Lite Paper 1.4
A decentralized dead man switch

Application to bequeath crypto assets.

J.D. Bertron

April 2022

1. Introduction

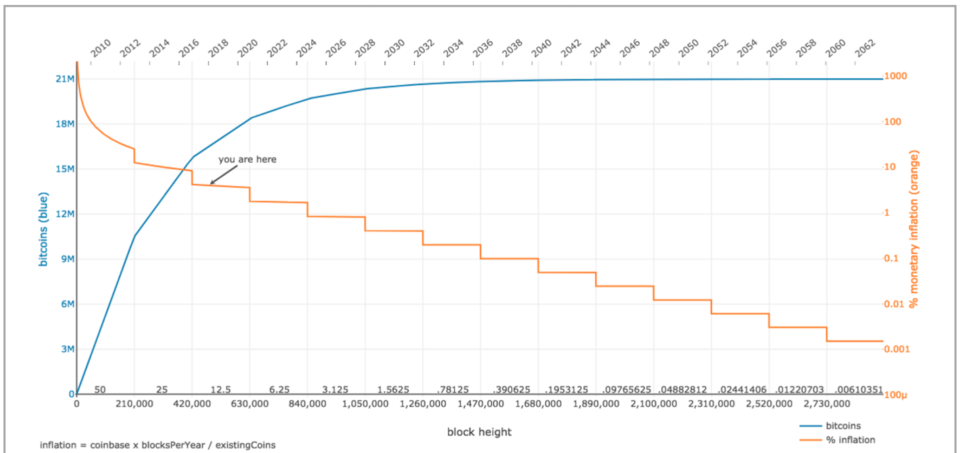
In this paper, we introduce a new method for implementing a blockchain based Decentralized Dead Man Switch that allows for a piece of information, known only to its creator, to be released freely to anyone upon their death. The method leverages a smart contract on the Ethereum blockchain to maintain and publish a time lock puzzle that can be replaced at any time by the user. The puzzle, when solved, will allow anyone to decrypt the information it conceals. The system requires a minimal setup from the user, with the help of an open-source Web3 application. It also requires funding of the contract instance to incentivize puzzle solving by disinterested parties.
The paper is organized as follows: The Market Need section discusses the types of applications this system can enable and provides insight into the various ways it can be used. In section Previous Work we discuss previous attempts to provide similar functionality. The Model section discusses the overall architecture of the system and the external components involved. In Preliminaries, we go over terminology and fundamentals that will be used in the paper. Subsection NuCypher in particular, discusses one of the major external dependencies of this system, as well as subsection ChainLink VRF. Finally the Implementation section discusses in detail how the system functions. A short discussion then precedes the Conclusion, in which the business opportunities are laid out.

2. Market Need

The search for a dead man switch technology has a long history. Recently it has become more important than ever to find one that can act as a canary for whistleblowers, protect people in oppressive regimes who want to secure information that might deter someone from murdering them or suppressing their work, or simply allow someone's life-long secrets to be revealed upon their death.

A market study can probably reveal the price people might be willing to pay to avoid trusting a third party with their crypto currency assets and the complexity of setup they would endure to put in place multi-signature wallets. Several companies have emerged to answer this demand, for example https://endowl.com/ or https://fortknoxster.com/ . These companies have not published details of their approach, and it is unlikely they have solved the problem of decentralized trustless inheritance.

Additionally, with cryptocurrency asset ownership on the rise, it is legitimate to question what the value of these crypto assets will be if a large portion of them remain dormant, lost forever, should their owner die. For example, before year 2030, the number of new Bitcoin blocks mined per year will dwarf significantly the number of bitcoins already mined, thus if a significant portion of these assets are lost when their owners die, the overall quantity of Bitcoins contributing to liquidity will shrink - an effective deflation of the supply.



Bitcoin supply schedule (courtesy of http://bashco.github.io/)

Yet in this scenario, one must wonder what will remain of the incentives to use the currency as a store of value absent the possibility to bequeath those assets in a decentralized, uncensorable and disintermediated fashion.

In its simplest form, the Decentralized Dead Man Switch simply allows information to be encrypted and published until some work is performed to unseal the secret. It should be evident that although a large secret can be locked in this manner, it is sufficient for most purposes to simply lock a small secret which can grant access to a much larger store of information. In recent history, for example, software businessman John McAfee claimed to have information implicating corporations and politicians, protected by a Dead Man Switch, yet the information was never found. It is also clear that other famous whistleblowers[1] might have benefited from such a technology. One can also speculate whether journalists might be looking for such a technology to prevent suppression of their investigations[2].

In her popular book, “Crypto Asset Inheritance Planning”, Pamela Morgan warns that relying solely on a smart contract based mechanism is ill advised, and we agree. A digital dead man switch is simply a new, storage location option to secure data into the future.

3. Previous Work

The following review is not chronological. A simple attempt to create a smart contract that can lock-up a certain amount of ERC20 tokens has been attempted by Mike Goldin, the creator of Token Curated Registries (TCR). The ERC20 tokens are locked in the contract, only allowing the user access to them. The concept is simple, but does not allow much secrecy about the amount of tokens stored, and in fact does not allow any secret to be locked. Also a beneficiary must be designated in advance, which introduces a vulnerability. The project is a proof of concept that could scale for multiple users but has not been developed beyond that stage. Ronak Doshi and others proposed Whistle, a Web3 App designed to assist Whistleblowers, during the 2018 Eth-India hackathon. The application is leveraging the smart contract to store NuCypher policy aliases per users, but it is unclear how the information is protected from early decryption, or if the decryption flow has been properly implemented.

John Wineman, a participant at ETHDenver 2020 contributed a Solidity smart contract designed to provide censorship resistant attestation named Living-proof. The contract functions as a canary and is only meant to keep track of a user’s last invocation of the contract. If the last proof of life is older than a certain number of block intervals, any external user is allowed to invalidate it. The contract has a concept of a reward it calls a Pinata, for whoever invalidates the last proof.

On the commercial side, a Bitcoin Inheritance solution has been advertised by Casa. The system is a legal escrow service offered that leverages up to 3 out of 5 multisignature control of a Bitcoin Wallet (a less expensive variation uses 2 out of 3 multisig control). There is no support for other blockchain assets or for secret keeping.

