Utilitas Mathematica
ISSN: 0315-3681 Volume 122, 2025

An efficient task matching schema with signature oriented encryption technique

Arijita Bhowmik[1], Somen Debnath[2†], Parijata Majumdar[3*], R. Chawngsangpuii[4†]

[1]Department of Computer Science and Engineering, Techno College of Engineering, Maheshkhola, Agartala, 799004, Tripura, India.
[2]Department of Computer Science and Engineering, Tripura University, Suriyamaninagar, Agartala, 799022, Tripura, India.
[3*]Department of Information Technology, Tripura University, 799022, Tripura, India.
[4]Department of Information Technology, Mizoram University, Tanhril, Aizawl, 796004, Tripura, India.

*Corresponding author(s). E-mail(s): parijatamajumdar@tripurauniv.ac.in;
Contributing authors: arijita.bhowmik@yahoo.com; somendebnath@tripurauniv.ac.in; mzut126@mzu.edu.in;
†These authors contributed equally to this work.

Abstract

Crowdsourcing platforms have transformed task outsourcing by enabling large-scale human collaboration over the internet. A key factor in their success is the efficient matching of workers to tasks based on individual skills and preferences. However, this process often involves sharing sensitive personal data, raising serious concerns about privacy and security. Existing solutions, such as traditional encryption and proxy re-encryption, struggle to balance scalability with strong privacy guarantees. In this paper, we propose a privacy-preserving task-matching framework that utilizes ring verification signatures to safeguard user anonymity and ensure data integrity. Unlike conventional methods, our approach removes the dependency on proxy re-encryption and effectively mitigates risks related to identity inference. We detail the system's design, implementation, and security analysis, demonstrating its practical effectiveness in preserving privacy within dynamic and large-scale crowdsourcing environments.

Keywords: Crowd sourcing, Identity Anonymity, Cryptographic Protocols, Secure Crowdsourcing Systems

1 Introduction

Crowdsourcing has become a powerful paradigm for tackling complex problems by leveraging the collective intelligence of individuals worldwide. Enabled by the proliferation of the internet, this model has grown rapidly over the past decade, leading to the emergence of prominent platforms such as Upwork, Amazon Mechanical Turk, and CrowdFlower. These platforms serve as intermediaries, linking task requesters i.e., those who post tasks with workers who are willing to complete them in exchange for compensation [1, 2]. A crucial factor in the effectiveness of such platforms is their ability to accurately match tasks with workers whose skills and preferences align with the task requirements.

At the core of any crowdsourcing system lies the task-matching process. This involves backend infrastructure, commonly referred to as the *crowd-server*, that connects workers to suitable job postings. However, this process typically requires access to sensitive personal information, including user identities, locations, and professional backgrounds. The exposure of such data raises significant concerns regarding user privacy and data security [3].

To ensure privacy while preserving the efficiency of task matching, robust encryption techniques are essential. Traditional encryption approaches, however, often fall short in balancing privacy, scalability, and accountability. For instance, using a common encryption key across users can compromise transparency, while generating unique keys for each task may lead to excessive data redundancy. Proxy re-encryption (PRE) has been introduced as a potential solution, yet it introduces security vulnerabilities through re-keying and adds operational complexity [4].

In light of these limitations, recent attention has turned to proxy-free encryption methods, which use a master key to derive user-specific keys. These approaches mitigate some drawbacks of PRE, but the risk of identity inference persists—especially when personal information such as home addresses, interests, or occupations is indirectly exposed. This underscores the need for stronger anonymity-preserving mechanisms within task-matching systems.

To address this challenge, advanced cryptographic techniques such as asymmetric encryption and digital signatures have gained prominence. Among them, *ring signatures* stand out for their ability to validate the authenticity of information without revealing the identity of the signer. By enabling a message to be verified as coming from a member of a predefined group without specifying which one ring signatures help preserve anonymity while ensuring data integrity [5].

In this paper, we propose a novel privacy-preserving task-matching framework that integrates ring verification signatures with searchable encryption to enhance pri- vacy in crowdsourcing environments. Our approach eliminates the need for proxy re-encryption and addresses the key vulnerabilities found in conventional systems.

The framework ensures that both task requesters and workers can interact securely without revealing sensitive personal details or compromising anonymity.

Main Contributions

Privacy-Preserving Architecture: We design a novel crowdsourcing framework that integrates ring signatures with searchable encryption, ensuring robust identity protection.
Secure Task Publication: Task requesters can encrypt task requirements and content while keeping their identities concealed.
Private Worker Queries: Workers generate encrypted trapdoors to query tasks securely, without exposing query terms or personal identities to the crowd-server.
Encrypted Task Matching: A lightweight protocol enables efficient matching between encrypted task data and worker queries, even on untrusted servers.
Formal Security Analysis: We rigorously prove our scheme's security in the Random Oracle Model, demonstrating:

- Unforgeability
- Anonymity and unlinkability
- Non-repudiation
- Collusion resistance
- Traceability by a trusted authority

We also present the architecture and implementation details of the proposed scheme and evaluate its effectiveness in protecting sensitive user data within a dynamic and scalable crowdsourcing environment.

