INCENTIVE POLITICS AND A SOLUTION TO THE HITMAN PROBLEM

Howard Harmon
howardharmon@proton.me

September 2023

Abstract

The existence of a marketplace for users to anonymously post large-scale tasks with an associated monetary reward, designed so the executors of the task almost certainly receive the reward while preserving anonymity, would provide humanity with remarkable new capabilities. Jim Bell proposed one solution specific for assassination marketplaces in his *Assassination Politics* essays, but several flaws contained in Bell's solution and others have prevented their adoption. We propose a protocol allowing for the existence of task marketplaces that can be built using smart contracts, and that rectifies the flaws contained in existing proposals. Specifically, our protocol provides a means by which participants can fund a public contract detailing some task, in a way that is trustless, decentralized, and preserves anonymity, while ensuring the executor of the task is almost certainly paid, and the task is almost certainly completed. Successfully funded and completed contracts are precisely those with significant support, thereby providing a natural defense against the proliferation of especially abhorrent or criminal contracts.

1 Introduction

A long-standing open problem left undefined up to now yet widely studied, is the *Hitman Problem*. The Hitman Problem, in its most general form, asks: *how can a group of anonymous participants fund a public contract detailing any task, for which the executor of the contract is almost certainly paid a reward, does not communicate with the funders, and remains anonymous, while the contract posters can be sure of its completion or otherwise recoup the reward*? The problem – specifically a subcase concerned with creating assassination marketplaces – has been considered for decades. Early study and popularization began with work by Timothy May and Jim Bell, with more recent studies taking place in the academic literature [1, 2, 3, 4].

It is no coincidence that the Hitman Problem and its application in assassination marketplaces continues to be considered decades after its conception [4, 5, 6, 7, 8, 9, 10]. After all, the implications of a functioning assassination marketplace, or, more broadly, an anonymous task marketplace, would be significant. Extending well-passed facilitating assassinations, such task marketplaces could be used to disassemble crime syndicates, incentivize legislators and corporations to act on climate change, and even change how wars are waged (see Section 4).

Nonetheless, practical attempts at creating such marketplaces have failed [2, 5]. This failure can be attributed to the fact that present solutions to the Hitman Problem leave much to be desired, precluding their widespread adoption. Critical issues hampering such solutions include a dependency on centralized governing organizations, a lack of barriers preventing the creation of contracts detailing egregious tasks, and few safeguards protecting the interests of both contract funders and executors, such as strong guarantees of reward issuance, or strong guarantees that tasks described in contracts will be completed should they receive funding.

What is needed is a decentralized, trustless protocol for implementing task marketplaces, designed so that all participants may preserve their anonymity and need not communicate, while contract executors can be confident they will receive the reward for completing the task, and contract funders can be sure their contributions will go toward a task that will be completed. Moreover, the protocol must contain mechanisms critical to its basic operation that prevent the creation of especially egregious contracts, so it cannot be copied with such a mechanism removed. Such a protocol would facilitate the construction of a task marketplace that typical participants would be likely to use. The purpose of this paper is to provide a solution to the Hitman Problem achieving the standards set forth above.

This paper is organized as follows. Section 2 describes our protocol solving the Hitman Problem. Section 3 touches on how the protocol can be practically built, and Section 4 discusses a number of the most consequential applications our protocol enables.

2 Solving the Hitman Problem

In what follows we describe a protocol solving the Hitman Problem. We aim for this paper to be readable by a broad audience, so we reduce the amount of technical language used. But this does imply a loss of precision; our proposal is simple and easily communicated without extensive formalism.

Solution requirements

Any solution to the Hitman Problem should satisfy the following requirements.

1. *Decentralized*: No component of the solution can rely on a centralized body.

2. *Trustless*: The solution cannot require participants to place trust in anything other than the underlying platform (i.e. a consensus mechanism) and the proper function of the incentives built into the protocol. And, any outcome should follow from a majority consensus of all participants, rather than the decision of any minority.

3. *Anonymous*: Any solution should preserve the anonymity of the participants.