On the research side, several interesting solutions have been proposed. In Paralysis Proofs, Zhang, Daian and Bentov propose to use a smart contract to restore multi-signature access to a bitcoin wallet when one member of a multisignature group goes missing. Participants can challenge one of the group members to prove they are ‘alive’, causing a special participant that runs on an Intel SGX trusted computing platform, to issue two new transactions that will be signed by SGX. One transaction is small while the other is designed to be invalid until a later time. This is supported by newer P2SH Bitcoin OpCodes CLVT and CVT. The small transaction can be spent immediately by the party that is challenged and that transaction invalidates the other. The system has limitations we would prefer to avoid. First the system only supports Bitcoin. Second it requires a trusted implementation to run on Intel SGX which holds all the keys as a trusted third party.

Another solution was proposed by Seres, Shlomovitz and Tiwari and named CryptoWills. The

[1] Julian Assange, Chelsea Manning, Seth Rich, Gary Webb, Edward Snowden.
[2] WikiLeaks famously published an ‘Insurance File’ which has never been decrypted.

system leverages a Trusted Execution Environment (TEE) similar to SGX to perform computation of a Time Lock Puzzle to release the TEE from needing an external source of time. Use of Multisignature Bitcoin instructions then allows release of Bitcoins to beneficiaries.

Finally a service named Sarcophagus.io, claims to have created a decentralized dead man switch for the purpose of providing inheritance of crypto assets. While convincing and much better designed than other solutions above, the system still relies on a single server 'archeologist' having control over the decryption of the payload. Besides the vulnerability this creates if the 'archeologist' could be compromised, or coerced, it is unclear what technological barriers exist to prevent early decryption. The Lite Paper discusses at length the cute terminology choices that help convey the concepts, but does not touch on the subjects of time keeping, trust relationships, forward secrecy, or resilience. The system also relies on bounties awarded in the form of an ERC20 Token which does not seem necessary. In the next section, we will set out the goals of our system and provide an overview of our approach, which avoids all the pitfalls of previous solutions.

4. Model

In the real world a Dead Man Switch is a device that constantly monitors the presence of stimulus from a person, in order to leave the state of a system intact. You can see them on power tools or treadmills. Often in movies it goes beyond the simple kill-switch application, in which the activation may cause a third party to regret interfering. The result may be an explosion, or the release of embarrassing documents.

In the digital world, everything can be copied however, and it is difficult to imagine how forward secrecy[3] can be maintained for secrets, long after private keys may have been revealed. Because blockchains are public ledgers, it is also unreasonable to expect a secret to be 'kept' locked inside a contract, since any curious miner or nation state is sure to be looking for it there.
Getting an accurate sense of time is also a challenge, which is exacerbated by the blockchain environment since every miner may have their own local time. This has been mitigated by the use of block-height, in some smart contracts and supported by blockchain opcodes, to prevent miners from cheating. We will first establish the security goals that have been considered, then provide a high level overview of the operation of the Dead Man Switch.

4.1. Security Goals

4.1.1. Decentralized, no Trusted Third Party

The worst time to find out a third party could not be trusted is when they are needed to secure the release of documents that protect your life. This is our most important security requirement. Decentralized systems tend to exhibit this property by preventing unwarranted updates, preventing the suppression of updates, etc. In other words, state transitions of a system are inevitable, predictable, unalterable and consistent.

4.1.2. Forward Secrecy

Forward secrecy is a cryptographic property that refers to the inability of anyone in possession of past encrypted messages to decrypt them, even after entering in possession of the private keys. In this context, this means we want to make sure whatever solution to an older puzzle will not allow someone to decrypt a current secret.

[3] In cryptography, forward secrecy is the assurance that session keys will not be compromised even if long-term secrets used in the session key exchange are themselves compromised.

4.1.3. Time integrity

Rather than rely on the blockchain for time keeping, which may be reliable for transaction locking, we need to remove the possibility of an offline attack, by which a miner could create a fake branch of the blockchain, under which the unlocking conditions could happen, simply based on block numbers. In Bitcoin, the CVLT opcode attempts to make sure blocks have not simply been added to the blockchain but also verified. This is sufficient to prevent one miner lying to another about the validity of a transaction, but it is not sufficient for ensuring the safety of a secret which once read by the miner, is no longer secured. We will leverage Time Lock puzzles for this purpose.

4.1.4. Revocable

The set-up of a contract instance should be revocable. This is for a few reasons, one being the forward security mentioned above, but also for plausible deniability purposes. Notice that this requirement invites the possibility of a 5-dollar-wrench attack on the user, but of course there is nothing that can be done about that. The revocability of the contract should be something that can be configured once and cannot be changed by the user later. (Thus, an irrevocable revocability setting.)

4.1.5. Uncensorable

Revealing a secret can be dangerous. If history is a guide, a powerful entity such as a nation state will attempt to suppress the release of information if it can. We want to provide a simple way that only the user can delay the release of information. Decentralizing the storage of information will play a key role for the user in ensuring the information cannot be suppressed before it is decrypted.

4.1.6. Affordable

The cost of instantiating such a system should be low. Indeed, if access to resources was made difficult, more powerful entities could easily starve the system and prevent its use by poorer individuals. This is therefore also a security concern for the system.

4.1.7. Low profile (few attack vectors)

The system should present few attack surfaces to prevent its use or jeopardize its stability. At this point, the only major vulnerability will reside with the user's initial set-up and device. Beyond the initial set-up, fewer vulnerabilities will be found as the system will no longer rely on sensitive communications, random number generation, or sensitive mathematical computation. By pushing the vulnerability to the initial set-up, it becomes possible to make the choice of device and its environment much more secure and for this to have a much bigger pay-off in terms of the overall security.

