Dragoon: Private Decentralized HITs Made Practical

Yuan Lu, Qiang Tang, Guiling Wang
Department of Computer Science, New Jersey Institute of Technology
Email: \{yl768, qiang, gwang\}@njit.edu

Abstract—With the rapid popularity of blockchain, decentralized human intelligence tasks (HITs) are proposed to crowdsource human knowledge without relying on vulnerable third-party platforms. However, the inherent limits of blockchain cause decentralized HITs to face a few “new” challenges. For example, the confidentiality of solicited data turns out to be the sine qua non, though it was an arguably dispensable property in the centralized setting. To ensure the “new” requirement of data privacy, existing decentralized HITs use generic zero-knowledge proof frameworks (e.g., SNARK), but scarcely perform well in practice, due to the inherently expensive cost of generality.

We present a practical decentralized protocol for HITs, which also achieves the fairness between requesters and workers. At the core of our contributions, we avoid the powerful yet highly-costly generic zk-proof tools and propose a special-purpose scheme to prove the quality of encrypted data. By various non-trivial statement reformations, proving the quality of encrypted data is reduced to efficient verifiable encryption, thus making decentralized HITs practical. Along the way, we rigorously define the ideal functionality of decentralized HITs and then prove the security due to the ideal/real paradigm.

We further instantiate our protocol to implement a system called Dragoon\(^1\), an instance of which is deployed atop Ethereum to facilitate an image annotation task used by ImageNet. Our evaluations demonstrate its practicality: the on-chain handling cost of Dragoon is even less than the handling fee of Amazon’s Mechanical Turk for the same ImageNet HIT.

I. INTRODUCTION

Crowdsourcing empowers open collaborations over the Internet. A remarkable case is to gather knowledge by human intelligence tasks (HITs). In a HIT, a requester specifies a few questions and lets some workers answer, such that the requester solicits answers while the workers get paid. Since HITs were firstly minted in Amazon’s MTurk\(^2\), they have been widely adopted, say to solicit training datasets for machine learning\(^2\)\(^-\)\(^4\). Notably, ImageNet\(^5\), an impactful deep learning benchmark, was created by thousands of HITs, and laid stepping stones for the deep learning paradigm.

Nevertheless, both academia and industry\(^6\)\(^-\)\(^13\) realize the broader adoption of HITs is severely impeded in practice, as a result of the serious security concerns of free-riding and false-reporting: (i) on the one hand, HITs suffer from low-quality answers, as misconducting workers or even bots would try to reap rewards without making real efforts\(^7\)\(^-\)\(^8\); (ii) on the other hand, many real-world practices set forth the idea of allowing the requester to reject low-quality answers\(^9\)\(^-\)\(^11\), but cause quite many requesters in the wild arbitrarily reject answers in order to collect data without paying\(^14\).

The issues of free-riding and false-reporting become the major obstacles to achieving broader adoption of HITs that are joined by mutually distrustful users\(^14\), and therefore raise a basic requirement of fairness in HITs, namely, the requester pays a worker, iff the worker puts forth a qualified answer. Many studies\(^6\)\(^-\)\(^12\) characterize the purpose and then design proper incentives and payment policies for the needed fairness.

Notwithstanding, most traditional solutions to fairness\(^6\)\(^-\)\(^12\) fully trust in a de facto centralized third-party platform to enforce the payment policies for the basic fairness requirement in HITs. Unfortunately, putting trust in a single party turns out to be vulnerable and elusive\(^2\)\(^-\)\(^13\) in practice, as a reflection of tremendous compromises, outages and misfeasance of real-world crowdsourcing platforms\(^14\)\(^-\)\(^16\). For instance, one of the most popular crowdsourcing platform, MTurk, is biased and allows corrupted requesters to reap data without paying\(^14\)\(^-\)\(^15\). Worse still, all well-known weaknesses of overtrusted third-parties, such as single-point failure\(^16\) and tremendous privacy leakage\(^17\) remain as serious vulnerabilities in the special case of crowdsourcing. Let alone the third-party platforms impose expensive handling fees, say MTurk charges a handling fee up to 45% of overall incentives\(^18\).

New challenges in decentralization. Recognizing those drawbacks of centralized crowdsourcing, recent attempts\(^19\)\(^-\)\(^20\) initiated the decentralized crowdsourcing through the newly emerged blockchain\(^7\) technology. Their aim is to “simulate” a virtual platform that is trustful to enforce the carefully designed payment policies, without suffering from the vulnerabilities of fully centralized systems.

However, as shown in\(^21\)\(^-\)\(^22\), decentralization atop open blockchain brought about a few “new” security challenges that can render the incentives of HITs completely ineffective\(^19\).

Privacy as a basic requirement. In particular, due to the transparency of blockchain\(^21\)\(^-\)\(^22\), once some answers are submitted, any malicious worker can simply copy and resubmit them to earn rewards without making any real efforts, which immediately allows free-riding and cracks the basic fairness of HITs. Namely, the transparent blockchain provides all workers a new choice: running a simple automated script to “copy-and-paste” other answers in the blockchain, which was infeasible in previous centralized systems. More seriously,

\(^1\)In German history, a dragoon or dragoner was a lancer that particularly was light and armed; as an analog, our HITs protocol is super efficient and provably secure regarding the modern freelancers.

\(^2\)Remark that we let the blockchain to refer the permissionless blockchain (e.g., Ethereum mainnet) that is open to any Internet node through the paper.
having the new path to free-riding in mind, rational workers would wait to copy, instead of doing any real efforts. Thus sorts “tragedy of the commons” occurs, and no one will respond with independent answers \[23\]–\[26\]. That said, the straightforwardly decentralized crowdsourcing arguably loses all basic utilities and fails to gather anything meaningful!

So the privacy becomes indispensable in the decentralized crowdsourcing systems, instead of a bonus property.

**State-of-the-art & open problem.** To overcome blockchain’s inherent limits, prior art \[19\] proposes the general outsource-then-prove framework for private decentralized HITs. It enables the requester to prove the quality of answers that are encrypted to her, without revealing the actual answers. Such the proof becomes the crux to ensure privacy, and deters both false-reporting and free-riding. Now, the blockchain needs to verify proofs, so a feasibility challenge sprouts up, considering the on-chain computational resources are too limited to support any large proof or costly verification.

For above reasons, prior work relies on some generic zero-knowledge proof (zk-proof) framework that is succinct in proof size and efficient for verifying, in particular SNARK\[29\]–\[31\] to reduce the on-chain verification cost.