4. *Permissionless*: Anyone can participate.

5. *No coordination*: Participants should not have to coordinate with each other in any way.

6. *High probability of payment*: If a collection of *Executors* – who are actors working to complete a contract – accomplish the task detailed in a contract proposed by the network, they should receive payment with overwhelmingly high probability.

7. *High probability of contract execution*: Should a *proposal* (a proposed task which is published to receive monetary support) become an active contract, the contract should be completed with overwhelmingly high probability.

8. *Solution minimality and uncloneability*: The solution should be "minimal," so the efficacy of highly distinct solutions is compromised. This minimality also implies the solution should be uncloneable, in the sense that no party should be able to make an equally effective solution designed only for specific purposes (such as facilitating crime).

The last requirement is rather opaque. Yet, we will show it is possible to gain a reasonably firm grasp on how "minimal" a solution is.

With these requirements in mind, we describe a solution to the Hitman Problem that meets all solution requirements. The following description is informal, however it contains all details needed to easily arrive at a formal description with which one may construct the protocol.

The Protocol (informal)

1. *Proposal construction*: A collection of *Designers* begins by constructing a proposal – a smart contract with a queryable reference to a text and image description of a task to be completed (full operating details described below). In constructing the smart contract, the Designers fix a collection of four parameters (RewardThreshold, Time2MeetThreshold, PredictorFee, VotingTime), and a piecewise function FeeFunction mapping into $[\frac{1}{2}, 1)$.
2. *Proposal publication*: Upon completing the construction of a proposal, the Designers initialize the smart contract, paying any associated fees (i.e. gas, publishing fee). The proposal (smart contract) is now taken to be

an immutable object those in the network can interact with (operating as follows).

3. *Proposal funding phase*: Upon successful publication of the proposal, a timer begins counting down from Time2MeetThreshold. During this period, participants we call *Funders* will send currency to the proposal. Whenever a Funder sends x units of currency to the proposal, a fee of size FeeFunction(x) is deducted from the contribution. The fee is held by the contract, while the remaining $x \cdot$ FeeFunction(x) units of currency contribute to a Reward value. It is required that $1/2 \leq$ FeeFunction(x) < 1.

   If the value of Reward meets or exceeds the parameter value RewardThreshold before the timer runs out, then the proposal becomes an (active) contract, and all the collected fees are burned. If not, the proposal "terminates" – refunding all Funders the currency they contributed (along with the fee), before ceasing to interact with the network.

4. *Contract activation*: If a contract activates, Funders may continue to send currency to the contract (charged a fee as before, now immediately burnt), however, it is now impossible for any Funder to get the currency they sent to the contract back. All Funder contributions after fees continue to contribute to Reward. The contract can be queried to share the current Reward[1]. The Reward represents the maximum amount of currency a contract Executor could obtain (an Executor is an entity working to complete the task detailed in a contract for the reward).

5. *Prediction phase*: Whenever a participant sends currency to an (active) contract, they must also attach a MemoHash. If the participant is a Funder, MemoHash will be a hash of a large random value. If the participant intends to be an Executor, they must contribute at least PredictorFee units of currency to the contract, and MemoHash will be a hash of a file containing a precise description of exactly when and how the task will be completed, along with any unique and unpredictable steps the Executor will take to distinguish themselves for use in proving their responsibility for task completion later on[2].

6. *Prediction revealment*: At any time after a proposal becomes an active contract, an Executor can reveal a prediction. To do so, they must send the

[1] In addition, the contract can be queried to share how many distinct funders there are, and how much money each funder contributed.

[2] The use of predictions is not ideal, however in order to maintain the general use of the protocol, they appear necessary. It is possible that with enough development of zero-knowledge proofs and sophisticated cryptographic techniques which can verify a user's location at a date and time for instance, the use of predictions can be phased out. Moreover, there are certain tasks for which predictions are not needed or are guaranteed to work, and Executor verification can be carried out automatically (see [4] for examples).

contract the file containing their prediction (the task execution description as above), along with an identifier and indicator that they are revealing a prediction. The contract then checks if the Executor's shared identifier matches with that of a Funder who sent at least PredictorFee units of currency to the contract, and whether the file hashes to the corresponding MemoHash. If both of these conditions are met, the contract distributes the prediction file to the network and initiates a vote.