4.2. Overview

At a very high level, the Dead Man Switch consists of setting up a Time Lock Puzzle, whose solution is only known to the user immediately after its creation, yet cannot be discovered by others until some amount of time has expired. The user can leverage the puzzle's solution to secure the re-encryption of the secret into a set of private/public keys that is not yet known. This is accomplished using a Proxy-ReEncryption (PRE) service. Ordinarily, the user can "**flip the hourglass**" - i.e. restart a new puzzle and by setting a new Proxy Re-Encryption policy. The Proxy Re-Encryption service will guarantee that only a new set of private/public keys corresponding to the new puzzle solution can be allowed to decrypt the secret. By design, the Proxy Re-Encryption service does not require trust in a third party and meets all of our security requirements.

But if the user does not renew the puzzle in time, the Proxy Re-Encryption service will provide re-encryption of the secret when someone uses the puzzle solution to request access. A system of incentives provides rewards for individuals who dedicate CPU cycles to solving puzzles. The rewards must be funded ahead of time and constitute the only cost of using the Dead Man Switch, other than Ethereum gas fees.

4.3. Preliminaries

The Decentralized Dead Man Switch system makes use of several common cryptographic primitives. Since Blockchain applications routinely use Hash functions, signatures and Elliptic Curve point operations, we will only focus here on some of the unusual mathematics involved in our system.

4.3.1. Time Lock Puzzles

Time Lock Puzzles (TLP) are a very active area of research in the cryptographic community. Variations on the subject take the name of Verifiable Delay Function (VDF) or Delay Encryption (DE). Interest in this primitive has grown from the need to force participants to wait in online decentralized auction systems. The verifiability aspect refers to the ability of a participant to provide a commitment to an intermediate result without revealing anything else. The very first instance of a time lock puzzle was proposed by Rivest, Shamir and Wagner (RSW 1996) and is also known as the LCS35 puzzle. In this original version, given a large integer $N = p \cdot q$ the product of two secret large safe primes the puzzle consisted of computing the value $y = x^{2^t} \pmod N$ given a random value x .
For anyone without the knowledge of the factorization of N , this can only be done by repeated modular squaring. Knowledge of the factorization of N provides a trap-door for computing the solution in mere milliseconds. The parameter t provides control of the puzzle difficulty.

4.3.2. Verification and Rewards

Recent research papers by Wesolowski and Pietrzak pioneered ways to provide proof that a RSW style puzzle has been solved. The problem they were trying to solve was that of proving that the result of puzzle computation is correct, more efficiently than simply re-doing the work and in environments in which one cannot afford to reveal the factorization of the modulus.
In the case of our Dead Man Switch, the result of solving a puzzle is used by possibly two different people. First there is the puzzle solver claiming a reward for publishing the solution. Note that claiming the reward must be a two-step process, (see section Commit & Reveal below)in which proof of completion as a commitment is provided to 'lock-out' any other claimants, then providing the solution that solves the puzzle and matches the commitment. [4] Second, there is the Proxy-ReEncryption (see next section below) client who needs to use the puzzle solution to decrypt the secrets.

There is also value in forcing puzzle solvers to show their work, to provide evidence of their progress. First, economically, it can let puzzle solvers self-organize around the work that needs done. Second, because preventing premature puzzle solving will be important for the Decentralized Dead Man Switch system, the ability to monitor how fast the community of puzzle solvers can perform repeated modular squarings will be necessary.

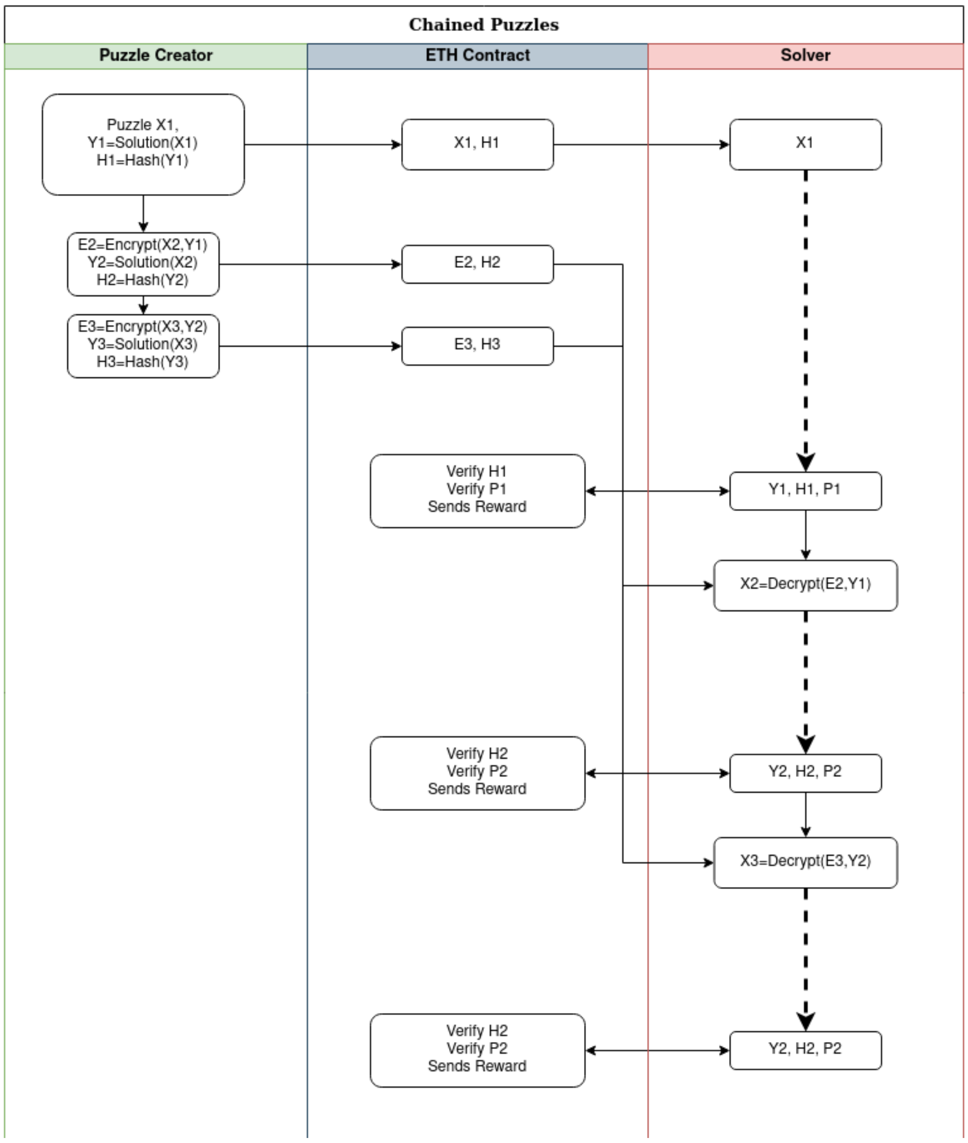
We have implemented Python and Solidity code for the Pietrzak VDFs proof generation and verification, and have confirmed that it is efficient and affordable in a smart contract environment.