Nonetheless, generic zk-proofs such as SNARK inevitably inherit low performance for the convenience of achieving generality, causing that prior private decentralized HITs suffer from an unbearable off-chain proving cost and a still significant on-chain verifying expense:

- **Infeasible proving (off-chain).** The proving of generic zk-proofs (e.g., SNARK) seems inherently complex, due to the burdensome NP-reduction for generality. In particular, prior study \[32\] reported 56 GB memory and 2 hours are needed to prove whether an encrypted answer coincides with the majority of all encrypted submissions at a very small scale, e.g., at most eleven answers. Such a performance prevents the previous protocol from being usable by any normal requesters using regular PCs.

- **Costly verification (on-chain).** Existing blockchains (e.g. Ethereum) are feasible to verify only few types of generic zk-proofs such as SNARK, whose verification need to compute a dozen of expensive pairings over elliptic curve \[29\],\[31\]. So the on-chain verification of these zk-proofs is not only computationally costly, but also financially expensive. Currently in Ethereum, 12 pairings already spend \(\sim\)500k gas \[33\], and verifying a SNARK proof costs even more (about half US dollar).

Given the insufficiencies of the state-of-the-art, the following critical problem remains open:

*How to design a practical private decentralized HITs protocol for crowdsourcing human knowledge?*

\[3\]Remark that though the rise of Intel SGX becomes a seemingly enticing alternative of SNARK to go beyond many limits of blockchain by remote attestations \[24\], unfortunately, recent Foreshadow attacks \[25\] allow the adversary to forge “attestations” by stealing the attestation key hardcoded in any SGX Enclave, which seriously challenges the already heavy assumption of “trusted” hardware, and makes it even more illusive to trust SGX in practice.

**Our contributions.** To answer the above unresolved problem, we present a practical private decentralized HITs protocol for the major tasks of crowdsourcing human knowledge. In sum, our core technical contributions are three-fold:

- To achieve the practical protocol for private decentralized HITs, we carefully explore various non-trivial optimizations to avoid the cumbersome generic-purpose zero-knowledge framework, and reduce the protocol to a special-purpose verifiable encryption. As such, we attain concrete improvements by orders of magnitude, regarding both the proving and verification:
  - For proving, our approach is *two order of magnitude* better than generic zk-proof\[17\] In particular, for the same HIT, the proving cost of our protocol is only 50 MB memory and 10 ms running time.
  - For verifying, our result improves upon the generic solution by *nearly an order of magnitude*. The on-chain cost of verifying a proof for answer quality is reduced to \(\sim\)180k gas in Ethereum (much smaller than verifying SNARK proofs and typically few US cents).

- We further implement our protocol to instantiate a practical private decentralized crowdsourcing system Dragoon, which can be deployed atop many real-world blockchains such as Ethereum.

We use Dragoon to launch a concrete HIT adopted by ImageNet \[12\] to solicit large-scale image annotations atop Ethereum. To handle the task, Dragoon attains an on-chain (handling) cost \(\sim\)$2 US dollars at the time of writing. In comparison, for the same task, the handling fee of MTurk is at least $4 currently \[18\]–\[34\].

Our result provides an insight that the on-chain handling fee (characterizing the users’ financial expense) in the decentralized setting can approximate or even less than the handling fee charged by centralized platforms. This hints the de facto users can financially benefit from decentralized crowdsourcing, though it is not contradictory to the common belief \[35\] that decentralization is more expensive regarding the overall system’s cost.

- Along the way, we firstly present the ideal functionality of decentralized HIT. This rigorous security model clearly defines what a secure HIT shall be, and allows us prove security against subtle adversaries in the blockchain, due to simulation-based paradigm.

In contrast, existing decentralized HITs \[19\]–\[32\] have quite different property-based definitions on “securities”, which at least causes the lack of well-defined benchmark to compare. Even worse, many of them are flawed, as fail to capture all respects of subtle adversaries in blockchain; say, they allow a corrupted requester to reap data without paying, if being given the standard ability of adversarially re-ordering message deliveries. Differently, our simulation-based security model precisely defines the security against subtle attacks in the blockchain.

\[4\]Generic zk-proof refers zk-SNARK in our context, since the only generic zk-proof that can be feasibly supported by existing blockchains is zk-SNARK.
Challenges & our techniques. The major challenge of making private decentralized HITs practical is that the blockchain must learn the quality of some encrypted answers, namely, to obtain some properties of what a few ciphertext are encrypting. The state-of-the-art [19, 32] proposed to reduce the problem to generic zk-proofs, by observing the requester can decrypt the answers, and then prove the quality of answers to the blockchain. But such the generic approach causes impractical expenses inherently, because of the underlying heavyweight NP-reduction for generality.

To conquer the above challenge, we conduct a different path that deviates from generic zk-proof frameworks to explore a concretely efficient solution. At the core of our private decentralized HITs protocol, we present a special-purpose non-interactive proof scheme to efficiently attest the quality of encrypted answers. Such the approach gets rid of heavyweight general-purpose zk-proof frameworks and then avoids the inefficiency caused by generality.

The ideas behind our efficient proving scheme are a variety of special-purpose optimizations to squeeze performance by removing needless generality, such that we reduce the problem of proving encrypted answers’ quality from generic-purpose zk-proof to particular verifiable encryption. As shown in Fig 1, our core ideas are highlighted as:

- **Abstracting real-world HITs.** The first step is to well abstract an incentive widely adopted by real-world HITs, namely, the only one incorporated by Amazon’s MTurk [10]. In the incentive, some golden standard challenges (i.e., questions with known answers) [7] are mixed with other questions, so the quality of a worker is due to her performance on the golden standards.

We carefully formulate the problem of proving the quality of encrypted answers for the above concrete incentive. So proving the quality of a worker can be reducible to a well-defined two-party problem, in which the verifier needs to output the performance of the worker on a set of golden standard questions, given only a set of ciphertext answering these golden standards challenges.

Nevertheless, solving this two-party problem is still challenging, as it needs to compute the property of what a set of ciphertext are encrypting. The generic version of the issue falls into multi-input functional encryption [36, 37], which is well known for its hardness, and has no (nearly)

\[ A \]

\[ \text{Concrete Proving} \]

\[ \text{Efficient construction of verifiable encryption} \]

\[ \text{Concrete proving scheme} \]

\[ \text{Reduced to verifiable encryption by removing all arithmetic relations} \]

\[ \text{Concretely efficient proof of quality} \]

\[ \text{Statement Reformation} \]

\[ \text{1. Prove upper-bound} \]

\[ \text{2. Particular zero-knowledge} \]

\[ \text{Abstract real-world HITs} \]

\[ \text{The only quality-based incentive incorporated by Amazon’s MTurk} \]

\[ \text{Generic zk-proofs} \]

\[ \text{A concrete crypto problem between 2-party} \]

\[ \text{Efficient construction of verifiable encryption} \]

\[ \text{Prove upper-bound of each worker’s quality instead} \]

\[ \text{Second, we realize that given the system’s public knowledge, a tiny and constant portion of each worker’s answer (i.e., the part answering gold standards) is already leaked, since this little portion becomes simulatable by the public knowledge; thus we explicitly relax our goal to leak these “already-leaked” information.} \]

\[ \text{So proving the upper bound of each worker’s quality instead of proving the exact number, which is a relaxation in the general cases, but does not scarify any utility in our context where the reward is an increasing function of quality. Second, we realize that given the system’s public knowledge, a tiny and constant portion of each worker’s answer (i.e., the part answering gold standards) is already leaked, since this little portion becomes simulatable by the public knowledge; thus we explicitly relax our goal to leak these “already-leaked” information.} \]

To sum up, the above reformations allows us to remove needless generality of proving answer quality, so that we can reduce the problem to standard verifiable encryption without giving up securities/utilities.