The remainder of this paper is organized as follows:

Section 2 reviews related work.
Section 3 outlines the necessary preliminaries.
Section 4 defines the security model, threat assumptions, and design goals.
Section 5 explains the schema constructon.
Section 6 discusses the security analysis of the proposed method.
Section 7 discusses experimental results.
Section 8 concludes the paper.

2 Related Work

Ring signatures have emerged as a powerful cryptographic primitive that offers both data authenticity and user anonymity, making them particularly well-suited for privacy-sensitive applications like crowdsourcing. Introduced by Rivest, Shamir, and Tauman [6], the ring signature allows any member of a defined group to sign a message on behalf of the group without revealing their individual identity. This property makes it ideal for scenarios where privacy and accountability must coexist.

Building upon this foundation, Bresson et al. [7] introduced threshold ring signatures, demonstrating their feasibility even under relaxed security assumptions. Other

early contributions include one-out-of-n signature schemes by Abe et al. [8] and ID-based ring signatures by Zhang and Kim [9]. Xu et al. [10] further advanced the domain by leveraging bilinear pairings, resulting in more efficient and secure constructions.

To address vulnerabilities, Bender et al. [11] proposed stronger security models and revealed that several earlier schemes were prone to public key substitution attacks. Robust alternatives followed: Boneh et al. [12] and Shacham and Waters [13] introduced constructions that improved both efficiency and security—especially the latter, which eliminated the reliance on random oracles. Chandran et al. [14] proposed a sub-linear ring signature scheme, trading off signature size for performance. Later, Gu and Wu [15] designed a constant-size traceable ring signature that ensured accountability without random oracles.

In parallel, certificateless ring signature schemes emerged to address the overhead of managing certificates. Deng et al. [16], Chow and Yap [17], Zhang et al. [18], and Chang et al. [19] each contributed designs that eliminate the need for a central certificate authority while ensuring strong security guarantees.

The application scope of ring signatures has expanded to domains like electronic voting [20], blockchain systems [21], and secure multi-party computation [22], reflecting the growing importance of privacy-preserving group authentication mechanisms. Further refinements have been introduced, such as the threshold ring signatures based on coding theory by Melchor et al. [23], and the universal ring signature model by Tso [24], which allows dynamic group formation without prior setup.

Novelty of Our Work

Despite these developments, ring signatures have yet to be fully integrated into actual crowdsourcing frameworks, notably for secure task matching. Existing encryption techniques, such as classical public-key encryption and proxy re-encryption (PRE), have scalability constraints, high processing overhead, and are vulnerable to identity inference via metadata leakage.

In contrast, we present a unique, privacy-preserving task-matching architecture that combines ring verification signatures and searchable encryption. This eliminates the need for proxy re-encryption, lowering system complexity and increasing scalabil- ity. Unlike previous efforts that only address signature efficiency or accountability, our approach considers user anonymity, task secrecy, and secure matching on untrusted servers. Thus, we try to provide a framework that integrates ring signature verifica- tion with privacy-preserving task search and matching in a large-scale crowdsourcing setting.

3 Preliminaries

A. Bilinear Pairings

Bilinear pairings operate over three cyclic groups G_1, G_2, and G_T , all of prime order p, with respective generators $g_1 \in G_1$, $g_2 \in G_2$. A bilinear map $e : G_1 \times G_2 \rightarrow G_T$ satisfies the following properties:

Bilinearity: For all $u \in G_1$, $v \in G_2$, and $a, b \in Z_p^*$, it holds that $e(u^a, v^b) = e(u, v)^{ab}$.

Non-degeneracy: $e(g_1, g_2) \neq 1$, ensuring the map is not trivial.

Computability: The map $e(u, v)$ can be efficiently computed for any $u \in G_1, v \in G_2$.

Pairings are categorized into three types: Type 1 where $G_1 = G_2$, Type 2 where $G_1 \neq G_2$ but with an efficient isomorphism between them, and Type 3 where $G_1 \neq G_2$ with no efficient isomorphism, often preferred for better performance and security.

