and v as these two formulas.

11. Vector Generation—Example

For easy understanding, we give an example. For Alice, the set $J$ of the positive attribute value is {1,4,6} and we set N equals 2. Then we compute $v$ in this way. V zero equals 1 to zero... Then the vector $v$ equals (3,11,53,-1).

For access policy $W_2$, the positive attribute value set is Q and the wildcard attribute value set is P. Then we compute the vector x zero to x two and Gamma in this way.

Finally, it is easy to check the inner product of v and x is zero.

12. Pedersen Commitment

Next, I will introduce a cryptographic commitment Pedersen Commitment, that allows one to generate a commitment to a message m without revealing m. At a later time, the commitment is opened by the sender to convince a receiver that the message is indeed m.

The commit algorithm takes as input a message m and a random element in $\mathbb{Z}_p$ and outputs a commitment value com=$g^mh^r$.

The verify algorithm outputs 1 if com equals g to m times h to r, otherwise, outputs 0.

We utilize the homomorphic property of the Pedersen Commitment, which means the multiplication of commitment of $m_1$ and $m_2$ equals the commitment of $m_1+m_2$.

13. Technique Overview

Now I'd like to move on to the technique overview. Note that traditional ABE doesn't have a keyword search algorithm. To enable keyword search, we design $TokenGen$ algorithm and $Search$ algorithm by adding $\beta$-components to ciphertexts and search tokens.

And then utilize the Viete formulas to generate policy and attribute vectors to reduce the ciphertext and key size.

Finally, to resist malicious workers submitting incorrect attribute vectors, we design $ProofGen$ algorithm and $ProofVerify$ algorithm based on Pedersen Commitment.

After that, the whole process of task matching is given.

14. System Initialization

First, in system initialization, the setup and authsetup algorithms are performed to generate public parameters and authority's public keys.

In the authority setup algorithm, we add the $\beta$ components to enable later keyword search.

15. User registration-1

Next, the worker computes his attribute vector v and sends it to the authority to get secret keys. To resist malicious workers submitting wrong vectors, authorities first perform proofGen for this worker. D is worker's attribute value managed by AU_i. AU_i computes commitment for D as this formula. Finally, the authority can perform proofverify algorithm to verify the vector.

For ease of understanding, we can see this example. Alice computes her vector as v. And then authorities compute the commitments of v as the table. For attribute 1, the commitments are in the first row, similarly, for attribute 4, the commitments are in the second row.

Authority can check the $v_3$ to satisfy this equation. we can conclude if the worker correctly computes v, then we have this equation for $v_1, v_2$, and $v_3$.

16. User registration-2

After verification, the authority generates secret key for vector v as this formula. Notably, $key_{i1}$ will be used to generate search token.

17. Task ciphertext uploading

In this phase, requester encrypts the task and uploads it to blockchain for storage. Hybrid encryption is employed. The requester first selects a random symmetric key sk from $G_T$. And sk is encrypted by ABE with access policy x. In addition, this cipher also contains keyword-associated components: $C_1$ and $\tilde{C}$.

18. Search Token Generation

The worker generates search token tk_v follows this way. We first compute $K'*i=g_2^{t_i}|cdot key*{i,1}$, and the keyword-associated component $tok_1$. Then, upload $tk_v$ to blockchain for task matching.

19. Reliable On-chain Task Matching

After that, the smart contract conduct mathcing with tk_v and ct_x.

First, the smart contract computes e_1, this formula is responsible for testing if the inner product of v and x equals 0.

After that, task matching is successful if this equation holds. This equation is responsible for testing keyword relations.

we can see the left part and right part of this equation are the same if v times x equals 0 and keyword w equals w'. At this point, our task matching is completed

20. Task Retrieval and Decryption

Finally, the worker retrieves task cipher from blockchain and decrypts it to perform the task.

21. Experiment Setting

Next, let me briefly talk about the experimental performance.These are some experiment settings. the task matching is deployed on hyperledger fabric. And we use a test tool called Tape to simulate transactions.

22. Off-chain

These graphs show the off-chain performance, we tested the time overhead of each algorithm with different $N$. The overhead of algorithms is increasing with $N$ because of the number of time-consuming operations related to $N$. Our scheme has a reasonable overhead.

23. On-chain-1

This is on-chain performance, Fig.(a) and fig.(b) show the throughput and latency of the task uploading phase and task matching with different N. Task matching is more time-consuming because it has more pairing operations. Moreover, our scheme is more efficient than a task-matching scheme DABTA.

24. Summary

Well, that concludes my presentation today, to refresh your memory, I'll repeat the main points:

we design a policy-hiding multi-authority searchable ABE. we discover keyword search based on traditional ABE and reduce the ciphertext and key size by utilizing viete's formulas.

Secondly, we proposed a decentralized crowdsourcing with no central trust and dual-side privacy-preserving and flexible task matching.

Finally, we also give the security analysis of the scheme in the paper.

25

Thanks for your attention. I am happy to answer any questions you might have.