[4] Simply providing the solution 'in the clear' could expose the puzzle solver to cheating from other participants, and simply granting the reward on the submission of the proof alone could not guarantee that the solution would be made public. This method is explored in the Chained Puzzles section.

4.3.3. Chained Puzzles

One way to allow puzzle solvers to check-in their work for rewards is to simply create multiple, shorter puzzles. For a year's worth of puzzles, for example, four distinct puzzles could be generated, lasting three months and chained, in such a way that the solution to the first one is necessary to start solving the next one, yet, the puzzle solver can still check-in a proof that they solved the puzzle. This method does not allow for a market in partial puzzles to develop, unless the number of intermediate puzzles is large. This also implies that the storage on the blockchain might be much higher, a definite drawback. This approach might be appropriate for cases when the puzzle solving is assured by contract with BqETH for example, and intermediate puzzles are only necessary to provide sanity checks on the work. It follows that the puzzle reward might be lower as well, making this method more appealing to whistleblowers.

For this set-up, n random inputs x_i for n puzzle challenges are generated, and then their solutions $y_i = x_i^{2^t} mod\ N$ are computed using the trapdoor. Then, a chain is set-up between the *n* results: the solution y_1 of the first puzzle is used as a key to encrypt the second challenge x_2 and so on. The initial challenge x_1 can be released, along with the *n* − 1 *encrypted* challenges. Since the encryption for each challenge is uncrackable, and there's no way to "jump forward" in repeated modular squaring, there is no way to reach the final result faster than a single puzzle of the same length. This technique can allow various puzzle difficulties (time parameter) to be combined, if the time parameters are provided along with the puzzle challenges. The benefit is of course to provide finer controls over the overall puzzle length, but also to gain visibility into how puzzle solving is tracking with the initial estimate. Shorter chained puzzles will also provide a disincentive for puzzle farmers to wait too long before claiming their rewards.


Chained Puzzles
Puzzle Creator
ETH Contract
Solver
Puzzle X1, Y1=Solution(X1) H1=Hash(Y1)
X1, H1
X1
E2=Encrypt(X2,Y1) Y2=Solution(X2) H2=Hash(Y2)
E2, H2
E3=Encrypt(X3,Y2) Y3=Solution(X3) H3=Hash(Y3)
E3, H3
Verify H1 Verify P1 Sends Reward
Y1, H1, P1
X2=Decrypt(E2,Y1)
Verify H2 Verify P2 Sends Reward
Y2, H2, P2
X3=Decrypt(E3,Y2)
Verify H2 Verify P2 Sends Reward
Y2, H2, P2