7. *Voting period*: Each Funder receives a copy of the file the Executor shared with the contract, along with the date the Executor funded the contract with PredictorFee and the MemoHash. It is now up to each Funder to review the Executor's shared file, along with the date the prediction was made, and any other information the Executor has made public (possibly after the prediction was made), and decide whether they, and any other Executors who revealed a prediction, deserve the reward. Accordingly, each Funder submits a "yes" or "no" vote to the contract for each Executor revealing a prediction, where each Funder has a weight of $x(1 - \mathsf{FeeFunction}(x))$ votes if they contributed x units of currency[3].

8. *Payment and contract termination*: After VotingTime has passed since the last prediction revealment, *or* VotingTime has passed since 67% of the participants have cast a vote, the contract tallies the votes and determines which Executors, if any, are to be given the reward by 50% majority consensus of Funders (at least half of *all* Funders, not just those who voted).

   The contract then distributes the reward between Executors in a manner corresponding to how each (voting) Funder indicated they would like their submitted currency to be distributed. For instance, if Funder α decides to send 5 units to Executor A and 2 units to Executor B, and Funder β decides to send 1 unit to Executor A and 3 units to B, then Executor A will get 6 units in reward, and Executor B will get 5 units in reward.

   But importantly, a majority of all Funders must agree on which Executors are "valid" and can be rewarded. If a majority of Funders vote "no" or do not agree on a common set of Executors to reward, then the voting halts with no Executor being rewarded, and the protocol continues. If, conversely, 50% or more of all Funders agree to reward the same collection of Executors, those Executors are rewarded with just the funds the voting 50% contributed (after fees). If at least 67% of the Funders agree to reward the same collection of Executors, then the entire reward – all

[3]Voting weighted by contribution amount is needed to avoid a number of attacks against the protocol. However, if "proof of humanity" technology reaches a sufficient level of sophistication, alternative voting methods could be used instead.

of the contributions (after fees) – are sent to the collection of Executors. In such an event, if a Funder did not vote, then their contributed funds (after fees) are equally distributed amongst the Executors agreed upon by the majority. The contract then terminates, ceasing to interact with the network.

9. *(Optional) contract Timeout*: The Designers may opt to put a timer on the contract, so that, following activation, if the task is not completed during the specified time window, or the Funders cannot agree on who to reward, the contract is terminated, and all Funder contributions are burnt. It is imperative that all contributions be burnt in such a situation.

   An alternative option is for the Designers to allow for a Funder to choose to burn their contribution after a contract is activated. This would provide an alternate form of "contract deescalation" to that above and would be a desirable feature in many situations.

As described, many of the steps of the protocol are computationally expensive and even dangerous, especially in the *Proposal construction* and *Prediction revealment* phases. This was deliberate to allow for readability and ease of understanding. For comments on practical implementation considerations resolving these issues see Section 3.

We now discuss how the parameters should be selected depending on the task described by the Designers, and in doing so we both justify our design choices and reveal why we deem this protocol to be a "minimal" solution to the Hitman Problem.

2.1 Incentives and optimal parameter selection

This subsection forms the heart of this paper. We describe the purpose of the RewardThreshold, Time2MeetThreshold, PredictorFee, and VotingTime parameters along with the FeeFunction, and in doing so reveal why the protocol functions as intended while meeting the solution requirements.

