Appendix A
Specification

A. Matching Feasibility

We can match two offers if they satisfy the following conditions: Job offer limit variables must be lower than the resource offer cap variables. Job offer max variables must be higher than the resource offer price variables. Additionally, there should be a mediator with the same architecture of JO which is trusted by both JC and RP.

$$RO.\mathit{instructionCapacity} \geq JO.\mathit{instructionsLimit} \tag{6}$$
$$RO.\mathit{ramCapacity} \geq JO.\mathit{ramLimit} \tag{7}$$
$$RO.\mathit{localStorageCapacity} \geq JO.\mathit{localStorageLimit} \tag{8}$$
$$RO.\mathit{bandwidthCapacity} \geq JO.\mathit{bandwidthLimit} \tag{9}$$
$$RO.\mathit{instructionPrice} \leq JO.\mathit{instructionMaxPrice} \tag{10}$$
$$RO.\mathit{bandwidthPrice} \leq JO.\mathit{maxBandwidthPrice} \tag{11}$$
$$JO.\mathit{architecture} = RP.\mathit{architecture} \tag{12}$$
$$JO.\mathit{directory} \in RP.\mathit{trustedDirectories} \tag{13}$$

$$\begin{aligned} \exists M :\ & M \in (JC.\mathit{trustedMediators} \cap RP.\mathit{trustedMediators}) \\ & \wedge M.\mathit{architecture} = RP.\mathit{architecture} \quad (14) \\ & \wedge JO.\mathit{directory} \in M.\mathit{trustedDirectories} \quad (15) \end{aligned}$$

$$\mathit{currentTime} + RP.\mathit{timPerInstruction} \cdot JO.\mathit{instructionLimit} \leq JO.\mathit{completionDeadline} \tag{16}$$

B. Payment

When the JC or the RP wants to post an offer for a job or resource, it will pay a deposit value to prevent it from cheating on the platform. This deposit value is a function of the posted offer which is more than the price of the job plus mediation. As the price of job or mediation is unclear at the time of posting the offer, a static $penaltyRate \gg 1$ is applied as security deposit.

After the job is finished and both the JC and RP agree on the outcome, JC will receive the deposit minus the cost of the job and the RP will receive the deposited value plus the cost of the job.

In case of disagreement, the match will go for mediation. The mediators have a fixed price for their resources. After the submission of their verdict, the deposit of the party who is at fault will be forfeited in favor of the winner and winner will receive both the deposits minus the cost of mediation.

$$JO.\mathit{deposit} = (instructionLimit \cdot instructionMaxPrice + bandwidthLimit \cdot bandwidthMaxPrice) \cdot (\theta + n) + \pi_a \tag{17}$$

$$RP.\mathit{deposit} = (instructionCap \cdot instructionPrice + bandwidthCap \cdot bandwidthPrice) \cdot (\theta + n) + \pi_a \tag{18}$$

$$\begin{aligned} \pi_c =& result.instructionCount \cdot \mathit{resourceOffer}.instructionPrice + \\ & result.bandwidthUsage \cdot \mathit{resourceOffer}.bandwidthPrice \end{aligned} \tag{19}$$

Appendix B
Data Structures

The platform is a composition of a smart contract, resource providers, job creators, Mediators, and Directories.

```
Platform{ mediators: Mediator[],
          resourceProviders: ResourceProvider[],
          jobCreators: JobCreator[],
          resourceOffers: ResourceOffer[],
          jobOffers: JobOffer[],
          matches: Match[],
          results: JobResults[],
          mediatorResults: MediatorResult[],
          penaltyRate : uint }
```

All information is stored within well-known storage services. The platform itself is agnostic to a particular service. However, it is assumed that all storage providers, which we call Directories, support SFTP/SCP Gateways for uploading and downloading documents. To use SFTP, the client must authenticate itself and then upload or download files using fully qualified name. A file once uploaded can be shared with other users if their user-id is known. The platform can support multiple architectures. However, initially it will only have amd64 and armv7.

```
enum Architecture{ amd64,
                   armv7 }
```

A. Entities

```
JobCreator{ trustedMediators: Mediator[]}

ResourceProvider{ trustedMediators: Mediator[],
                  trustedDirectories: address[],
                  arch: Architecture,
                  timePerInstruction: uint}
```

`trusteDirectories` lists Directory entities which the Resource Provider can use for both download and upload (the latter typically requires having some account on the Directory). `timePerInstruction` is an estimate of how much time the Resource Provider needs on average to execute an instruction, which is used when matching to check if the Resource Provider could finish the job before the deadline.