- **Concretely efficient proving scheme.** Following the above optimizations, the problem eventually is reduced to verifiable encryption, which becomes representable in concrete algebraic relations. Along the way, we present a certain variant of verifiable encryption that is concretely tailored for the scenario of HITs where the plaintexts are short, and thus squeeze most performance out of it.

This completes our special-purpose design to boost private decentralized HITs, practically.

II. Other related Work

Besides existing private decentralized HITs [19, 32] discussed earlier, here we briefly review some pertinent generic cryptographic frameworks and discuss their insufficiencies in the concrete context of private decentralized crowdsourcing.

Privacy-preserving blockchain. A variety of studies [22, 38, 39] consider the general framework for privacy-preserving blockchain and smart contract. The approaches are powerful in the sense of their generality, yet are expensive for concrete use-cases in practice. For example, Hawk [22] leverages generic zk-proofs to keep blockchain private, but incurs expensive proving expenses. As such, it is unclear how to leverage these generic frameworks to design concretely efficient protocol for the special-purpose of crowdsourcing [19].

Fair MPC using blockchain. Decentralized crowdsourcing is a special-purpose fair MPC using blockchain. Kiayias, Zhou and Zikas [21] consider the generic version of fair MPC in the presence of blockchain, but it is unclear how to adopt their generic protocol in practice without expensively computational costs. Recently, increasing interests focus on special-purpose variants of fair MPC in aid of blockchain. For example, [40, 42] consider poker games. But these special-purpose solutions
are over-tuned for distinct scenarios and are unclear how to be used for private decentralized crowdsourcing.

**Multi-input functional encryption.** The core problem of private decentralized crowdsourcing is to let the blockchain learn the quality of encrypted answers, which is straightforwardly reducible to multi-input functional encryption (MIFE) \[36\]. But MIFE relies on indistinguishability obfuscation \[36\] or multi-linear maps \[37\], which currently we do not notice how to instantiate under standard cryptographic assumptions.

III. Preliminaries

Here we briefly review some relevant cryptographic notions. Following convention, we let \( \mathcal{S} \) to denote uniformly sampling and \( \approx_c \) to denote computationally indistinguishable.

**Cryptocurrency ledger.** The cryptocurrency maintained atop the blockchain instantiates a global bookkeeping ledger (e.g., denoted by \( \mathcal{L} \)) to deal with “coin” transfers, transparently. It can be called out by an ideal functionality (i.e., a standard model of so-called smart contract \[21, 22\]) as a subroutine to assist conditional payments. Formally, cryptocurrency \( \mathcal{L} \) can be seen as an ideal functionality interacting with a set of parties \( \mathcal{P} = \{P_i\} \) and the adversary; it stores the balance \( b_i \) for each \( P_i \in \mathcal{P} \), and handles the following oracle queries \[22, 43\]:

- **FreezeCoins.** On input \( (\text{freeze}, P_i, b) \) from an ideal functionality \( \mathcal{F} \) (i.e. a smart contract), check whether \( b_i \geq b \) and proceed as follows: if the check holds, let \( b_i = b_i - b \) and \( b_F = b_F + b \), send \( (\text{frozen}, \mathcal{F}, P_i, b) \) to every entity; otherwise, reply with \( (\text{nofund}, P_i, b) \).
- **PayCoins.** On input \( (\text{pay}, P_i, b) \) from an ideal functionality \( \mathcal{F} \) (i.e. a smart contract), check whether \( b_F \geq b \) and proceed as follows: if that is the case, let \( b_i = b_i + b \) and \( b_F = b_F - b \), send \( (\text{paid}, \mathcal{F}, P_i, b) \) to every entity.

**Commitment scheme.** The commitment scheme is a two-phase protocol among a sender and a receiver. In the commit phase, a sender “hides” a string \( \text{msg} \) behind a commitment string \( \text{comm} \) with using a blinding key, namely, the sender transmits \( \text{comm} = \text{Commit} (\text{msg}, \text{key}) \) to the receiver. In the reveal phase, the receiver gets \( \text{key}' \) and \( \text{msg}' \) as opening for \( \text{comm} \), and executes \( \text{Open}(\text{comm}, \text{msg}', \text{key}') \) to output 0 (reject) or 1 (accept). We require computational hiding and computational binding. The former one requires the commitments of any two strings are computationally indistinguishable. The latter one means the receiver would not accept an opening to reveal \( \text{msg}' \neq \text{msg} \), except negligible probability.

**Decisional Diffie-Hellman (DDH).** DDH problem is to tell that \( d = ab \) or \( d \approx \mathcal{S} \) \( \mathbb{Z}_p \), given \( (g, g^a, g^b, g^d) \) where \( a, b \approx \mathcal{S} \) \( \mathbb{Z}_p \) and \( g \) is a generator of a cyclic group \( \mathcal{G} \) of order \( p \). The DDH assumption states \( \{ (g, g^a, g^b, g^d) \} \approx_c \{ (g, g^a, g^b, g^{ab}) \} \). We assume DDH assumption holds along with the paper.

**Verifiable encryption.** We consider the verifiable public key encryption scheme (VPKE) consisting of a tuple of algorithms \((\text{KeyGen, Enc, Dec, ProvePKE, VerifyPKE})\).