Funder and Executor behavior assumptions

The protocol integrates two basic Funder and Executor behavioral assumptions. The first key assumption we make is *any Funder seeks to maximize both contribution safety and effectiveness. That is, if a Funder contributes currency to a proposal/contract, they want to be sure that the task detailed in the proposal/contract will be executed with high probability, and if not then they can get all of their money back.*

The second key assumption we make is *Executors want to ensure that if they complete a task detailed in a contract, then they will receive their fair share of the reward with high probability.*

The protocol implicitly incorporates a third assumption about the Funder and Executor behavior. Namely, *both Funders and Executors wish to minimize the overall cost (in terms of time, money, energy, etc.) of interacting with a proposal or contract in any way, so long as the two basic assumptions continue to be met.* This implicit assumption greatly influences whether a solution to the Hitman Problem can be deemed "minimal" or not.

It is also assumed that Funders partake in voting and deciding which set of Executors receive the reward; Funders failing to interact with the contract upon contributing are relinquishing the power and influence their contribution purchased – something a rational actor is unlikely to do.

The FeeFunction and attacks

The protocol mandates a steep fee: for monetary contributions totaling x units of currency, $\mathsf{FeeFunction}(x)$ should not be less than $1/2$, indicating a fee of at least 50% for all contributions. To see the purpose of such a steep fee, consider the attack where Funders seek to obtain a partial or total refund of their contribution to a contract after the task has been completed by an Executor, thereby seeing the task completed with little cost incurred.

The only way for a Funder to obtain currency after a proposal becomes an active contract is to vote to have the reward directed to their own wallet, rather than the wallet of an Executor. By construction, a contract directs all or part of the reward into a collection of Executor wallets following a vote initiated by an Executor. And, the contract will only direct the reward into Executor wallets that a majority of Funders agree on (at least half of all contributors, including non-voting Funders).

Hence, for a Funder or group of Funders to receive all or part of the reward, they must control at least 50% of the votes. But by the design of the protocol, each Funder gets at most one vote for every two units of currency contributed to the proposal/contract (recall the number of votes is determined by $x(1 - \mathsf{FeeFunction}(x))$, and $\mathsf{FeeFunction}(x) \geq 1/2$). And, for any party to extract reward contributions others made, they must control 67% (2/3rds) of the vote. So, suppose the current reward associated with a contract is R, and a malicious party wishes to extract the reward. Then they must contribute at least $4R$ units of currency to obtain at most $2R$ votes, and hence a 2/3rds voting majority. But then said party can only extract $3R$ units of currency, which is less than the $4R$ units of currency they contributed to hijack the vote. Moreover, if a party contributed between $2R$ and $4R$ units of currency, then they would control between $1/2$ and 2/3rds of the vote, and so by the design of the protocol they could only extract half the funds they themselves contributed. *It follows that any conspiring group of funders can, by voting, only ever get out strictly less money than they put in.*

Of course, the protocol is left open to an attack where large Funders fund a contract to incentivize its completion (having contributed the majority of the reward), and then recoup some of their contribution after the task has been completed, by voting to have the reward directed into their wallet. But this attack is avoided by both publishing the number

of Funders and the total size of their contribution, along with the use of a non-constant FeeFunction. If FeeFunction places a fee steeper than 50% on smaller contributions, then such an attack would be infeasible even for a very wealthy adversary, as such an adversary would have to make many small contributions to fool the network into thinking the contract is supported by many distinct Funders. Moreover, supposing the majority of Funders (by contribution) are honest, and do not conspire to reap a partial refund after a task has been completed, then such an attack is prevented.

One more attack an adversary could mount is to "fake" completing the task. That is, an adversary could pose as an Executor, and try to "trick" the Funders into sending them the reward by revealing a fake prediction. But this attack is also easily avoided upon implementing the PredictorFee (see below), and by assuming basic due diligence on the part of both the Funders and the "true" Executor(s) when submitting a prediction.