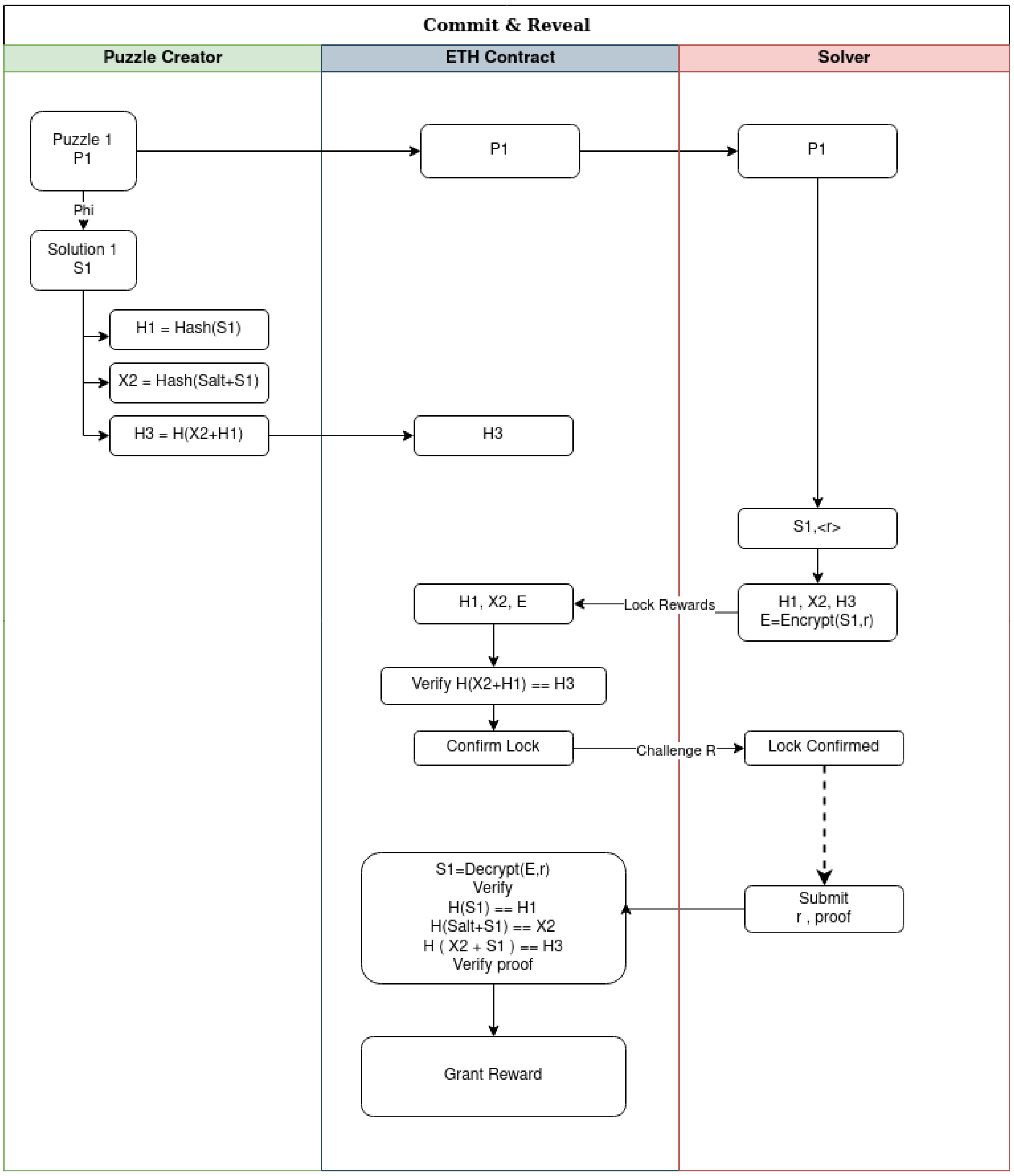
4.3.4. Commit and Reveal

Because puzzle farmers must receive the rewards for their work without allowing an enterprising Ethereum miner to substitute their own address, we must provide a way for them to prove they are the legitimate solver of the puzzle, before providing the solution and its proof.

The way this is typically implemented is as follows. The puzzle creator initializes and publishes puzzle $P_1 : X_1^{(2^t)}$ from some source of randomness to generate X1, and then calculates the solution $S1 = X_1^{2^t}$. To generate the next puzzle P_2, a salted hash of the solution can be used for the value $X_2 = Hash(Salt + S1)$. The salt is a public parameter. The next puzzle P_3 can be generated the same way. In addition the puzzle creator computes the hash of the solution S_1 as $H_1 = Hash(S_1)$, and then the hash of the combination of these results $H_3 = Hash(Hash(Salt + S_1) + Hash(S_1)) = Hash(X_2 + H_1)$ is calculated. Finally, H_3 is submitted to the blockchain contract as a commit condition for releasing the reward.

Once a puzzle solver has spent time computing the solution S_1, it is a simple matter for them to compute H_1, X_2 and H_3. To claim the reward, the puzzle solver must then first submit $X_2 + H_1$ and draw a random value r to encrypt S_1, to send as E to the blockchain contract to prove they have found a value that hashes to H_3 and to lock the reward to their address.

Notice this does not reveal anything useful to competing puzzle solvers, and does not allow a malicious miner to substitute their address to claim the reward. It also prevents a lucky guess of the value $X_2 + H_1$ from being able to claim the reward without providing proof that they know S_1 later on. Finally the puzzle solver can submit the value r, which can be used by the contract to first decrypt S_1 from E and then confirm that the puzzle solver whose initial submission of H_3 is indeed the one deserving the reward, by computing H_1, X_2, H_3, allowing a new transaction to credit the puzzle solver with the reward.



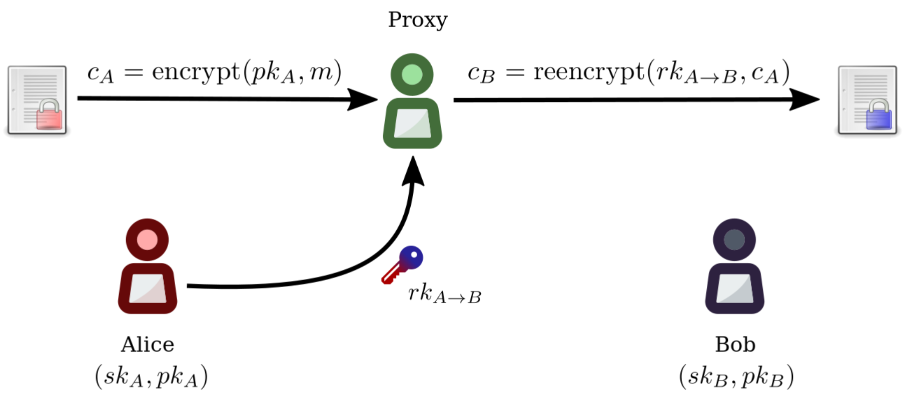
Because the puzzle solver does not have to submit S_1 immediately after submitting H_3, they can gain a head start on all other puzzle solvers who will learn of X_2 as the reward is claimed. Of course, if they wait too long, they may be preempted by a different puzzle solver with the correct solution. A malicious Ethereum node has no way to 'grab' values from the requests to claim the puzzle solution for themselves. They do not know S1 and therefore cannot manufacture a value of E that will yield S_1 until the legitimate puzzle farmer has submitted r to the network allowing the contract, and anyone listening including the malicious miner, to decrypt S_1 from E.

4.3.5. NuCypher

NuCypher is a Decentralised Proxy-ReEncryption service. The concept of Proxy-ReEncryption (PRE) is shown below, with a diagram from NuCypher's white paper.

Information encrypted for Alice using her public key pk_A can be re-encrypted by a Proxy agent into a version that can only be decrypted by Bob, using his secret key sk_B .
There are three important things to remember about Umbral, the fundamental building block for NuCypher:

- The Proxy can only re-encrypt if it is given some re-encryption key $rk_{A\to B}$ by Alice.
- The Proxy is never in possession of a decrypted version of the document.
- Alice must have access to Bob's public key pk_B .


Proxy
$c_A = \text{encrypt}(pk_A, m)$
$c_B = \text{reencrypt}(rk_{A\to B}, c_A)$
$rk_{A\to B}$
Alice
(sk_A, pk_A)
Bob
(sk_B, pk_B)

In the NuCypher decentralized system, the Proxy is actually a network of independent entities - referred to as Ursulas, who will store a piece of the threshold (m of n) re-encryption key $rk_{A\to B}$ for a period of time specified by Alice. The relationship is described by a PRE policy, that specifies the duration, threshold parameters and of course the participant Bob from his public key. The policy can be revoked or will expire.