1) Mediators: Mediators are used for mediation. The mediator structure looks very much like the resource provider, except that it includes additional costs for mediation.

```
Mediator{ arch: Architecture,
          instructionPrice: uint,
          bandwidthPrice: uint,
          trustedDirectories: address[],
          supportedFirstLayers: uint,
          availabilityValue: uint  }
```

B. Offers

The format of the job offer is shown below. `jobCreator` is a unique identifier for the JC. The `depositValue` is the JC's security deposit for the job. The limits specify how many instructions the Job Creator is willing to pay for (after executing this many, the Resource Provider can give up and still get paid), how much RAM, local storage, and bandwidth the Resource Provider may have to use (again, after reaching these limits, the Resource Provider may stop). `instructionMaxPrice` specifies the maximum price per instruction that the Job Creator accepts, `bandwidthMaxPrice` specifies the maximum price per downloaded/uploaded byte (for the job execution layer, not for the base layer) that the Job Creator accepts. `completionDeadline` is the deadline of RP for submitting the solution. `matchIncentive` is the reward offered by the JC to the Solver. `firstLayerHash` is the hash of the job's Docker image. `URI` gives the location of the job-specific files on the Directory. `directory` is the identifier of the trusted directory where the job is located. The `jobHash` is a hash of the Docker image. `arch` specifies which architectures support the job.

```
JobOffer{ depositValue: uint,
          instructionLimit: uint,
          bandwidthLimit: uint,
          instructionMaxPrice: uint,
          bandwidthMaxPrice: uint,
          completionDeadline: uint,
          matchIncentive: uint,
          jobCreator: JobCreator,
          firstLayerHash: uint,
          ramLimit: uint,
          localStorageLimit: uint,
          uri: bytes,
        directory: address[],
        jobHash: uint,
        arch: Architecture;}

ResourceOffer{ resProvider: address,
               depositValue: uint,
               instructionPrice: uint,
               instructionCap: uint,
               memoryCap: uint,
               localStorageCap: uint,
               bandwidthCap: uint,
               bandwidthPrice: uint,
               matchIncentive: uint,
               verificationCount: uint;}
```

Prices are per instruction or per byte. Capacities specify what resources the Resource Provider has, and they are used as constraints for matching. All of the participants can deposit as much as they want as long as it is more than the required `JO/RP.deposit`. `depositValue` is the variable that holds the amount of deposited value for each offer.

C. Results

```
JobResult{ status: ResultStatus,
           uri: bytes,
           matchId: uint,
           hash: uint,
           instructionCount: uint,
           bandwidthUsage: uint,
           reacted: Reaction,
           timestamp: uint; }
```

`uri` gives the location of the result on the Directory. `instructionCount` is the number of instruction that were executed by the ResourceProvider. `bandwidthUsage` is the number of bytes downloaded / uploaded by the Resource Provider for the job (not counting the download of Docker layers). The JC has a specific deadline for responding to a result. Whether to approve or decline it. If the deadline is missed the RP can accept the result instead of the JC.

```
MediatorResult{ status: ResultStatus,
                uri: bytes,
                matchId: uint,
                hash: uint,
                instructionCount: uint,
                bandwidthUsage: uint,
                verdict: Verdict,
                faultyParty: Party; }
```