In short, \text{KeyGen} can set up a pair of encryption-decryption algorithms \((\text{Enc}_h, \text{Dec}_h)\), where \( h \) and \( k \) are public and private keys respectively. We let any \((\text{Enc}_h, \text{Dec}_h) \leftarrow \text{KeyGen}(1^\lambda) \) is a public key encryption scheme satisfying semantic security. For presentation simplicity, we also let \((\text{Enc}_h, \text{Dec}_h) \) denote the public-secret key pair \((h, k)\). Moreover, for any \((h, k) \leftarrow \text{KeyGen}(1^\lambda) \), the ProvePKE\(_h\) algorithm explicitly inputs the private key \( k \) and the ciphertext \( c \), and outputs a message \( m \) with a proof \( \pi \); the VerifyPKE\(_h\) algorithm explicitly inputs the public key \( h \) and \((m, c, \pi)\), and outputs 1/0 to accept/reject the statement that \( m = \text{Dec}_h(c) \). Beside semantic security, VPKE also satisfies the following extra properties:

- **Completeness.** \( \Pr[\text{VerifyPKE}_h(m, c, \pi) = 1 \mid (m, \pi) \leftarrow \text{ProvePKE}_h(c)] = 1 \), for \( \forall c \) and \((h, k) \leftarrow \text{KeyGen}(1^\lambda) \);
- **Soundness.** Given any \((h, k) \leftarrow \text{KeyGen}(1^\lambda) \) and any ciphertext \( c \), any P.P.T. adversary \( \mathcal{A} \) cannot produce a proof \( \pi \) fooling VerifyPKE\(_h\) to accept that \( c \) is decrypted to \( m' \) if \( m' \neq \text{Dec}_h(c) \), with except negligible probability;
- **Zero-knowledge.** The proof \( \pi \) can be simulated by a P.P.T. simulator on input only public knowledge \( m, h \) and \( c \), which ensures the protocol leaks nothing more than the truthness of the statement \( m = \text{Dec}_h(c) \).

**Random oracle.** We treat the cryptographic hash function as a global and programmable random oracle \[44\], and denote the hash function with \( \mathcal{H} \) through the paper.

**Simulation-based paradigm.** To formalize and prove security, a real world and an ideal world can be defined and compared: (i) in the real world, there is an actual protocol \( \Pi \) among the parties, some of which can be corrupted by an adversary \( \mathcal{A} \); (ii) in the ideal world, an “imaginary” trusted ideal functionality \( \mathcal{F} \) replaces the protocol and interacts with honest parties and a simulator \( \mathcal{S} \). We say that \( \Pi \) securely realizes \( \mathcal{F} \), if for \( \forall \) P.P.T. adversary \( \mathcal{A} \) in the real-world, \( \exists \) a P.P.T. simulator \( \mathcal{S} \) in the ideal-world, s.t. the two worlds cannot be distinguished, which means: no P.P.T. distinguisher \( \mathcal{D} \) can attain non-negligible advantage to distinguish “the joint distribution over the outputs of honest parties and the adversary \( \mathcal{A} \) in the real world” from “the joint distribution over the outputs of honest parties and the simulator \( \mathcal{S} \) in the ideal world”.

Moreover, we consider static adversary who is only allowed to corrupt some parties before the protocol starts. Protocols proven secure in the real/ideal paradigm can be composed sequentially, due to the transitivity of security reductions \[45\].

The advantage of simulation-based paradigm is that all desired behaviors of the protocol can be precisely described by the ideal functionality. Remarkably, the approach has been widely adopted to analyze decentralized protocols \[21,22,40\] to capture subtle adversaries in the decentralized setting.

IV. Formalization of Decentralized Human Intelligent Tasks

This section rigorously defines our security model, by giving the ideal functionality of Human Intelligent Tasks (HITs) that captures the security/utility requirements of the state-of-the-art HITs in reality \[12,15\]. Our security modeling sets forth a clear security goal, that is: the HITs in the real world shall be as “secure” as the HITs in an admissible ideal world.
Reviewing the HITs in reality. Let us briefly review the HITs adopted in reality [2,15], before presenting our abstraction of their ideal functionality.

Parties & process flow. There are two explicit roles in a HIT, i.e., the requester and some workers. The requester, uniquely identified by \( R \), can post a task \( T \) to collect a certain amount of answers. In the task, \( R \) also promises a concrete reward policy. The worker with a unique identifier \( W_j \), submits his answer \( a_j \) to expect the receipt.

Task design. A HIT consists of a sequence of questions denoted by \( T = (q_1, \ldots, q_N) \), where each \( q_i \) is a multiple choice question and \( N \) is the number of questions in the task. The answer of each question must lay in a particular range \( \mathbb{R} \) pre-specified when \( T \) is published.

The above HIT design is based on batched choice questions, which follows real-world practices [2,15] to remove ambiguity, thus letting workers precisely understand the task. For example, Fei-Fei Li et al. [2,12,27] used the technique to create the deep learning benchmark ImageNet, and Andrew Ng et al. [3] suggested it for language annotations.

Answer quality. The quality of an answer is induced by a function \( \text{Quality}(a_j; sp) \), where \( a_j = (a_{1,ij}, \ldots, a_{N,ij}) \) is the answer submitted by worker \( W_j \), and \( sp \) is some secret parameters of requester. The output of \( \text{Quality}(\cdot) \) is denoted by \( \chi_j \), which is said to be the quality of worker \( W_j \).

The above abstraction captures the quality-based incentive mechanism adopted by real-world HITs in Amazon’s MTurk [10-13]. For example, a task \( T \) consists of \( N \) questions, out of which \( M \) questions are golden-standard questions that are “secretly” mixed. The quality of a worker can be computed, due to her accuracy in the \( M \) golden-standard questions.

Formally, in the quality function \( \text{Quality}(a_j; sp) \), the parameter \( sp = (G, G_S) \), where \( G \subseteq [1, N] \) represents the randomly chosen indexes of the golden-standard questions, and \( G_S = \{s_i \mid s_i \in \text{range} \} \) represents the known answers of the golden-standard questions. Following the real-world practices [10-13], the quality of an answer \( a_j = (a_{1,ij}, \ldots, a_{N,ij}) \) is:

\[
\text{Quality}(a_j, (G, G_S)) = \sum_{i \in G} a_{i,j} \in s_i
\]

where \([\cdot] \) is Iverson bracket to convert any logic proposition to 1 if the proposition is true and 0 otherwise.

Defining the decentralized HITs’ functionality. Now we are ready to present our security notion of HITs in the presence of cryptocurrency. We formalize the ideal functionality of HITs (denoted by \( F_{hit}^L \)) in the \( L \)-hybrid model as shown in Fig. 2. Intuitively, \( F_{hit}^L \) abstracts a special-purpose multi-party secure computation, in which: (i) a requester recruits \( K \) workers to crowdsourcing some knowledge, and (ii) each worker gets a payment of \( \mathcal{B}/K \) from the requester, if submitting an answer meeting the minimal quality standard \( \Theta \).

In greater detail, the ideal functionality \( F_{hit}^L \) of HITs immediately implies the following security properties:

- **Fairness.** Our ideal functionality captures a strong notion of fairness, that means: the worker get paid, if and only if s/he puts forth a qualified answer (instead of copying and pasting somewhere else). In greater detail, the requester specifies a sequence of \( N \) multi-choice questions, which are multi-choice questions having some options in range and contain \( |G| \) gold-standard challenges. For each worker, s/he has to (i) meet a pre-specified quality standard \( \Theta \) and (ii) submit answers in the range of options, in order to receive the pre-defined payment \( \mathcal{B}/K \).