B. Complexity Assumptions

Security of many cryptographic protocols is based on the hardness of certain assumptions over bilinear groups:

BDH (Bilinear Diffie–Hellman) Assumption: Given $(g, g^a, g^b, g^c) \in G_1$, compute $e(g, g)^{abc} \in G_T$. This is assumed hard for any probabilistic polynomial-time (PPT) adversary.

q-SDH (Strong Diffie–Hellman) Assumption: Given $(g_1, g_2, g_2^{\xi}, \ldots, g_2^{\xi^q})$, compute $(g_1^{1/(x+\xi)}, x)$ for some $x \in Z_p^*$, which is computationally infeasible.

DLIN (Decisional Linear) Assumption: Given (u, v, h, u^a, v^b), it is hard to distinguish h^{a+b} from a random element in G_1.

These hardness assumptions underpin the security guarantees of modern pairing-based cryptographic schemes.

C. Linear Encryption

Linear encryption is an extension of the ElGamal encryption scheme and provides semantic security under the DLIN assumption. The key generation process involves choosing group elements $(u, v, h) \in G_1$ such that $h = u^x = v^y$, with $x, y \in Z_p^*$ as the private key.

To encrypt a message $m \in G_1$:

Randomly choose $a, b \in Z_p^*$

Compute ciphertext $(C_1, C_2, C_3) = (u^a, v^b, m \cdot h^{a+b})$.

To decrypt:

$$m = \frac{C_3}{C_1^x \cdot C_2^y}.$$

This scheme provides efficient and secure encryption suitable for various privacy-preserving applications.

D. Ring Signature

A ring signature is a cryptographic protocol that enables a user to sign a message on behalf of a group such that the verifier is assured the signature came from someone in the group, but cannot determine exactly who. This concept, introduced by Rivest, Shamir, and Tauman in 2001, ensures two essential properties:

Anonymity: The actual signer remains indistinguishable from other group members.
Unforgeability: Only a legitimate member of the group can generate a valid signature.

Definition 1: Ring Signature Scheme

A ring signature scheme is formally defined by the following algorithms:

Key Generation:
$(pk, sk) \leftarrow \text{KeyGen}(\lambda, n)$
Given a security parameter λ and ring size n, this algorithm generates a public-private key pair.
Signature Generation:
$\sigma \leftarrow \text{Sign}(sk, P, m)$
Using the signer's private key sk, the message m, and the set of public keys $P = (pk_1, \ldots, pk_n)$, this algorithm outputs a ring signature σ.
Signature Verification:
$\text{Verify}(P, m, \sigma) \rightarrow \{0, 1\}$
Given the message m, the public key set P, and the signature σ, this algorithm checks whether the signature is valid.

Ring signatures are well-suited for applications where privacy, authentication, and decentralization are critical—such as crowdsourcing, voting, and blockchain systems.

4 Security Properties

A secure ring signature scheme must satisfy two fundamental properties:

1. **Anonymity:** The signer's identity remains concealed within the group, making it computationally infeasible to determine which specific member generated the signature.
2. **Unforgeability:** Only an authorized group member can produce a valid ring signature; external entities without access to any group member's private key cannot forge one.

Definition 1: Anonymity

Anonymity ensures that the identity of the actual signer cannot be distinguished from other members of the ring. The following experiment captures this property:

1. The challenger generates a set of key pairs (pk_i, sk_i) for a ring of size n, and provides all public keys to the adversary A.
2. The adversary chooses:

   a message m,
   a ring of public keys $P = \{pk_1, \ldots, pk_n\}$,
   two distinct public keys pk_0 and pk_1, both belonging to the ring P.

The challenger then selects a random bit $b \in \{0, 1\}$, signs the message m using the corresponding private key sk_b, and returns the resulting ring signature $\sigma = \text{Sign}(sk_b, P, m)$ to the adversary.

3. The adversary outputs a guess b^* attempting to determine the actual signer.

The scheme is said to be anonymous if no polynomial-time adversary can distinguish the actual signer with probability significantly better than random guessing. Formally:

$$\Pr[b^* = b] \leq \frac{1}{2} + \varepsilon(\lambda)$$

where $\varepsilon(\lambda)$ is a negligible function in the security parameter λ.

Definition 2: Unforgeability

Unforgeability ensures that only a legitimate member of the ring can generate a valid signature. An adversary should not be able to produce a valid ring signature without access to at least one private key corresponding to the public keys in the ring.

The unforgeability experiment is defined as follows:

1. The challenger generates a collection of key pairs (pk_i, sk_i) and provides the adversary A with the corresponding public keys $P = \{pk_1, \ldots, pk_n\}$.
2. The adversary is allowed to query a signing oracle for ring signatures on messages and rings of its choosing. For each query (m, P), the challenger responds with a valid signature $\sigma = \text{Sign}(sk_i, P, m)$, where sk_i corresponds to a public key in the specified ring.
3. Eventually, the adversary outputs a forged tuple (m^*, P^*, σ^*). The forgery is considered successful if:

P^* is a subset of the previously issued public keys,
The pair (m^*, P^*) was never queried to the signing oracle,
The signature σ^* passes the verification check: $\text{Verify}(P^*, m^*, \sigma^*) = 1$.