The platform on itself cannot determine which party is cheating merely based on the `hash` of the Mediator's Results. For example, the smart contract can not check whether the job was correctly posted to the directory by the Job Creator or the result was correctly posted to directory by the Resource Provider. Therefore, it is the responsibility of the Mediator to decide who should be punished in the ecosystem. `verdict` is the reason for deciding on the cheating party, and `faultyParty` is the cheating party.

```
Match { resourceOffer: ResourceOffer,
        jobOffer: JobOffer,
        mediator: Mediator}

enum ResultStatus{ Completed,
                   Declined,
                   JobDescriptionError,
                   JobNotFound,
                   MemoryExceeded,
                   StorageExceeded,
                   InstructionsExceeded,
                   BandwidthExceeded,
                   ExceptionOccured,
                   DirectoryUnavailable }
```

`Completed` means that the Resource Provider finished the job successfully and posted the results on the Directory. `JobDescriptionError` means that there is an error in the job description. `MemoryExceeded`, `InstructionsExceeded`, `BandwidthExceeded` mean that the job exceeded the limits specified in the JobOffer. `ExceptionOccured` means that an exception was encountered while executing the job, while `DirectoryUnavailable` means that the Resource Provider is unable to post results because the Directory is unavailable.

```
enum Verdict{ ResultNotFound,
              TooMuchCost,
              WrongResults,
              CorrectResults,
              InvalidResultStatus}
```

This is the reason that the Mediator chose to punish an actor. `ResultNotOnDirectory` means that the Resource Provider did not put the results in the directory. `TooMuchCost` means that the Resource Provider charged too much for the job. `WrongResults` means that the Resource Provider provided wrong results. `CorrectResults` means that the Resource Provider completed the job correctly and the JobCreator should be punished by sending the result for mediation. `InvalidResultStatus` means that the Resource Provider's mentioned ResultStatus is wrong. For example, it said that the job description was invalid and returned `JobDescriptionError`, but the description was correct.

```
enum Party{ ResourceProvider, JobCreator }
```

These are the trustless parties in the ecosystem. Mediators should specify who cheated in a job and should be punished by specifying one of these parties.

D. Smart Contract Functions

TABLE V
FUNCTIONS PROVIDED BY THE SMART CONTRACT AND THEIR CORRESPONDING EVENTS, AS WELL AS THE AGENTS WHO CALL THE FUNCTIONS AND THE AGENTS THAT SUBSCRIBE TO THOSE EVENTS.

Function	Event	Caller	Subscriber
`setPenaltyRate`	`PenaltyRateSet`	Deployment	NA
`setReactionDeadline`	`ReactionDeadlineSet`	Deployment	NA
`registerMediator`	`MediatorRegistered`	M	M,S,JC,RP
`mediatorAddTrustedDirectory`	`MediatorAddedTrustedDirectory`	M	M,S
`mediatorAddSupportedFirstLayer`	`MediatorAddedSupportedFirstLayer`	M	M,S
`registerResourceProvider`	`ResourceProviderRegistered`	RP	RP,S
`resourceProviderAddTrustedMediator`	`ResourceProviderAddedTrustedMediator`	RP	RP,S
`resourceProviderAddTrustedDirectory`	`ResourceProviderAddedTrustedDirectory`	RP	RP,S
`resourceProviderAddSupportedFirstLayer`	`ResourceProviderAddedSupportedFirstLayer`	RP	RP,S
`registerJobCreator`	`JobCreatorRegistered`	JC	JC,S
`jobCreatorAddTrustedMediator`	`JobCreatorAddedTrustedMediator`	JC	JC,S
`postResOffer`	`ResourceOfferPosted`	RP	RP,S
`postJobOffer`	`JobOfferPosted`	JC	JC,S
`cancelJobOffer`	`JobOfferCanceled`	JC	JC,S
`cancelResOffer`	`ResourceOfferCanceled`	RP	RP,S
`postMatch`	`Matched`	S	S,RP
`postResult`	`ResultPosted`	RP	RP,JC
`rejectResult`	`ResultReaction,JobAssignedForMediation`	JC	JC,M,RP
`acceptResult`	`ResultReaction,MatchClosed`	JC	JC,RP
`postMediationResult`	`MediationResultPosted,MatchClosed`	M	M,JC,RP
`timeout`	`MatchClosed`	JC	JC,RP