![The ideal functionality of HIT](image)
• **Audibility of gold-standards.** The choose of golden standards is up to the requester, so it becomes a realistic worry that a malicious requester uses some bogus as the answers of golden standard questions. The ideal functionality aims to abstract the best prior art \[14, 15\] regarding this issue so far, that means the golden standards become public auditable once the HIT is done. This abstraction “simulates” the ad-hoc reputation systems maintained by the MTurk workers to grade the reputations of the MTurk requesters in reality \[14, 15\].

• **Confidentiality.** It means any worker cannot learn the advantage information during the course of protocol execution. Without the property, workers can copy and paste to free ride, so it is a minimal requirement to ensure the usefulness of decentralized HITs. Our ideal functionality naturally captures the property.

**Adversary.** We consider probabilistic polynomial-time adversary in the real world. It can corrupt the requester and/or some workers statically, before the real-world protocol begins. The uncorrupted parties are said to be honest. Following the standard blockchain model \[21, 22\], we also abstract the ability of the real-world adversary to control the communication (between the blockchain and honest parties) as: (i) it follows the synchrony assumption \[22, 48\], namely, we let there is a global clock \[22, 48\], and the adversary can delay any messages sent to the blockchain up to a-priori known time (w.l.o.g., up to the next clock); (ii) the adversary can manipulate the order of so-far-undelivered messages sent to the blockchain, which is known as the “rushing” adversary.

**Expressivity of the ideal functionality** \(\mathcal{F}_{hit}\). Our ideal functionality of HITs is rather expressive, as it not only captures the elegant state-of-the-art of collecting image/language/video annotations \[2, 4, 11, 13, 47\], but also reflects the common scenario of crowdsourcing human knowledge. Consider the next example: Alice is running a small startup, and aims to prove the quality of encrypted answers. Then we showcase the smart contract functionality \(\mathcal{G}_{hit}\) that interacts with the workers and the requester. Later, the detailed protocol is given in the presence of \(\mathcal{G}_{hit}\). We finally prove that our protocol securely realizes the ideal functionality \(\mathcal{F}_{hit}\) of HITs.

A. Proof of quality of encrypted answer (PoQoEA)

The core building block of our novel decentralized protocol is to allow the requester efficiently prove the quality of encrypted answers. We formally define this concrete purpose to set forth the notion of PoQoEA, and then present an efficient reduction from it to verifiable encryption (VPKE).

**Defining PoQoEA.** The problem we are addressing here is to prove that: an encrypted answer \(c_j\) can be decrypted to obtain some \(a_j\) s.t. the quality of \(a_j\) is \(\chi\), without leaking anything other than \(c_j, \chi\) and the parameters of quality function.

To capture the problem, the state-of-the-art \[19, 22\] adopts the standard notion of zk-proof in order to support generic quality measurements. Different from existing solutions, we particularly tailor the notion of zk-proof to obtain a fine-tuned notion of PoQoEA for the widely adopted quality function defined in \[14\]. Namely, we consider Quality\((\cdot; G, G_s)\) where \(G\) is the index of gold-standards and \(G_s = \{s_i\}_{i \in G}\) is the ground truth of golden standards, and aim to remove the unnecessary generality in the concrete setting.

Precisely, given the quality function Quality\((\cdot; G, G_s)\) and any established public key encryption scheme \((\text{Enc}_h, \text{Dec}_h) \leftarrow \text{KeyGen}(1^\lambda)\), we can define PoQoEA as a tuple of hereunder algorithms (ProveQuality\(_h\), VerifyQuality\(_h\)):

1. **ProveQuality\(_h\)**: \(c_j, \chi, G, G_s \rightarrow \pi\). Given the encrypted answer \(c_j = (c_{1j}, \ldots, c_{nj})\), the quality \(\chi\) and the golden standards \((G, G_s)\), it outputs a proof \(\pi\) attesting \(\chi\) is the quality of \(c_j\); the algorithm explicitly takes the secret decryption key \(k\) as input;

2. **VerifyQuality\(_h\)**: \(c_j, \chi, \pi, G, G_s \rightarrow 0/1\). It outputs 0 (reject) or 1 (accept), according to whether \(\pi\) is a valid proof attesting \(\chi\) is the actual quality of \(c_j\); the algorithm explicitly takes the public encryption key \(h\) as input;

Moreover, PoQoEA shall satisfy the following properties:

- **Completeness.** PoQoEA is complete, if for any \(G, G_s, c_j, \chi\) and \((\text{Enc}_h, \text{Dec}_h)\) s.t. \(\chi \geq \text{Quality}(\text{Dec}_h(c_j); G, G_s)\), there is \(\text{Pr}[\text{VerifyQuality}_h(c_j, \chi, \pi, G, G_s) = 1 | \pi \leftarrow \text{ProveQuality}_h(c_j, \chi, G, G_s)] = 1\);

- **“Upper-bound” soundness.** PoQoEA is upper-bound sound, if for any \(G, G_s, c_j, \chi\) and \((\text{Enc}_h, \text{Dec}_h)\), for all P.P.T. \(A\), there is \(\text{Pr}[\text{VerifyQuality}_h(c_j, \chi, \pi', G, G_s) = 1 | \chi < \text{Quality}(a_j; G, G_s) \wedge a_j = \text{Dec}_h(c_j) | \pi' \leftarrow A(G, G_s, \chi, c_j, \lambda, \text{Enc}(k), \text{Dec}(k)) \leq \text{negl}(\lambda)]\), where negl(\(\lambda\)) is a negligible function in \(\lambda\); so it is computationally infeasible to produce a valid proof, if \(\chi\) is not the upper bound of the quality of what \(c_j\) is encrypting;

- **“Special” zero-knowledge.** Conditioned on \([G]\) and the range of elements in \(G_s\) are small constants, for any \(G, G_s, c_j, \chi\) and \((\text{Dec}_h, \text{Enc}_h)\), \(\exists\) a P.P.T. simulator \(S\) that can simulate the communication scripts of PoQoEA protocol on input only \(h, G, G_s, c_j,\) and \(\chi\).

**Rationale behind the finely-tuned abstraction.** The notion of PoQoEA is defined to remove needless generality in the special case of HITs. Compared to the state-of-the-art notion \[19\], PoQoEA is more promising to be efficiently constructed, as it brings the following definitional advantages:
• We adopt upper-bound soundness to prove the upper bound of quality instead of proving the exact quality of each worker. Such the tuning steps from a basic fact that: the reward of a worker is an increasing function in quality, so the upper bound of the worker’s quality exactly reflects the well-deserved reward of the worker.