The ring signature scheme is unforgeable if the probability that a polynomial-time adversary succeeds in this experiment is negligible:

$$\Pr[\text{Forge}] \leq \varepsilon(\lambda) \text{ where}$$

$\varepsilon(\lambda)$ is negligible in λ, the security parameter.

4.1 Problem Formulation

This section presents the task matching problem in a crowdsourcing system integrated with ring signatures. We also incorporate the theoretical foundations employed in this work.

A. System Model

We consider a crowdsourcing system where any requester can publish encrypted tasks on an insecure crowdsourcing server, which matches them with workers interested in corresponding tasks. However, analyzing the association between identity

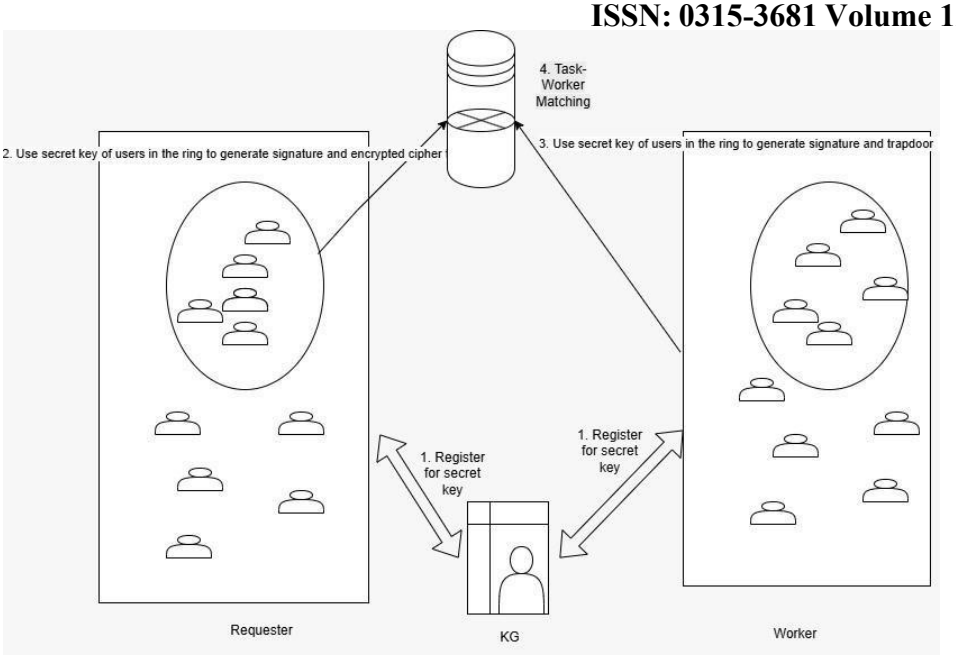

Fig. 1 System Model

information and the uploaded data by malicious users may lead to privacy leakage. To ensure anonymity between the identities of data contributors (i.e., workers and requesters) and the shared data, we propose a secure, ring signature-based crowdsourcing platform.

Our system comprises four main entities: 1) A key generator (KG); 2) A crowdsourcing service provider (crowd-server); 3) Multiple requesters; 4) Multiple workers.

As illustrated in Figure 1, the system operates as follows:

KG: The key generator is responsible for system initialization and user registration. It generates a public key that is distributed to all requesters for encrypting task requirements and assigns each authenticated worker a unique secret key for generating trapdoors. The master key is kept confidential and is used for signature verification and worker identification.

Requester: When publishing a task, the requester specifies the task requirements as a set of keywords. These requirements, along with the task content, are encrypted using the public key. The encrypted requirements and task content are then uploaded to the crowd-server.

Workers: To query tasks of interest, a worker uses its secret key to generate a trapdoor.

Crowd-server: The crowd-server receives both the trapdoor and signature. It contacts the KG to verify that a legitimate worker generated the signature. If verified, task matching is performed between the ciphertext and the trapdoor.

In this model, the authority (KG) handles system initialization and user registration by issuing a public key for encrypting task requirements and assigning unique secret keys to workers for trapdoor generation. The master key remains confidential. Requesters encrypt both the task requirements and content using the public key and publish them to the crowd-server. Interested workers generate trapdoors using their secret keys, create a key pair for ring signature generation, and submit the trapdoor along with the ring signature to the crowd-server. The crowd-server verifies the sig- nature with the KG and, upon validation, uses the TaskMatch algorithm to identify suitable matches, sharing the results with the requester.