APPENDIX C
PROOFS: EQUILIBRIUM ANALYSIS

TABLE VI
RP PAYOFFS BY DECISION

	verify	pass
execute	$\overbrace{-c_e - g_r - \pi_a + \pi_c\Big(np_a^n\,(p_a - 1) + p_a + p_a^n\theta\,(p_a - 1)\Big) + \pi_d\,(1 - p_a)\,(1 - p_a^n)}^{U_{EV}^{RP}}$	$\overbrace{\pi_c - c_e - g_r - \pi_a}^{U_{EP}^{RP}}$
deceive	$\underbrace{-c_d - g_r - \pi_a + p_a^n\pi_c\,(-n - \theta) + \pi_d\,(1 - p_a^n)}_{U_{DV}^{RP}}$	$\underbrace{\pi_c - c_d - g_r - \pi_a}_{U_{DP}^{RP}}$

TABLE VII
JC PAYOFFS BY DECISION

	verify	pass
execute	$\overbrace{b - g_j - \pi_c\,(n + \theta)\,(1 - p_a)\,(1 - p_a^n) + (1 - p_a)\,(-g_m + p_a^n\pi_d) - c_v - p_a\pi_c - \pi_a}^{U_{EV}^{JC}}$	$\overbrace{b - g_j - \pi_a - \pi_c}^{U_{EP}^{JC}}$
deceive	$\underbrace{-c_v - g_j - g_m + p_a^n\pi_d - \pi_a - \pi_c\,(n - p_a^n\,(n + \theta) + \theta)}_{U_{DV}^{JC}}$	$\underbrace{-g_j - \pi_a - \pi_c}_{U_{DP}^{JC}}$

TABLE VIII
RP PAYOFFS BY DECISION WHEN $\pi_d = \pi_c$

	verify	pass
execute	$\overbrace{-c_e - g_r - \pi_a + \pi_c + p_a^n\pi_c(p_a - 1)\,(n + \theta + 1)}^{U_{EV}^{RP}}$	$\overbrace{\pi_c - c_e - g_r - \pi_a}^{U_{EP}^{RP}}$
deceive	$\underbrace{-c_d - g_r - \pi_a + \pi_c - p_a^n\pi_c\,(n + \theta + 1)}_{U_{DV}^{RP}}$	$\underbrace{\pi_c - c_d - g_r - \pi_a}_{U_{DP}^{RP}}$

TABLE IX
JC PAYOFFS BY DECISION WHEN $\pi_d = \pi_c$

	verify	pass
execute	$\overbrace{b - c_v - g_j - g_m\,(1 - p_a) - \pi_a + \pi_c\,(1 - 2p_a) - \pi_c\,(1 - p_a)\,(1 - p_a^n)\,(n + \theta + 1)}^{U_{EV}^{JC}}$	$\overbrace{b - g_j - \pi_a - \pi_c}^{U_{EP}^{JC}}$
deceive	$\underbrace{-c_v - g_j - g_m - \pi_a + \pi_c + \pi_c\,(p_a^n - 1)\,(n + \theta + 1)}_{U_{DV}^{JC}}$	$\underbrace{-g_j - \pi_a - \pi_c}_{U_{DP}^{JC}}$

Theorem 1 (JC should not always pass ($p_v > 0$))**.** *If the JC always* passes *(i.e., $p_v = 0$), then the RP's best response is to always* deceive *(i.e., $p_e = 0$).*

Proof. Assume that the JC always passes, then the RP will either execute or deceive. Since $U_{EP}^{RP} < U_{DP}^{RP}$ is always true then the RP will always deceive. This corresponds to the JC utility U_{DP}^{JC} which is always negative, thus if the JC will always pass then it should not participate. Therefore a JC that is participating will not always pass. □

TABLE X
SIMPLIFY RP UTILITY TO ASSESS DOMINANT STRATEGY WITH $\pi_d = \pi_c$.