Following the discussion of these attacks, we reach the following conclusion: *an Executor wishing to maximize the likelihood of receiving the reward upon task completion is incentivized to only complete tasks associated with contracts supported by high numbers of Funders, with no small number of Funders contributing more than 50% of reward funds, and to carefully craft their predictions submitted to a contract. As a consequence, Funders wishing to maximize the chances a contract they support is executed, are incentivized to only support proposals designed to become active contracts only after a great many Funders have contributed currency (implying a high RewardThreshold value).*

Assuming a rational Executor will only pursue tasks supported by a large number of participants with a large reward, we can conclude that the tasks they will seek to complete coincide with those having some high degree of popularity. That is, tasks large numbers of participants are willing to put money into supporting necessarily correlate with globally popular tasks, thereby implying very unpopular proposals would be unlikely to become active contracts, and if they did, it is unlikely that an Executor would pursue completion of that contract in fear that they may not be paid in view of the above attacks.

As such, contracts detailing unpopular tasks such as acts of terrorism or crime are unlikely to be pursued by Executors. And, if such a contract is both activated and pursued by an Executor, then the chances that the very same task would have been pursued by the same Executor without the support of the protocol are high. Hence, we argue it is unlikely that the use of the protocol would facilitate any increase in crime and terrorism (this is in contrast to, say, Augur's betting protocol which can easily be used in abhorrent ways as such systems have no built-in safeguards).

RewardThreshold & Time2MeetThreshold

Recalling our assumption concerning Funder behavior, we suppose that Funders who contribute to a proposal or contract will only do so if they are confident the task detailed will be completed, and if it is not then they will be refunded all of their contributions. The RewardThreshold parameter along with the Time2MeetThreshold parameter control how easy

it is to turn a proposal into an active contract: the RewardThreshold determines how much money must be contributed to a proposal in time Time2MeetThreshold for the proposal to become an active contact.

If a proposal becomes an active contract, no Funder can get their money back. As a consequence, *the RewardThreshold parameter must be so high that any Funder can be sure a reward of that magnitude will almost certainly incentivize a group of Executors to complete the task. In addition, the Time2MeetThreshold parameter must be large enough for Funders to notice the proposal and fund it, but not so large as to turn potential Funders away due to the time they would have to wait to receive a refund on a proposal that does not meet the RewardThreshold.*

PredictorFee

The PredictorFee parameter prevents pollution attacks wherein adversaries make large numbers of fake predictions to overwhelm the network with votes. In addition, the PredictorFee prevents adversaries from submitting many predictions that account for the various ways a task could be completed, so that the adversary can pretend to be the task Executor to fool the network into sending them all or part of the reward instead of the true Executor of the task.

To this end, the PredictorFee should but high. However, Designers ought to consider the task detailed in the proposal and ask if the PredictorFee is so high that it could outprice the typical Executor who may seek to complete the task. Deciding the PredictorFee thus requires the Designers to consider the canonical profile of an Executor, and set the PredictorFee to be the maximum possible value the Designers think potential Executors could afford.

VotingTime

Any Designer constructing a proposal needs simply ensure the VotingTime is a value large enough to ensure Funders will have time to cast a vote, while not being so large as to clog the flow of events.

We note that a vote will be closed if VotingTime has passed since at least 67% of all participants have cast a vote, in order to prevent an attack where an adversary keeps revealing a new prediction after VotingTime has passed, so that a vote can never close.

2.2 Additional remarks on design choices and consequences

The purpose of the MemoHash submitted with each contribution following contract activation is to make it infeasible for any party analyzing the network behavior to determine if a prediction has been made, and thus respond in some way, as contributions from Funders and Executors are indistinguishable.

The answer to the question *"Why not incorporate a mechanism into active contracts that allows Funders to get refunds in the event the task is not or cannot be completed?"* is clear. In short, any mechanism that allows Funders to obtain some kind of refund in the active contract phase will only incentivize behavior that reduces the chances that an honest task Executor wins the reward.

An optional "commitment" feature for proposals/contracts

In some cases a contract will be more likely to succeed if some Executing party de-anonymizes and commits to accomplish the task, supposing they are guaranteed payment upon task completion. If the Designers deem this to be a feature that should be implemented in the case of their specific task, then such a commitment scheme can be implemented as follows.