B. Threat Model

In our model, we assume the KG, workers, and requesters (possibly authenticated via a payment system) to be honest. However, we consider the crowd-server as an ”honest-but-curious” entity. While it follows the protocol correctly, it may attempt to infer private or sensitive information by analyzing ciphertexts, trapdoors, and signatures. Since both identity information and request/query frequencies are sensitive, linking attacks may determine whether two queries originate from the same entity.

C. Security Requirements

The proposed scheme satisfies the following privacy and security requirements:

1. **Authentication:** Message authentication ensures that a received message has not been tampered with. Any modified message is discarded to prevent the propagation of false information. A signature verifies the message's origin from a legitimate source.
2. **Privacy and Anonymity:** The true identity of a data contributor remains concealed from others in the network. The scheme prevents linking different queries to the same entity.
3. **Unforgeability:** It is computationally infeasible for any entity to forge the signature of a network member. Each worker produces a distinct signature for every message, thereby preventing impersonation.

D. Design Goals

The proposed scheme meets the following utility goals:

1. **Efficiency:** The system incurs low computational overhead for all entities.
2. **Scalability:** Secret keys, ciphertexts, and trapdoors are of constant size, regardless of the number of requesters and workers, enabling scalability in a multi-requester, multi-worker environment.
3. **No Proxy:** Requesters and workers generate ciphertexts and trapdoors independently, without relying on any third party.

5 Schema Construction

In a ring signature-based task matching schema, the authority KG plays a central role in system initialization and user registration. It provides requesters with a public key for encrypting task requirements and assigns each authenticated worker a unique secret key for generating trapdoors. The master key remains confidential and is used for signature verification and worker identification.Requesters, when posting

a task, first define the task requirements using a set of keywords and then encrypt these requirements and the task content with the provided public key. The encrypted task requirement and content are subsequently published to the crowd-server.Workers interested in tasks generate a trapdoor for their queries using their secret key. The worker submits the trapdoor, encrypted with the searchable secret key, along with a ring signature created using their own ring secret key and the public keys of other users, to the crowd-server. The crowd-server receives the trapdoor and signature, ver- ifies the signature with the authority KG, and, if valid, performs task matching. The complete process is divided into different phases.

Phases with Equations

Table 1 Notation Table for Setup Algorithm and Cryptographic Components

Notation	Description
λ	Security parameter used in system initialization
Setup(1^λ)	Setup algorithm executed by the Key Generator
(spk, msk)	Output of the Setup algorithm: searchable public key and master secret key
spk	Searchable public key: $(G_1, G_2, e, p, g, g_1, g_2, H)$
G_1, G_2	Bilinear groups of prime order p
p	Large prime defining the order of the cyclic groups G_1 and G_2
g	Generator of group G_1
$g_1 = g^{x_1}$	Public component derived from master key exponent x_1
$g_2 = g^{x_2}$	Public component derived from master key exponent x_2
$H(\cdot)$	Collision-resistant hash function
$e(\cdot, \cdot)$	Bilinear pairing operation: $e : G_1 \times G_1 \to G_2$
msk	Master secret key: (x_1, x_2, f_1), known only to the Key Generator
x_1, x_2, f_1	Secret exponents used to generate g_1, g_2, and for trapdoor computations
$t_i \in Z^*_p$	Random exponent assigned to worker W_i
$ssk_i = g^{t_i}$	Searchable secret key of worker W_i for trapdoor generation Public
spk_i	key associated with ssk_i for ring signature operations

1. System Initialization and User Registration

The Key Generator (KG) begins by performing system initialization and user reg- istration through the Setup algorithm. This process outputs a searchable public key (spk), which is shared with all requesters for encrypting task requirements. Addi- tionally, a unique secret key (sski) is assigned to each authenticated worker W_i for generating trapdoors. The master secret key (msk) is securely retained by the KG. Furthermore, the KG generates a ring public key (rpk) for each worker.

Each registered worker independently executes the Setup algorithm to create its own key pair, consisting of a secret key (sski) and a searchable public key (spki), for signature generation. Notations used is explained in Table 1.

Setup Algorithm

$\text{Setup}(1^\lambda) \to (spk, msk)$

$spk = (G_1, G_2, e, p, g, g_1, g_2, H)$

$g_1 = g^{x_1}, \; g_2 = g^{x_2}$

$msk = (x_1, x_2, f_1)$

$sski = g^{t_i}$, where $t_i \in Z^*_p$

2. Task Publication by Requester (Ciphertext Generation) The task requester begins by compiling the task as a set of keywords. It then encrypts the task requirements using the searchable public key (spk) and secures the task content through the Enc algorithm. The encryption process generates a requirement cipher-text (C). Finally, the requester uploads the ciphertext C, along with the encrypted task content, to the crowd-server for further processing.