	verify	pass
execute	$-c_e + p_a^{n+1}\pi_c\,(n+\theta+1)$	$-c_e$
deceive	$-c_d$	$-c_d$

TABLE XI
SIMPLIFY JC UTILITY TO ASSESS DOMINANT STRATEGY WITH $\pi_d = \pi_c$.

	verify	pass
execute	$-\pi_c\,(1-p_a)\,(1-p_a^n)\,(n+\theta+1) + (1-p_a)\,(-g_m + 2\pi_c)^{*1,2}$	$c_v^{*3,4}$
deceive	$2\pi_c - g_m - \pi_c\,(1-p_a^n)\,(n+\theta+1)^{*1,3}$	$c_v^{*2,4}$

Theorem 2 ($p_e > 0$)**.** *If $p_v > 0$ and $p_a^{n+1} > \frac{1}{2}$, then a rational RP must execute the jobs with non-zero probability.*

Proof. If the JC verifies the The RP dominant strategy is to execute if $U_{DV}^{RP} < U_{EV}^{RP}$. Deriving Table X from Table VIII we know that to execute is dominate if

$$\begin{aligned} -c_e + p_a^{n+1}\pi_c\,(n+\theta+1) &> -c_d \\ \text{rearrange terms} \quad p_a^{n+1}\pi_c\,(n+\theta+1) &> c_e + -c_d \end{aligned} \tag{20}$$

From Table I we know that $n > 0$ so then $2\pi_c < \pi_c(\theta + n + 1)$. We also know $c_e < \pi_r \leq \pi_c$. Substituting these into Eq. (21) results in:

$$\begin{gathered} \left\{p_a^{n+1}\pi_c\,(n+\theta+1) \geq 2\pi_c p_a^{n+1}\right\} > \left\{\pi_c \geq c_e \geq c_e - c_d\right\} \\ \text{Simplify:} \quad 2\pi_c p_a^{n+1} > \pi_c \\ \text{Solve for } p_a^{n+1}\text{:} \quad p_a^{n+1} > \frac{1}{2} \end{gathered} \tag{21}$$

Thus since $p_v > 0$ from Theorem 1, if Eq. (21) is true then executing is the dominant strategy and $p_e > 0$. □

Theorem 3 (Bounded p_a)**.** *If the parameters in the JC utility function are all set to minimize the optimal value for p_a (maximizing a rational JC's dishonesty), then any deviation will increase p_a. The platform controls n and θ, and so controls the minimum optimal value of p_a.*

Proof. The system must bound p_a for any set of values in the system, thus we use the JC's total expected utility.

$$\begin{aligned} U^{JC} =& p_v p_e U_{EV}^{JC} + p_v(1-p_e)U_{DV}^{JC} + (1-p_v)p_e U_{EP}^{JC} + (1-p_v)(1-p_e)U_{DP}^{JC} \\ U^{JC} =& b\sigma_e - c_v\sigma_v - g_j + g_m\sigma_v\,(p_a\sigma_e - 1) - 2p_a\pi_c\sigma_e\sigma_v - \pi_a + \\ & \pi_c\sigma_v\,(n+\theta+1)\,(-p_a p_a^n\sigma_e + p_a\sigma_e + p_a^n - 1) + 2\pi_c\sigma_v - \pi_c \end{aligned} \tag{22}$$

take the derivative with respect to p_a:

$$\frac{\partial}{\partial p_a}U^{JC} = g_m\sigma_e\sigma_v - 2\pi_c\sigma_e\sigma_v + \pi_c\sigma_v\,(n+\theta+1)\left(-np_a^n\sigma_e + \frac{np_a^n}{p_a} - p_a^n\sigma_e + \sigma_e\right) \tag{23}$$

Set equal to 0 and simplify:

$$\frac{2\pi_c - g_m}{\pi_c(n+\theta+1)} = 1 - p_a^n + \frac{np_a^{n-1}}{p_e} - np_a^n \tag{24}$$

If g_m increases lhs decreases meaning p_a will increase.
If π_e increases then rhs decreases meaning p_a will increase.
π_c has no impact on p_a when $g_m = 0$. If $g_m \neq 0$ then when π_c increases, the left-hand side (lhs) increases and approaches $\frac{2}{\theta+n+1}$ (approaching parity with $g_m = 0$), then the right-hand side (rhs) must also increase meaning that p_a must decrease.
If θ increases lhs decreases meaning p_a will increase.
If n increases p_a increases, see where the curves cross 0 in Fig. 9.