• Another major difference is the relaxed special zero-knowledge, which means: PoQoEA is zero-knowledge, when $|G|$ and $s_i \in G_s$ are small, so anything simulatable by the gold standards can be leaked. Nevertheless, the conditions are prevalent in the special context of HITs [2][15], because $G$ represents few golden standard questions, and the range of $s_i$ represents the options of each multiple-choice question in HITs, thus both of which are small in reality.

In sum, even though PoQoEA is seemingly over-tuned, it essentially coincides with the generic zk-proof of the quality of encrypted answers in the context of HITs.

Construction and security analysis. Here is an efficiency-driven way to constructing PoQoEA with the quality function $Quality(a_j; G, G_s)$ defined in [4][4]. We reduce the problem to the standard notion of verifiable encryption. More precisely, if being given (Enc_k, Dec_k, ProvePKE_k, VerifyPKE_k) that is an established verifiable encryption scheme, PoQoEA for (Enc_k, Dec_k) can be constructed as illustrated in Fig 3.

**Lemma 1.** Given any verifiable encryption VPKE, the algorithm in Fig 3 satisfies the definition of PoQoEA regarding the quality function defined in [4][4].

**Proof.** (sketch) The completeness is immediate to see once considering the definition of quality function, the correctness of encryption and the completeness of VPKE. To prove the upper-bound soundness, we can assume by contradiction to let an adversary break it, then the adversary can immediately be leveraged to break the soundness of VPKE, which leads up to contradiction. The special zero-knowledge is also clear to see: considering $|G|$ and the range of $s_i$ are small constants, we can construct a P.T. simulator $S$ that inverts at most polynomial number of VPKE subroutines [49] to obtain VPKE proofs, thus allowing $S$ to internally craft a simulated PoQoEA proof. □

---

**The HITs contract functionality $C^L_{hit}$**

Given accesses to $L$, $C_{hit}$ interacts with $R$, $\{W_j\}$, and $A$.

**Phase 1: Publish Task**

- Upon receiving (publish, $N, S, K$, range, $\Theta, h, comm_{gs}$) from $R$, leak the message and $R$ to $A$, until the beginning of next clock, proceed with the delayed executions down below:
  - send (freeze, $P_i, S_i$) to $L$, if returns (frozen, $F_{hit}, P_i, S_i$):
    * store $N, S, K$, range, $\Theta, h$ and $comm_{gs}$
    * initialize answers $\leftarrow \emptyset$, comms $\leftarrow \emptyset$
    * send (published, $R, N, S, K$, range, $\Theta, h, comm_{gs}$) to all entities, and goto phase 2-a

**Phase 2-a: Collect Answers (Commit phase)**

- Upon receiving (commit, $comm_{gs}$) from $W_j$, leak the message and $W_j$ to $A$, then proceed with the following delayed executions until the beginning of next clock, with consulting $A$ to re-order all received commit messages:
  - for each received commit message (sent from $W_j$):
    * if $(W_j, \cdot)$ is in $comm_{gs}$ and $\cdot, (\cdot, comm_{gs})$ is not in $comm_{gs}$:
      - let comms $\leftarrow$ comms $\cup (W_j, comm_{gs})$
      - if $comm_{gs} = K$, send (committed, comms) to all entities, and goto the reveal phase

**Phase 2-b: Collect Answers (Reveal phase)**

- Upon entering this phase, leak all received messages and their senders to $A$, till the next clock period, proceed as:
  - for each $W_j \in \{W_j | (W_j, \cdot) \in comms\}$:
    * if receiving the message (reveal, $c_j, key_{j}$) from $W_j$ such that $Open(comm_{c_j}, c_j, key_j) = 1$:
      - answers $\leftarrow$ answers $\cup (W_j, c_j)$
    * else answers $\leftarrow$ answers $\cup (W_j, \bot)$
    * send (revealed, answers) to all, and goto the next phase

**Phase 3: Evaluate Answers**

- Upon entering this phase, leak all received messages and their senders to $A$, till the next clock period, proceed as:
  - if receiving (golden, $G, G_s, key_{gs}$) from $R$, such that $Open(comm_{gs}, G||G_s, key_{gs}) = 1$:
    - for each $W_j \in \{W_j | (W_j, \cdot) \in comms\}$:
      - if receiving (outrange, $W_j, i, a_{ij}, \pi_i$) from $R$:
        - send (pay, $W_j, \bot/K$) to $L$, if $a_{ij} \in range$ or $VerifyPKE_k(a_{ij}, c_{ij}, \pi_i) = 0$
        - else if receiving (evaluate, $W_j, \chi_j, \pi$) from $R$:
          - send (pay, $W_j, \bot/K$) to $L$, if $\chi_j \geq \Theta$ or $VerifyQuality_k(c_j, \chi_j, \pi, G, G_s) = 0$
        - else if $c_j \neq \bot$, send (pay, $W_j, \bot/K$) to $L$
      - otherwise, for each $W_j \in \{W_j | (W_j, \cdot) \in answers\}$, send (pay, $W_j, \bot/K$) to $L$

---

**B. HIT contract and HIT protocol**

Now we are ready to present our concretely efficient decentralized protocol $\Pi_{hit}$ for HIT. Our design centers around a smart contract $C^L_{hit}$, which is formally described in Fig 4. The contract $C^L_{hit}$ is the crux to take best advantage of the rather limited abilities of blockchain to make our protocol securely realize the ideal functionality $F^L_{hit}$. Thus given contract $C^L_{hit}$, our HITs protocol $\Pi_{hit}$ can be defined among the requester, the worker and the contract, as formally illustrated in Fig 5.

Informally, our HIT protocol $\Pi_{hit}$ proceeds as follows:
1) **Publish task.** The requester $\mathcal{R}$ announces her public key $h$, and publishes a task $\mathcal{T}$ of $N$ multi-choice questions to crowdsourced $K$ answers for the task. Each question in $\mathcal{T}$ is specified to have some options in range. The task mixes some golden standard questions, whose indexes $G$ and ground truth $G_s$ are committed to $\text{comm}_{gs}$. Also, $\mathcal{R}$ places $\mathcal{B}$ as deposit to cover her budget, which promises that a worker would get a reward of $\mathcal{B}/K$, if submitting an answer beyond a specified quality standard $\Theta$.

2) **Commit answers.** Once the task is published, the workers can commit their answers (encrypted to the requester) in the task. To prevent against copy-and-paste attacks, duplicated commitments are rejected. The contract moves to the next phase, once $K$ distinct workers commit.

3) **Reveal answers.** After $K$ workers commit their answers, these workers can start to reveal their answers in form of ciphertexts encrypted to the requester. Note that the submissions of answers explicitly contain two subphases, namely, committing and revealing, which is the crux to prevent the network adversary from taking advantages by adversarially scheduling the order of submissions.