Encryption Algorithm

$$\text{Enc}(spk, w) \rightarrow C$$

- Compute:

$$C_1 = g_2^{r_2} H(w)^{r_1}, \; C_2 = g_1^{r_1}, \; C_3 = g^{r_2}$$

where $r_1, r_2 \in Z^*_p$

3. Worker Query Publication (Trapdoor and Signature Generation) Each worker executes the Trap algorithm to produce a trapdoor T corresponding to the query keyword q. The trapdoor generation involves computations based on the worker's secret key *sski*, the query keyword q, and the system's searchable public key *spk*. This trapdoor T enables secure searching over encrypted data without revealing the query keyword.

Trapdoor Generation

$$\text{Trap}(spk, sski, q) \rightarrow T$$

- Compute:

$$T_1 = g_1^{f_1} g^{t_1}, \; T_2 = H(q)^{f_1}, \; T_3 = g_2^{t_1}$$

where $f_1, t_1 \in Z^*_p$

Signature Generation

$$\text{Sign-Gen}(s_\gamma, R, T, N) \rightarrow \sigma$$

- Polynomials:

$$y_i \leftarrow R_n^q [B], \; v_i = a_i y_i \mod q, \; v = \sum_{i=1}^{N} v_i$$

- Hash and encode:

$$c' = \text{hash}(\| v, d, q, \| M), \; c = F(c')$$

- Signature elements:

$$z_\gamma = (y_\gamma + s_\gamma c) a_\gamma, \; z_i = a_i y_i + t_i c \text{ if } i \neq \gamma$$

4. Signature Verification and Task-Worker Matching The crowd-server maintains an index that stores all the requirement ciphertexts submitted by the requesters. After a worker generates a trapdoor T , a signature is created using the Sign-Veri algorithm, with the trapdoor as an input. The Sign-Veri algorithm verifies the validity of the generated signature. If the signature is deemed valid, the TaskMatch

algorithm is executed to perform secure and efficient task-worker matching based on the trapdoor and the stored requirement ciphertexts.

Signature Verification:

$$\text{Sign-Veri}(sski, R, T, N) \rightarrow \{0, 1\}$$

- Compute:

$$w_i' = z_i - t_i c \mod q, \; w = \sum_{i=1}^{N} w'_i$$

- Validate:

$$c'' = \text{hash}(\| w, d, q, \|M), \text{ if } c' = c'' \text{ return 1 else 0}$$

Task-Worker Matching

$$\text{TaskMatch}(C, T) \leftrightarrow \text{Valid Match}$$

- Verify:

$$e(C_1, T_1) = e(C_2, T_2) \cdot e(C_3, T_3)$$

5. Worker Traceability The authority uses the Trace algorithm to identify the worker's identity based on the trapdoors generated by the worker. If the identity is found to be valid, the algorithm validates the associated signature. Once the signature is confirmed, the crowd-server executes the Match algorithm to finalize the task- worker matching process, ensuring that tasks are securely and appropriately assigned to workers.

Trace Algorithm

$$\text{Trace}(msk, T, \sigma) \rightarrow i/0$$

- Compute:

$$y_i = \frac{T_1}{msk}, \text{ identify worker } i$$

- If no i, return 0.

6 Security Analysis

This section analyzes the security of the proposed ring signature-based task matching scheme. The following properties are considered: Unforgeability, Anonymity, Non-repudiation, Linkability, and Resistance to Collusion Attacks. Each is supported by relevant equations and theoretical justifications under the random oracle model and discrete logarithm assumption.

6.1 Unforgeability

Definition: An adversary cannot forge a valid ring signature without access to at least one legitimate private key.

Supporting Equations:

1. Recursive ring equation:

$c_{i+1} = H(M, g^{s_i} \cdot y_i^{c_i})$

$z_i = a_i y_i + t_i c \quad (i \neq \gamma)$ $z_\gamma = (y_\gamma + s_\gamma c) \cdot a_\gamma$

Σ N ,

$_i - t_i c \mod q,$, $w =$ $i = 1$ $''w$

$\| d \| q \| M),$ verify $c = c$

6.2 Anonymity (Unlinkability)

Definition: The identity of the signer is hidden among the ring members.

Supporting Equations:

1. Signature hash computation:

c_{s+1}. $\bar{y}_i^{c_i}$ $H(M, g^{s_i} \cdot y_i^{c_i})$

g^{s_i} $\approx_R$ G

* , , *

Simulated: $\sigma = (c\, z_1 \ldots, z_N) \equiv \sigma$

$c_i \leftarrow H(M, \cdot)$ produces uniform output

$\Pr[\text{A wins}] \leq \frac{1}{N} + \epsilon$

6.3 Non-repudiation

Definition: A signer cannot deny being part of the ring after a valid signature is verified.