Because the network of Ursulas is comprised of nodes worldwide who stake a significant amount of NuCypher tokens to earn more tokens by performing accurate work, as well as because they stand to lose their stake if they fail, forget, lie or collude, the system can be trusted to perform this function extremely well. The Ursula nodes must be paid with ETH, in direct proportion to the number of nodes required by the threshold parameters, and in proportion to the number of days in the policy. The Decentralized Dead Man Switch will leverage this framework to bind the solution of a puzzle with a PRE policy.[5]

[5] At the time of this writing, NuCypher is undergoing a merge with Keep to become the first ever blockchain merger of two projects, under the name of Threshold Network. The PRE system will remain in place, with minor improvements.

4.3.6. ChainLink VRF

The initial set-up of the Decentralized Dead Man Switch will require numerous random numbers to be generated, to initialize parameters such as Elliptic Curve points for private keys and composite modulus generation from random primes, as well as an initial puzzle input. Beyond this initial set-up however, a new random number will be necessary every time the puzzle is replaced, as the proof of life is being provided. Because random number generation has historically been a point of vulnerability, especially on devices that may be compromised, we have decided to leverage the on-chain random number generation provided by ChainLink's Verifiable Random Function .

This operation is costly, but will provide the certainty that a random number is truly unpredictable. Further transformation of the random number will allow the Decentralized Dead Man Switch to formulate a puzzle that cannot be predicted in advance. This is important as the threat of puzzle prediction is at least as important as the threat of modulus factorization, given that the puzzles may involve long time periods.

4.3.7. IPFS

We anticipate that the storage of secrets will be relegated to the Inter Planetary File Systems (IPFS) or similar decentralized storage. It is outside the scope of this paper to discuss IPFS in detail, but one important thing must be mentioned: IPFS replication is not guaranteed. Documents never leave the user's device until another user requests the document by its hash. Therefore, provisioning of such replication will need to be planned before the guarantee can be made that a secret can be accessed post mortem. There too, a prearranged commitment can be made with BqETH to guarantee some level of replication.

5. Implementation

At the core of the system, an Ethereum smart contract will manage the publishing of puzzles, their configuration including the required intermediate checkpoints and difficulty, the allocation of puzzle rewards, verification of solutions and unlocking of rewards, as well as storage of auxiliary information such as IPFS links.
A Web3 application, capable of linking to wallets such as Metamask will provide the interface for users to perform the initial puzzle creation, as well as the set up of public parameters for their instance of the contract, including funding of puzzle rewards. Because at the moment, funding of NuCypher policy management can only be done by communicating directly with Ursula nodes, and with the NuCypher Policy Manager contract, the application will also need to communicate with the NuCypher network, as well as the Polygon network for payment. [6]
We will refer to a user's Ethereum address as their account, while referring to their active invocation of the contract as their instance.

5.1. User level secrets

At the core of a user's puzzle generation is the prime composite N, product of two safe primes. From the analysis of VDF research papers, a minimum security parameter of 112[7] will be used, implying that N should be at least 2048 bits. Recent research recommendations relating to the generation of large prime numbers in adversarial environments, will be included in the Web3 application so there can be no leakage of the factorization.

[6] Version 6.0 of NuCypher's PRE introduces a Level 2 payment rail on the Polygon network, which allows for cheaper policy creation. Further improvements include the Porter proxy layer to decouple applications who do not want to connect directly to the Ursula nodes.
[7] From the NIST recommendation. Section $5.6.1
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
Also https://www.keylength.com/en/4/ .

The factorization of N can be discarded after the computation of the totient factor ϕ has been calculated, encrypted and saved along with N in the public parameters. This will ensure the user application can always re-use the trap door to the solutions without requiring any subsequent storage.

5.2. Secret Encryption

We have not yet discussed the primary purpose of the Decentralized Dead Man Switch, which is to secure a secret. This secret, a short testament for example, a phrase allowing the decryption of more secrets stored on a decentralized file system, or a token whose hash can unlock a Bitcoin P2SH transaction[8], will be encrypted using the NuCypher policy's public key, which is associated with the user's Ethereum wallet. The Ethereum contract will provide limited storage for such a secret and enough to also provide a location for larger documents. In this encrypted state, only the user will be able to decrypt the secret, using his or her NuCypher private key.

5.3. Puzzles

5.3.1. Generation

As explained in Preliminaries, puzzle generation will involve picking a random seed s, which will allow the creation of the first puzzle. The initial puzzle generation is assumed to occur under controlled conditions such that the randomness of the first value is not a concern. The next puzzle however, might be generated under less ideal or friendly circumstances. For this purpose, a setting in the contract and in the application can leverage the ChainLink Verifiable Random Function (VRF), a random beacon, or alternatively the Keep random beacon facility, to issue a public, yet unpredictable random value. Details of the puzzle generation are provided in our White Paper.

5.3.2. Difficulty Adjustment

Standard difficulty values could be adjusted in the published contract for common time frames, such as 1 month, 3 months, 1 year, so that a standard value can be 'picked up' by the application from the blockchain with no interaction with BqETH. The parameter **t** will be adjustable by each user within their instance, and in the contract by BqETH, with the constraint that it can only be adjusted upward by BqETH. This is to avoid the possibility that a State actor could pressure the BqETH to shorten the life of the next puzzle for users using default values. With this framework, if the puzzle solving community finds faster ways to solve puzzles, the expected solving time can be maintained for every user's next puzzle. For more drastic changes in solving speed, notifications to "flip the hourglass" - i.e. restarting a new puzzle and setting a new policy - sooner than expected can be arranged with BqETH.

5.3.3. Puzzle Verification

Puzzle verification will proceed as described in the Preliminaries section. There is little more to add here. The contract will hold a temporary 'lock' on any Ethereum address which successfully submits the correct X2+H1 combination that Hashes to H3. The puzzle solver will then be able to wait as long as they wish to claim their reward. However, knowing someone else who might have completed the work could still claim the reward will motivate every puzzle solver to claim theirs as soon as possible.