4) **Evaluate answers.** Eventually, the requester is supposed to instruct the blockchain to correctly pay these encrypted answers to facilitate the critical fairness. To this end, the protocol leverages our novel notion of PoQoEA. So the requester can efficiently prove to the contract to reject a certain answer, if the worker does not meet the pre-specified quality standard $\Theta$. If an answer is out of the specified range, the requester is allowed to use verifiable encryption VPKE to reveal that to reject payment.

**Remark.** $\mathcal{C}_{hit}$ captures the essence of smart contracts in reality, as it: (i) reflects the transparency of Turing-complete smart contract that is a stateful program handling pre-specified tasks publicly; (ii) captures a contract that can access the cryptocurrency ledger to honestly deal with conditional payments; (iii) models the network adversary who is consulted to schedule the delivering order of so-far-undelivered messages.

C. Instantiating cryptographic building blocks

For sake of completeness, we hereafter give the constructions of cryptographic building blocks. Let $\mathcal{G} = (g)$ be a cyclic group of prime order $p$, where $g$ is a random generator of $\mathcal{G}$.

**(Short range) verifiable encryption (VPKE)** is based on exponential ElGamal. The private key $k \overset{\$}{\leftarrow} \mathbb{Z}_p$, the public key $h = g^k$, the encryption is $\text{Enc}_h(m) = (c_1, c_2) = (g^r, g^m h^r)$, and the decryption is $\text{Dec}_k((c_1, c_2)) = \log(c_2/c_1^k)$, where $\log$ is to brute-force the short plaintext range to obtain $m$; if decryption fails to output $m$ in range, then $c_2/c_1^k$ is returned. In addition, to efficiently augment the above ($\text{Enc}_h, \text{Dec}_k$) to be verifiable, we adopt a variant of Schnorr protocol with Fiat-Shamir transform in random oracle model.

- **ProvePKE$_k((c_1, c_2)).** Run $\text{Dec}_k((c_1, c_2))$ to obtain $m \in \text{range (or } g^m \text{ if } m \not\in \text{range}).$ Let $x \overset{\$}{\leftarrow} \{0, 1\}^\lambda$. Compute $A = c_1^x$, $C = H(A||g^{m/2})$, $Z = x + kc$ and $\pi = (A, Z)$. If $m \in \text{range},$ output $(m, \pi);$ else, output $(g^m, \pi).$

**The protocol of HITs $\Pi_{hit}$**

$\Pi_{hit}$ is among the requester $\mathcal{R}$, the workers $\{W_j\}$ and $\mathcal{C}_{hit}$. The protocol is divided into three phases:

---

**Phase 1: Publish Task**

- **Requester $\mathcal{R}$:**
  - Upon receiving the parameters $G, G_s, \Theta, N, \text{range}, \mathcal{B}, K$ of a HIT to publish:
    - $key_{h} \overset{\$}{\leftarrow} \{0, 1\}^\lambda$
    - $\text{comm}_{gs} \leftarrow \text{Commit}(G||G_s, key_{h})$
    - send (publish, $N, \mathcal{B}, K, \text{range}, \Theta, h, \text{comm}_{gs}$) to $\mathcal{C}_{hit}$

**Phase 2: Collect Answers**

- **Worker $W_j$:**
  - Upon receiving (published, $\mathcal{R}, N, \mathcal{B}, K, \text{range}, \Theta, h, \text{comm}_{gs}$) from $\mathcal{C}_{hit}$:
    - get the answer $a_j = (a_{(1,j)}, \ldots, a_{(N,j)})$
    - $c_j \leftarrow (\text{Enc}_h(a_{(1,j)}), \ldots, \text{Enc}_h(a_{(N,j)}))$
    - $\text{comm}_c \leftarrow \text{Commit}(c_j, key_{h})$, where $key_{h} \overset{\$}{\leftarrow} \{0, 1\}^\lambda$
    - send (commit, $\text{comm}_c$) to $\mathcal{C}_{hit}$
  - Upon receiving (committed, $\text{comms}$) from $\mathcal{C}_{hit}$:
    - if $(W_j, a_j)$ in $\text{comms}$, send (reveal, $c_j$, $key_{h}$) to $\mathcal{C}_{hit}$

**Phase 3: Evaluate Answers**

- **Requester $\mathcal{R}$:**
  - Upon receiving (revealed, answers) from $\mathcal{C}_{hit}$:
    - send (golden, $G, G_s, \text{comm}_{gs}$) to $\mathcal{R}$
    - for each $(W_j, c_j)$ in answers:
      - decrypt each item in $c_j$ to get $a_j = (a_{(1,j)}, \ldots, a_{(N,j)})$
      - $\exists a_{(i,j)} \in a_j$ s.t. $a_{(i,j)} \not\in \text{range}$:
        - $(a_{(i,j)}, \pi_i) \leftarrow \text{ProvePKE}_k(c_{(i,j)})$
        - send (outrange, $W_j, i, a_{(i,j)}, \pi_i$) to $\mathcal{C}_{hit}$
      - else if $\chi_j = \text{Quality}(\text{Dec}(c_j, sk_R); G, G_s) < \Theta$:
        - $\pi \leftarrow \text{ProveQuality}_k(c_j, \chi_j, G, G_s)$
        - send (evaluate, $W_j, \chi_j, \pi$) to $\mathcal{C}_{hit}$

---

**D. Security analysis**

**Theorem 1.** Conditioned on the hardness of DDH problem and static corruptions, the stand-alone instance of $\Pi_{hit}$ securely realizes $\mathcal{F}_{hit}$ in $\mathcal{C}_{hit}$-hybrid, random oracle model.

**Proof.** (sketch) Let $\mathbb{C}$ denote the set of corrupted parties controlled by the adversary $\mathcal{A}$, and let $\mathbb{H}$ denote the set of rest honest parties. For any P.P.T. adversary $\mathcal{A}$ in the real world, we
can sketch a P.P.T. simulator in the ideal world to interact with the ideal functionality $F_{hit}$ and corrupted parties, such that $S$ can emulate the actions of honest parties and the contract $C_{hit}$. $S$ proceeds as follows:

- **Publish Task (Phase 1).** If $R \in C$, considering that the corrupted $R$ sends the publish message to $C_{hit}$ in the real world, $S$ can trivially simulate that with interacting with $F_{hit}$. If $R \in H$, when the honest $R$ sends the publish message to $F_{hit}$, $S$ is informed and thus allows $S$ to simulate the phase of publish task (in the real world) for that $R$.

- **Collect Answers (Phase 2).** In the real world, the P.P.T. adversary $A$ might: (i) corrupt a set of parties $C$ up to including the requester and a set of the workers, and (ii) is also consulted to reorder the so-far-undelivered messages sent to $C_{hit}$ (till the next clock).