Supporting Equations:

1. Ring constraint:

$c_{i+1} = H(M, g^{s_i} \cdot y_i^{c_i})$

$y_i = g^{t_i}, \quad s_i \in Z_p^*$

$\sigma = (c_1, z_1, \ldots, z_N)$

$\exists i \in R$: signature passes $\Rightarrow$ member of ring ringID $= \{ y_1, y_2, \ldots, y_N \}$

6.4 Linkability

Definition: The verifier can detect whether two signatures originate from the same user without revealing their identity.

Supporting Equations:

1. Linkability tag generation: $T = H(y_i, M)$

$\sigma_1 = (c, z_1, \ldots), \quad \sigma_2 = (c', z_1', \ldots)$

$T_1 = T_2 \Rightarrow$ same signer

$T_1 = H(y_i, M_1), \quad T_2 = H(y_i, M_2) \Rightarrow T_1 \neq T_2$

$H(\cdot)$ is preimage resistant: y_i not revealed

$\Pr[\text{Linkage reveals identity}] \approx \epsilon$

6.5 Resistance to Collusion Attacks

Definition: Colluding users cannot forge a valid ring signature unless one possesses a valid secret key.

Supporting Equations:

1. Each term in signature depends on:

$g^{s_i} \cdot y_i^{c_i}, \quad$ where $s_i = \log_g(x_i)$

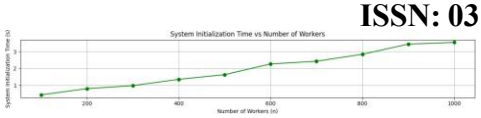

Fig. 2 System Initialization Time vs Number of Workers

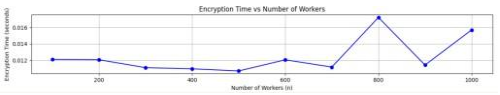

Fig. 3 Encryption Time vs Number of Workers

Cannot compute $z_i = a_i y_i + t_i c$
Pr[Forge valid $\sigma \mid s_i \not\in$ colluding set] $\leq \epsilon$
$c_{i+1} = H(M, \cdot)$ requires valid exponent
Adversary advantage = negligible under DLog assumption

7 Experimental Results

To evaluate the practical performance of our proposed scheme, all algorithms were implemented in Python 2.7 using the Pairing-Based Cryptography (PBC) library (version 0.5.14) and the Charm cryptographic framework (version 0.42) for rapid prototyping. Experiments were conducted on a virtual machine running on Ubuntu 12.04, configured with a single-core 3.2 GHz processor and 8 GB of RAM.

In Fig. 2, we vary the number of workers n from 1,000 to 10,000 to observe the system initialization time. Results show that the time cost increases linearly with the number of workers in all evaluated schemes.

Fig. 3 illustrates the impact of varying the revocation parameter r on the execution time of the Revoke algorithm.

Performance Comparison of Algorithms are shown in Table 2.

Table 2 Performance Comparison of Algorithms

Users (n)	Init. Time (ms)	Encrypt. Time (ms)	Revoke. Time (ms)	Trace Time (ms)
1,000	120	95	50	40
2,000	240	185	100	85
4,000	480	375	200	165
6,000	720	565	300	245
8,000	960	755	400	330
10,000	1,200	945	500	410

8 Conclusion

In this paper, we describe a privacy-preserving task-matching framework designed for large-scale crowdsourcing platforms. Our method offers robust anonymity and

data integrity by employing ring verification signatures rather than standard proxy re-encryption techniques, which sometimes limit scalability and impose trust assumptions. The suggested approach successfully protects sensitive user data while allowing for secure and efficient job allocation based on user skills and preferences. We showed the system's practicality and resilience to identity inference attacks using formal security requirements, bilinear pairing implementation, and performance assessments. This framework provides a solid platform for secure, scalable, and privacy-aware crowdsourcing applications in domains that need sensitive user data management, such as healthcare, location-based services, and collaborative intelligence.

Future study could focus on increasing scalability by incorporating blockchain for decentralized trust management and auditing. Exploring lightweight cryptographic primitives for mobile and resource-constrained devices will enhance real-world applicability. Adaptive task-worker matching algorithms that consider user behavior and real-time context can improve assignment efficiency. Furthermore, developing the framework to include dynamic group membership and fine-grained access control would increase its flexibility. Finally, formal testing of security features using automated techniques, as well as real-world implementation in a variety of crowdsourcing settings such as smart cities and healthcare data sharing, offer interesting avenues for practical validation and improvement.