[8] Careful planning will play a large role to ensure such transactions are not accidentally invalidated by the user by spending the UTXOs linked to the P2SH transactions. Similar

5.4. Proxy ReEncryption Policy

5.4.1. Network Access

During the initial set-up and every time the puzzle is replaced, in addition to a web3 call to an Ethereum network provider, the application needs to access the NuCypher network. In the current architecture, this network will provide Ursula nodes from whom the application can learn from other nodes. Once this is properly established, a request to issue a policy can be issued, and accepted by a number of Ursula nodes. Invocation

At the core, a simple Proxy-ReEncryption policy will be requested and issued by the NuCypher network. The application will not issue an Ethereum network request to publish and activate a new puzzle until this is complete. The choice of PRE threshold parameters m and n will be left to the user with advice from BqETH on how to select them.

5.4.2. Revocation

When the user decides to ‘flip-the-hourglass’ and create a new puzzle, after the new policy has been issued and before the new puzzle is published to the contract, the obsolete policy will be revoked. A certain buffer of time will be factored into each policy such that enough time is allocated to account for a lack of incentives in puzzle solving that make its expected solving time greater than average, but not so much that it is paying for Ursulas to keep secrets active past the expected length of the puzzle. The buffer size and reasonable limits will be built in the application to prevent major problems.

5.5. Reward Management

5.5.1. Reward Escrow

We expect that the Ethereum contract will be holding ETH tokens in escrow on behalf of its users, to be dispensed for puzzle rewards in the amount specified by users. Monitoring will alert BqETH of accounts whose escrow amount is running low so that it can provide an alert service to its clients. Functions in the contract’s API will allow users to contribute to the escrow fund of any instance. This means a user wishing to remain anonymous can obfuscate the origin of their funds. Withdrawing funds from the escrow will of course be restricted to the original owner of the instance and to funds not already allocated to active puzzles.[9]

5.5.2. Reward Allocation

Reward allocation is initially set by the user on a per-puzzle basis. Depending on the configuration, as advised by BqETH, some users may choose to reward puzzle farmers in proportion to their demonstrated work, or not at all. In all cases, the remaining unclaimed reward amount will be dispensed to whoever can prove they have arrived at the last puzzle solution. A discussion of whether a puzzle reward should be adjustable should lead one to conclude it should not. It should be obvious that the mere ability of adjusting it lower would destroy the incentives of puzzle solvers. Likewise the ability to adjust it higher is only a binary incentive: if the puzzle is already being solved, it does not affect the expected solving time by much[10], and if the puzzle is not actively worked on, it does not provide assurance it will. In both these cases, the user can leave the reward for long-tail

[9] Once the user has flipped the hourglass and published a new puzzle, older puzzles remain active until their reward has been claimed, yet they can no longer provide access to proxy re-encryption.
[10] Some very smart people are working on ways to make repeated squarings much faster using ASICs, creating a ‘solving gap’ between off the shelf computers with spare cycles and dedicated hardware. Higher rewards may only affect the decision to use such hardware for some puzzle farmers.

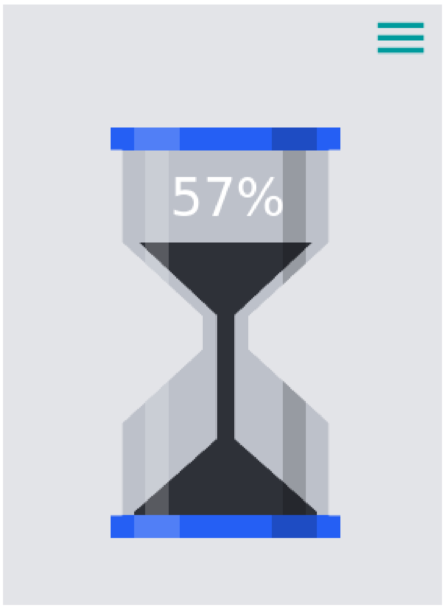
puzzle farmers who will dedicate old and slow hardware to collect the reward, while simply ‘flipping the hourglass’ and spending a little extra on the next puzzle reward.

5.6. Interaction

5.6.1. Web3 Application

We envision a simple Web3 application for mobile devices, allowing a user to check on the status of their current puzzle and the date by which they are expected to ‘flip the hourglass’. With a desktop application available for in person training and setup, more secure hardware may be employed so that the initial prime number generation is more secure, reliable and free of interference.
All applications will be open-source and audited by BqETH.

The user application, because security is very important, will feature a minimal interface allowing very few customizations and exposing very few parameters. Its function will be to simply provide a status so the user can check progress toward solving their last active puzzle, an estimated date by which the puzzle should be switched (by flipping the hourglass) and perhaps few other notifications for clients of the monitoring services.