The system assumes a value for each parameter such that any change results in p_a increasing. Specifically if the parameter and p_a are inversely related set the parameter to its maximum allowed value, and if they are directly related set the parameter to its minimum value. Thus the worst case values for each of the parameters are: $g_m = 0$, $p_e = 1$, $\theta = 0$, $n = 1$. The plot in Fig. 5 when $n = 1$ uses those parameters, and shows that increasing n does cause the optimal value for p_a to increase. We see that when $n = 1$, the optimal $p_a = 0.5$; and when $n = 4$, the optimal $p_a = 0.943$. Thus, we see that setting n can set a lower bound on the optimal p_a. Parameter θ also can adjust the lower bound on the optimal value for p_a.

□

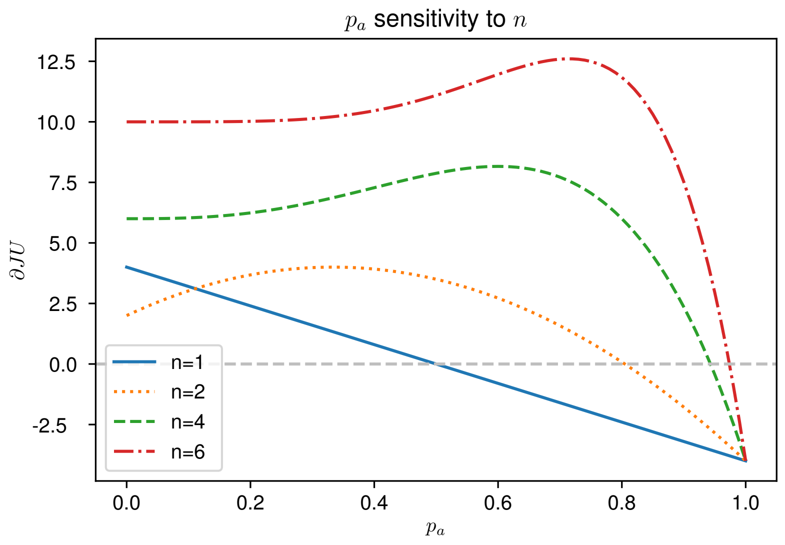

Fig. 9. In this plot we vary the value of n and plot p_a against $\frac{\partial U^{JC}}{\partial p_a}$. This shows that as n increases so does the optimal value of p_a. For this plot $\pi_c = 2$, $g_m = 0$, $\theta = 0$, $c_v = 1$, $b = 4$, $p_e = 1$

Theorem 4 (JC type 1)**.** *If the JC is type 1, it will always verify ($p_v = 1$) since c_v is sufficiently low. This results in a pure strategy equilibrium <execute,verify>.*

Proof. Type 1 means that $U^{JC}_{EV} > U^{JC}_{EP}$ and $U^{JC}_{DV} > U^{JC}_{DP}$ so verify strictly dominates passing for the JC. From Theorem 3 we know that $p_a^(n+1) > \frac{1}{2}$ and from Theorem 2 we know that if $p_a^(n+1) > \frac{1}{2}$ and the JC verifies then the RP will execute thus <execute,verify> is a pure strategy equilibrium. □

Theorem 5 (JC type 2)**.** *If the JC is type 2, it results in two pure strategy equilibria <execute,verify>, <deceive,pass>, and one mixed strategy Nash equilibrium where the JC randomly mixes between verifying and passing.*

Proof. Type 2 means that verify dominates when the RP executes ($U^{JC}_{EV} > U^{JC}_{EP}$) and pass dominates when RP deceives ($U^{JC}_{DV} < U^{JC}_{DP}$). Following Theorem 4 we know that <execute,verify> is a pure strategy equilibrium if the JC always verfies and since $U^{JC}_{EV} > U^{JC}_{EP}$ the JC will not choose to pass.