Upon contract activation, there will be a period during which candidate Executors reveal their identity and distribute descriptions as to why they should be the ones to complete the detailed task, paying a fee to do so. After some time has elapsed, a vote will occur, where Funders will choose a single Executor to complete the task. At this point, the Reward will be earmarked for the chosen Executor, unless the Funders vote to revoke their guarantee later on (as in the case that the Executor is not meeting specified progress points). After the Executor has accomplished the task, they will indicate as such to the network, and the Funders will vote to decide if the Reward is to be sent to the Executor. If so, all of the Reward is distributed to the Executor, and the contract terminates. If not, the process begins over again.

Optimality

In previously proposed solutions to the Hitman Problem, such as Bell's solution [3], anyone can "make their own" marketplace that is purpose-built for specific tasks that may not be supported on a different marketplace. This is especially the case for solutions relying on a centralized organizing body, as separate groups can create their own marketplaces for any purpose they wish. This is a serious problem considering how marketplaces could be easily constructed for the sole purpose of supporting abhorrent contracts.

In contrast, any collection of marketplaces implemented using our protocol would be fundamentally identical. Even if the underlying protocols differ in certain ways, the same basic mechanisms must be maintained to preserve the solution requirements, such as decentralization and strong guarantees of payment or task execution, while meeting the basic Funder and Executor behavior assumptions. But this implies our solution meets the "uncloneability" or "minimality" requirement, as any protocol meeting the same outcome must incorporate the key mechanisms our protocol integrates (particularly for those marketplaces supporting large-scale contracts with large rewards).

3 Remarks on Implementation

We begin by observing that the protocol can be easily implemented atop any blockchain supporting smart contracts. The *Proposal construction* step is essentially that of constructing a smart contract, and participants interacting with a proposal/contract are interacting with a smart contract. We do not provide more details on the subject: this paper aims to simply establish that construction of the protocol is indeed possible using modern technology and to leave full construction details to future work.

In constructing a proposal, the onus is on the Designers to write the task in extremely clear language, with little to no room left for interpretation, and clear parameters on what task completion looks like. To this end, Designers should follow a *one proposal one simple task* rule.

In a similar vein, it would be ideal if the implementation of the protocol incorporated some major technical or financial barrier to creating a proposal. Doing so would increase the likelihood that proposal creation becomes a "group effort," which would only improve the quality of proposed tasks. In addition, it would have the benefit of focusing funders on just a few proposals, as opposed to a sea of proposals of varying quality created by varying groups. In a typical situation, adding a barrier to use of the protocol would decrease the likelihood that the protocol would be used over others that lack such a barrier. However, in this instance, adding a high cost to proposal creation would improve the quality of the platform to a degree that typical participants would prefer to use the platform when such a barrier is in place.

As the protocol was described, when an Executor reveals a prediction they send a file containing their prediction to the contract, which then distributes this file to the Funders in some way. This step is one of the most computationally expensive and dangerous. For instance, such files could be quite large, especially if they contain images, and it would be difficult to ensure that such files do not also contain malware. Although at this time we do not have an elegant solution for this problem, one simple solution would be that after the contract verifies the hash of the file, the contract only distributes a link to the network, directing to a website exhibiting the full contents of the prediction in a way that the Funders can be sure of their security, modulo any zero-day exploits in their web-browsers for instance.

Similarly, as defined, the Designers of a proposal are to describe a task, which could be quite lengthy. However, posting the description of a task on a website which the proposal/contract then links to is not an option, because the description ought to be immutable after a proposal is published. Hence, an alternative is to present the file in some way so that version history can be tracked and authenticity can be cryptographically verified. For instance, the smart contract can reveal a link to the task description along with a hash of the description for use in ensuring the description has not been changed.

4 Applications

We conclude this paper by identifying a number of important applications our protocol enables.

The obvious application: atomizing the Grim Reaper

Pursuant to Bell's Assassination Politics, the protocol could indeed enable the creation of an assassination marketplace. However, given the incentives of both Funders and Executors, such a marketplace – if it existed – would, fortunately, look nothing like that envisioned by Bell or others, where anyone can easily become a target. Indeed, if contracts were activated with the task of killing someone, that person would need to be a widely hated person many would pay to see killed. To this end, such targets would likely need to be clear violators of human rights to be successfully targeted.