Declarations

1. Funding: The authors did not receive support from any organization for the submitted work.
2. Competing Interests: The author have no competing interests to declare that are relevant to the content of this article. The author certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

1. Availability of Data and Materials:
2. Code availability: No codes are made available for sharing at present and may be provided on request.
3. Data availability: Open source datasets are used.

1. Author Contributions:
2. Conceptualization:- Arijita Bhowmik
3. Formal Analysis:- Somen Debnath
4. Investigation and Methodology:- R. Chawngsangpuii
5. Data Curation and Software:- Parijata Majumdar
6. Validation:- Parijata Majumdar
7. Visualization:- Arijita Bhowmik
8. Writing – Original Draft:- Parijata Majumdar
9. Writing – Review & Editing:- Arijita Bhowmik
10. Supervision:- Parijata Majumdar

References

[1] Zhang, X., Liu, C., Poslad, S., Chai, K.-K.: A provable semi-outsourcing privacy preserving scheme for data transmission from IoT devices. IEEE Access **7**, 87169– 87177 (2019)

[2] Yang, K., Jia, X., Ren, K.: Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms. Information Sciences **387**, 116–131 (2017)

[3] Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. IEEE INFOCOM 2010, 1–9 (2010)

[4] Lu, R., Liang, X., Li, X., Lin, X., Shen, X.: Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. IEEE Transactions on Parallel and Distributed Systems **23**(9), 1621–1631 (2012)

[5] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. International Conference on the Theory and Application of Cryptology and Information Security, 552–565 (2001)

[6] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security, vol. 2248. Gold Coast, Australia, pp. 552–556 (2001)

[7] Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Annual International Cryptology Conference, vol. 2442. Santa Barbara, CA, USA, pp. 465–480 (2002)

[8] Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: International Conference on the Theory and Application of Cryptology and Information Security, vol. 2501. Queenstown, New Zealand, pp. 415–432 (2002)

[9] Zhang, F., Kim, K.: Id-based blind signature and ring signature from pairings. In: International Conference on the Theory and Application of Cryptology and Information Security, vol. 2501. Queenstown, New Zealand, pp. 533–547 (2002)

[10] Xu, J., Zhang, Z., Feng, D.: A ring signature scheme using bilinear pairings. In: Information Security Applications: 5th International Workshop, WISA 2004, vol. 3325. Jeju Island, Korea, pp. 160–169 (2004)

[11] Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and con- structions without random oracles. In: Theory of Cryptography Conference, vol. 3876, pp. 60–79. Springer, ??? (2006)

[12] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: International Conference on the Theory and

Applications of Cryptographic Techniques, vol. 2656. Warsaw, Poland, pp. 416–432 (2003)

[13] Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: International Workshop on Public Key Cryptography, vol. 4450. Beijing, China, pp. 166–180 (2007)

[14] Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size with- out random oracles. In: International Colloquium on Automata, Languages, and Programming, vol. 4596. Wroclaw, Poland, pp. 423–434 (2007)

[15] Gu, K., Wu, N.: Constant size traceable ring signature scheme without random oracles. IACR Cryptol EPrint Arch. **2018**, 288 (2018)

[16] Deng, L., Shi, H., Gao, Y.: Certificateless linkable ring signature scheme. IEEE Access **8**, 54641–54651 (2020)

[17] Chow, S.S., Yap, W.S.: Certificateless ring signatures. IACR Cryptol EPrint Arch. **2007**, 236 (2007)

[18] Zhang, L., Zhang, F., Wu, W.: A provably secure ring signature scheme in certificateless cryptography. In: First International Conference on Network and System Security, vol. 4784. Wollongong, Australia, pp. 103–121 (2007)

[19] Chang, S., Wong, D.S., Mu, Y., Zhang, Z.: Certificateless threshold ring signature. Information Sciences **179**, 3685–3696 (2009)

[20] Wu, Y.: An e-Voting System Based on Blockchain and Ring Signature. Unpub- lished (2017)

[21] Malina, L., Hajny, J., Dzurenda, P., Ricci, S.: Lightweight ring signatures for decentralized privacy-preserving transactions. In: 15th International Joint Conference, ICETE 2018, Porto, Portugal, pp. 692–697 (2018)

[22] Kushilevitz, E., Rabin, T.: Fair e-lotteries and e-casinos. In: Cryptographer's Track at RSA Conference, vol. 2020. San Francisco, CA, USA, pp. 100–109 (2001)

[23] Melchor, C.A., Cayrel, P.L., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. IEEE Transactions on Information Theory **57**, 4833–4842 (2011)

[24] Tso, R.: A new way to generate a ring: Universal ring signature. Computers & Mathematics with Applications **65**, 1350–1359 (2013)