  The basic strategy to emulate $A$ is that: $S$ invokes the adversary $A$ to obtain how $A$ is re-ordering the commit messages (sent from workers), let $W$ to represent the set of workers whose commit messages are scheduled as the first $K$ to deliver; then $S$ delays all answer messages that are not sent from the workers in $W$. Then, $S$ internally simulates the ciphertexts sent via reveal messages to open commitments.

  If $R \in H$, the ciphertexts can be simulated as they are indistinguishable from the uniform distribution over the ciphertext space; if $R \in C$, $S$ is informed about all answer submissions sent from the workers, thus can internally simulate the submissions of the workers in the real world.

  Moreover, if $A$ corrupts a worker whose commit message is scheduled in the first $K$ to deliver but does not send any reveal message to open the commitment, the simulator $S$ can simulate that since it can let the corrupted worker to send an answer message containing $\perp$ with $F_{hit}$. In addition, it is trivial to see $S$ can internally simulate the parties as well as $C_{hit}$, when the adversary $A$ corrupts a worker to submit duplicated commitment.

- **Evaluate Answers (Phase 3).** The simulation becomes clear, if considering the security requirements of commitment scheme, VPKE, and PoQoEA. If the requester $R \in C$, the simulator $S$ invokes $A$ to obtain all outrageous and/or evaluate messages sent to $C_{hit}$, and then simulates the interactions. If the requester $R \in H$, whenever $R$ sends outrageous and/or evaluate messages to $F_{hit}$, $S$ is informed and hence is allowed to simulate the interactions between $C_{hit}$ and $R$ in the real world.

VI. Dragoon: Implementation & Evaluation

To demonstrate the feasibility of our protocol, we implement it to build Dragoon, and then use the system to launch a typical image annotation task for ImageNet atop Ethereum.

**System overview.** Dragoon consists of an on-chain part and an off-chain part: the on-chain smart contract is deployed in Ethereum ropsten network; the requester client and worker clients are implemented in Python 3.6. The off-chain clients are installed in a PC that uses Ubuntu 14.04 LTS and equips Intel Xeon E3-1220V2 CPU and 16 GB main memory.

**ImageNet’s HIT task.** We demonstrate our system through an ImageNet task $[12, 47]$, which is specified as: each task is made of 106 binary questions, 100 out of which are non-gold-standard questions, while the remaining 6 questions are requester’s gold-standard challenges; 4 workers are allowed to participate; if a worker cannot correctly answer at least four golden standard questions, his submission will be rejected without being paid, otherwise he deserves to get the payment.

**Cryptographic modules.** The hash function is instantiated by keccak256. We choose the cyclic group $G$ by using the $G_1$ subgroup of BN-128 elliptic curve, over which all concrete public key primitives are instantiated.

**Code availability.** The code of our prototype is available at
https://github.com/njit-bc/dragoon. An experiment instance is atop Ethereum ropsten network (https://ropsten.etherscan.io/address/0x5481b096c78c8e09c1bfbf694e934637f7d66698).

**Implementation details.** Many non-trivial on- and off-chain optimizations are particularly made for practicability.

**Off-chain ends.** The requester end warps: (i) an Ethereum node to interact with the blockchain, e.g. publish task, download workers’ submissions, etc; (ii) the prover of verifiable encryption to generate necessary proofs to instruct the contract to reward workers; (iii) a Swarm API to publish the detailed questions of each crowdsourcing task. Swarm $[53]$ is an off-chain storage network, where the questions of HIT is stored; in addition, to ensure integrity of HIT questions, the digest of the questions is committed in the contract, which significantly reduces on-chain cost, without violating securities.

The worker client wraps Ethereum to interact with the blockchain to read task and submit answers, and also incorporates Swarm client to allow download task questions.

**On-chain optimizations.** We carefully perform a few non-trivial system-level optimizations to lighten the task contract: (i) we implement all public key schemes over $G_1$ subgroup of BN-128 $[54]$, since we can use some precompiled contracts in Ethereum to do algebraic operations there cheaply $[53]$; (ii) it is expensive to store ciphertexts in the contract as internal
variables, so we make the contract store their 256-bit hashes instead and let the actual ciphertexts included in the chain as emitted event logs \[50\].

**Evaluations.** We conduct intensive experiments to measure the concrete performance, and discuss the system feasibilities from the on-chain side and the off-chain side.

**Off-chain costs.** First, Dragoon enables the requester to manage only one private-public key pair throughout all her tasks, because all protocol scripts are simulatable without secret key and therefore leak nothing relevant. More importantly, the off-chain cost of proving relevant cryptographic proofs is significantly reduced by removing unnecessary generality.

**On-chain costs.** We measure the critical on-chain performance from many angles including the cost of verifying zk-proofs and the on-chain gas usage of the whole protocol. First, we compare the verifying cost of concrete constructions and generic zk-proofs for VPKE and PoQoEA (six golden standards) in Table I. The concrete proof is fast, even compared to generic zk-proof (SNARK) known for efficient verification. For example, in the case of ImageNet task, only 6ms is needed to verify each concrete PoQoEA proof.

Table II clarifies the requester suffers from hindersome off-chain burden of generating generic zk-proofs. In contrast, our concrete constructions remove such bottleneck and boosts decentralized HITs practically. First, the requester can generate a proof to reject a worker’s submission within only a few milliseconds, which costs nearly 2 minutes if using generic zk-proof. Second, the concretely efficient constructions also save in memory usage. For example, by generic zk-proof, rejecting a submission requires a peak memory usage of 10 GB, which is reduced to only 53 MB by concrete constructions.

**Open problems.** It hints that the special-purpose protocols are promising to decentralize various crowdsourcing with high-security assurance as well as efficiency. It immediately corresponds to a few realistic problems to explore. For example, can we design a concretely efficient protocol to decentralize participatory crowd-sensing that is minimally meaningful with the needed fairness and privacy? Such the problem is challenging, since there seems no explicit requester to “prove” the quality of encrypted data anymore. Unfortunately, letting the blockchain learn encrypted data’s quality (without a prover) falls into the category of (multi-input) functional encryption, which is unclear how can be solved practically till today.

Another fundamental problem is that we consider security due to conventional cryptographic notions, where corrupted parties are fully controlled by an adversary and honest parties follow the protocol independently. The model has an inherent drawback to explain why rational workers would not deviate (e.g. by colluding). To resolve the concern, an “incentive-compatible” protocol is required, so “following the protocol” is a Nash equilibrium to deter rational workers from deviating.

**ACKNOWLEDGMENT**

We would like to thank the anonymous reviewers for their valuable suggestions and comments about this paper.