If a contract was activated tasking Executors with killing a John Doe, then in order for the contract to be executed the RewardThreshold parameter would need to be quite low, and the number of Funders quite low. This would imply that those funding such a contract would likely know John Doe personally, and because of the high risk that an Executor would not be paid in full, it is also likely that such an Executor would have killed the John Doe independent of any contract.

However, the primary concern of the authors is "meme killings." That is, we are concerned about internet memes that lead to the creation and activation of contracts with large rewards tasking Executors with assassinating an undeserving person. Fortunately, we believe the solution to this would be constructing the protocol atop a blockchain that does not provide strong anonymity guarantees, like Ethereum. If the protocol was implemented atop such a blockchain, then participants could preserve a basic level of anonymity (pseudonymity) that ensures their activity is safeguarded against the general public (in the typical case), and yet allows police to track and arrest those funding or creating proposals/contracts detailing criminal tasks.

Even so, if the protocol was designed in such a way to provide strong anonymity guarantees for all users, it is possible that "evil" contracts would seldom gain notable funding due to the fact that it would require many people to spend money on highly unethical tasks that are unlikely to provide them with any benefit.

Nonetheless, one can argue that the ability of the protocol to enable an assassination marketplace is precisely what makes it so important. For instance, if such a technology were available during World War II, would a contract tasking Executors with killing Adolf Hitler not be activated in days, and with a dramatically large reward? And would such a contract not have a high probability of success given the number of detractors in Hitler's regime? Moreover, in such a situation it is conceivable that the Allied powers would simply "look the other way," ignoring the crime their citizens may be performing by hiring an assassin to kill an enemy leader (even if contributors could be traced). Such behavior may even be

encouraged. As of this writing, one could envision a similar scenario in targeting Vladimir Putin.

Incentive Politics

With the most morally dubious application out of the way, we concern ourselves with the use of the protocol in giving humanity the ability to incentivize their governments to enact legislation they would not have passed otherwise. In particular, one can think of the protocol as providing The People with their own lobby, incentivizing legislators to act in a way supported by a large populous rather than a corporate lobbying body.

Consider climate legislation. Stringent climate laws are widely popular, and it would not be difficult to find hundreds of millions of people around the world willing to contribute hundreds if not thousands of U.S. dollars to many different proposals/contracts tasking legislators with passing various climate policies. In return for passing such policies, participating legislators could have reward funds directed into their campaign funds for instance.

More broadly, large groups of people could leverage the protocol to initiate and continually fund large infrastructure projects, lawsuits, or many other projects. Indeed, in many – if not most cases – it would be ideal to fund a company to accomplish some task, instead of legislators. In such cases, the use of commitment schemes as described in Section 2 are of great importance: using such schemes, a company who "wins a contract" can be sure that they will likely be paid should they complete the project as specified, and will not have to worry about the funding being split amongst many competing groups. And, so long as the contract is designed properly, the Funders of such a contract can be sure that the corporation carrying out the task will do so as they should (i.e. within budget and on time), for if they do not they will not be paid.

Counterterrorism and counternarcotics

An additional use of the protocol could be in targeting the leaders of terrorist or drug and human trafficking groups. If properly managed, the protocol could practically ensure a persistent disorder if not total destruction of such groups. Consider a situation in which a contract is activated tasking Executors with revealing the location of (or killing) a leading figure in a terrorist or drug trafficking group. Supposing such a task was completed, an identical contract could be activated shortly after, ad infinum (assuming sufficient financial support). As a consequence, the mere existence of such a system would provide a strong deterrence to running or partaking in such organizations, abolishing them.

In particular, we speculate such a system would be effective in targeting widely hated leaders of such groups. For example, if such a system were available after the September 11th attacks, it is likely that the American people would have funded incredibly large contracts to see the leaders of such attacks rapidly put to justice. After all, with a sufficiently

large reward in place, even the most devoted fanatics would be greatly tested.