<deceive,pass> is a pure strategy equilibrium because $U^{JC}_{DV} < U^{JC}_{DP}$ so the JC will not change to verify, and RP will not change to execute because $U^{RP}_{EP} < U^{RP}_{DP}$ is always true.

The JC will only verify all jobs if the cost of verification c_v is sufficiently low (0). Otherwise it may get better utility by adopting a mixed strategy, choosing to verify with probability p_v and choosing to pass with probability $1 - p_v$. However if the JC passes, the RP will prefer to deceive (assuming that $p_a^{n+1} > \frac{1}{2}$) and so will also choose to adopt a mixed strategy where it executes with probability p_e and deceives with probability $1 - p_e$.

□

Theorem 6 (JC type 3)**.** *If the JC is type 3, it will result in a Nash equilibrium where the JC randomly mixes between verifying and passing, and the RP mixes between executing and deceiving.*

Proof. Type 3 means that pass dominates when the RP executes ($U^{JC}_{EV} < U^{JC}_{EP}$) and verify dominates when RP deceives ($U^{JC}_{DV} > U^{JC}_{DP}$). Assume the JC will verify and the RP will execute: <execute,verify>. Then since $U^{JC}_{EP} > U^{JC}_{EV}$ the JC will transition to pass: <execute,pass>. Now since $U^{RP}_{EP} < U^{RP}_{DP}$ is always true the RP will transition to deceive: <deceive,pass>. Then the JC will transition to verify since $U^{JC}_{DV} > U^{JC}_{DP}$: <deceive,verify>. And finally from Theorem 2 the RP will transition to execute: <execute,verify>. Thus we return to the initial condition and there is not pure strategy equilibrium. From [34] we know that all finite games have an equilibrium and so there must exist a mixed strategy equilibrium. □

APPENDIX D
FUNCTION GAS COSTS

$$\begin{aligned}\text{nominal cost} &= \texttt{postjobOfferPartOne} + \texttt{postjobOfferPartTwo} + \texttt{acceptResult}\\ \text{nominal cost} &= 2.36E5 + 2.16E5 + 1.40E5 \qquad (25)\\ \text{nominal cost} &= 5.92E5\end{aligned}$$

TABLE XII
GAS COST FOR EACH PLATFORM FUNCTION CALL, MEASURED UPON RECEIPT OF EVENT EMITTED BY THAT FUNCTION. GASESTIMATE IS THE VALUE RETURNED WHEN CALLING THE ETH_ESTIMATEGAS METHOD

Function	mean	std.dev.	gasEstimate
setPenaltyRate	4.34E+04	0.00E+00	2.21E+04
setReactionDeadline	4.31E+04	0.00E+00	2.16E+04
acceptResult	1.40E+05	6.47E+04	6.29E+04
postResult	1.91E+05	1.60E+04	2.25E+05
postMatch	1.35E+05	1.70E+03	1.27E+05
postJobOfferPartOne	2.36E+05	4.59E+03	2.28E+05
postJobOfferPartTwo	2.16E+05	0.00E+00	2.27E+05
postResOffer	2.42E+05	2.16E+03	2.32E+05
registerMediator	1.37E+05	0.00E+00	8.51E+04
registerResourceProvider	8.12E+04	0.00E+00	2.54E+04
resourceProviderAddTM	6.50E+04	0.00E+00	4.23E+04
registerJobCreator	3.40E+04	0.00E+00	3.20E+03
jobCreatorAddTM	6.53E+04	0.00E+00	4.26E+04
postMediationResult	1.87E+05	1.56E+04	2.40E+05
rejectResult	5.39E+04	9.90E+00	4.71E+04
nominal cost	5.92E+05		