Of course, it is well known that many countries such as the U.S., France, or China, provide rewards to people who come forth with actionable knowledge about the whereabouts of certain criminals or terrorists for instance. However, any person in a position to do so would much prefer to act anonymously or pseudo-anonymously through the protocol instead of contacting the government directly, which carries a far greater risk. And, if such contracts manage to garner a large amount of support, then the reward such an individual could reap would be far higher than even the largest rewards governments put up.

An end to conventional war?

Following the protocol's application to deconstructing criminal organizations, as well as its potential use in developing assassination marketplaces, it becomes natural to ask how the protocol would be leveraged during war. Indeed, it is likely that opposing sides would promptly fund contracts tasking the assassination of opposing leadership. But the extent to which the contracts would succeed depends both on the magnitude of the rewards, and the loyalty of the leadership's entourage.

While in any war leadership must take basic precautions to prevent assassination, if the assassin is guaranteed an enormous reward, the pool of possible assassins could be so overwhelming that the chances of the leadership's long-term survival would be scant. To this end, the existence of such marketplaces acts as a serious preventative measure in starting a war. In fact, the problem would extend well beyond governmental leadership: any leader of any war-critical entity would become a potential target. After all, unless leaders can be certain of the loyalty of their entourage, and even those several rungs below them, a rational leader would not support a war for fear of their likely assassination.

5 Conclusion

If the protocol is ever implemented and widely adopted, there is no doubt it will be abused. In fact, in the extreme case, it could fundamentally rewrite the rules of how society functions. Despite this potential (or certainty), it is also true that the protocol could provide humanity with an incredible gift. After all, such a technology could provide humanity with a "nuclear option" that acts less like a bomb and more like a scalpel.

Of course, our treatment of the negatives of building such a protocol is wanting. Yet this is largely why we have chosen to share this work: to start a conversation around if such a protocol ought to be built, and what to keep in mind when doing so. In Jim Bell's essay, he stated that assassination marketplaces are "as inevitable as gunpowder." While there is some truth to this, Bell's proposal is so extreme that it is unlikely it will ever be adopted totally, but something like the protocol given here very well could be. While it is not without its issues, the ability for people to organize and incentivize participants to accomplish some common goal is, in our belief, a truly remarkable prospect.

e9756d04ff1599baecd06389860331c195c916d0e765474459b9a0abe74e302e
3b22fe7de5eb2bdd79cc1b4a2100c894101453ca05e5c5430a256e66b93cd355

References

[1] T. C. May, "The cyphernomicon." https://nakamotoinstitute.org/literature/cyphernomicon/.

[2] T. C. May, *Crypto anarchy and virtual communities*. Timothy C. May, 1994.

[3] J. Bell, "Assassination politics." http://www.outpost-of-freedom.com/jimbellap.htm, 1995-6.

[4] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 283–295, 2016.

[5] "Darkleaks." https://github.com/darkwallet/darkleaks.

[6] J. Cox, "Darkmarket, the decentralized answer to silk road, is about more than just drugs." https://www.vice.com/en/article/9akagz/darkmarket-the-decentralized-answer-to-silk-road-is-about-more-than-just-drugs.

[7] D. Robitzski, "The rise of an "assassination marketplace" shows the dark side of decentralized networks." https://futurism.com/augur-assassination-marketplace-decentralized-blockchain.

[8] D. Oberhaus, "Assassination markets for jeff bezos, betty white, and donald trump are on the blockchain." https://www.vice.com/en/article/gy35mx/ethereum-assassination-market-augur.

[9] B. Merchant, "How anonymous crowd-funders put a 75,000 dollar price on ben bernanke's head." https://www.vice.com/en/article/wnje3z/online-assassination-markets.

[10] A. Greenberg, "Meet the 'assassination market' creator who's crowdfunding murder with bitcoins." https://www.forbes.com/sites/andygreenberg/2013/11/18/meet-the-assassination-market-creator-whos-crowdfunding-murder-with-bitcoins/.