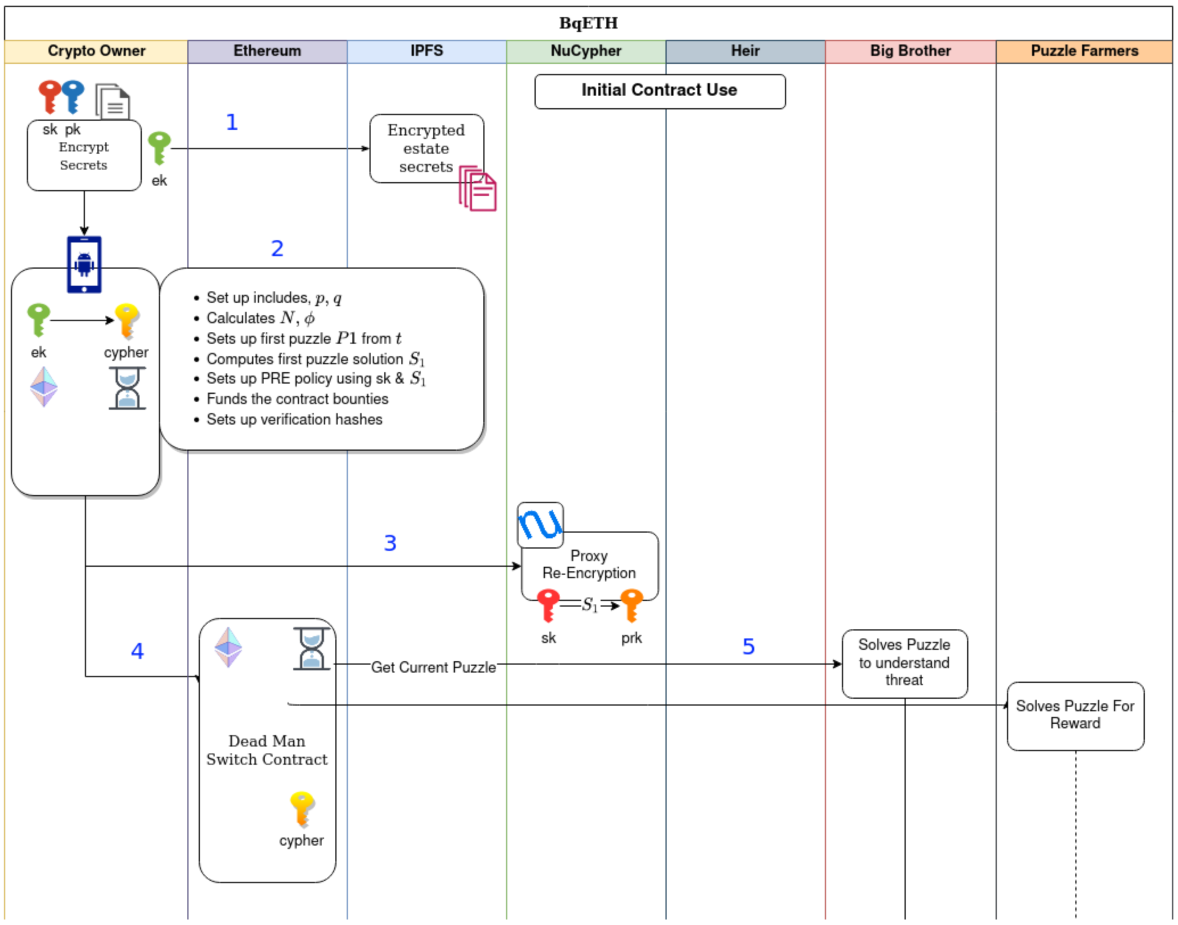


5.6.2. Wallet Integrations

Since funding of the Puzzle Reward Escrow of each contract may be performed from any address, an application integration with a standard software or hardware wallet will be important to allow a smooth operation of the contract invocation, NuCypher network interactions and reward funding.

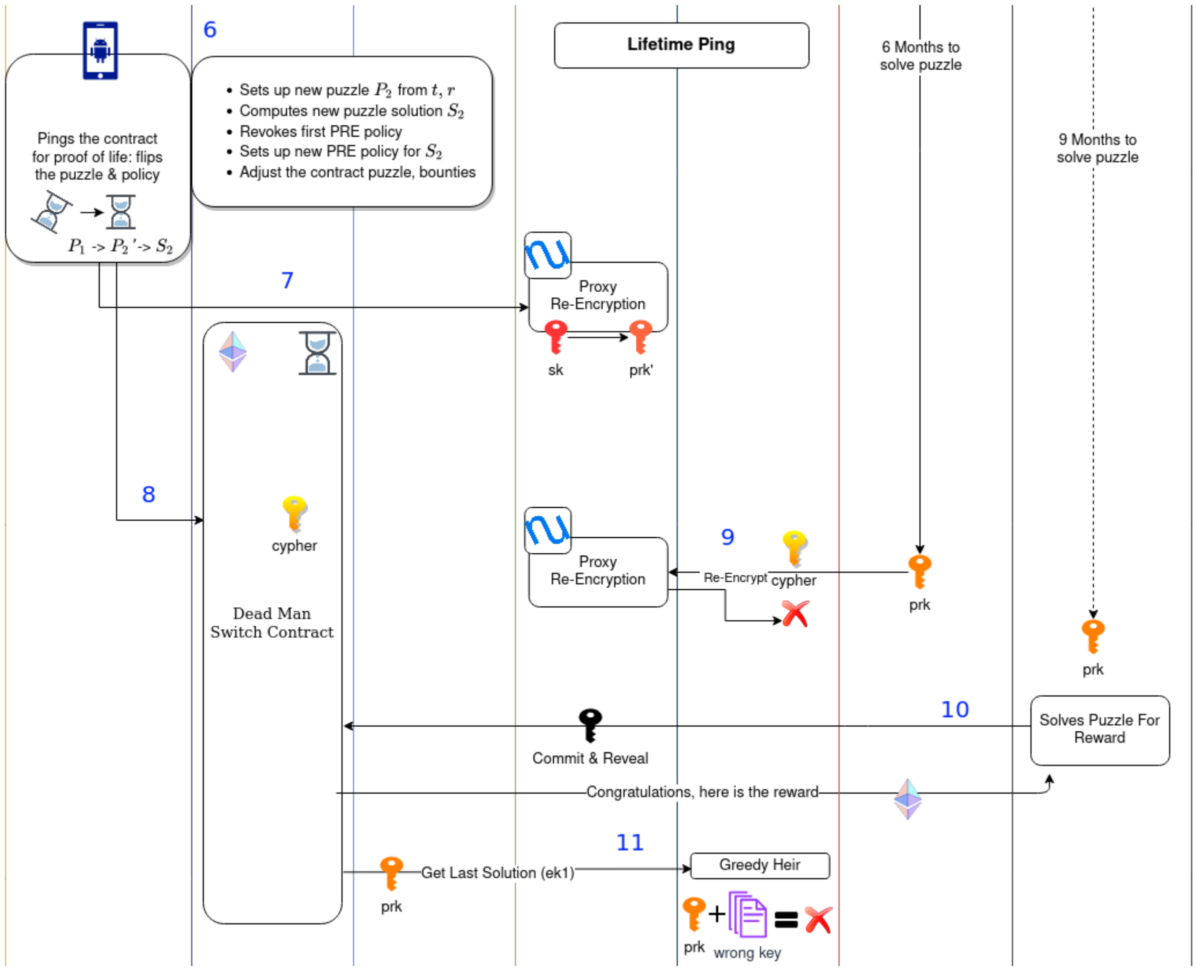
5.7. Workflow

This section gives a visual representation of the workflow and interactions between the Application, the Ethereum Contract, the NuCypher network and Puzzle Farmers. Using the blue numbers in the pictures below, we can walk through the various moving parts and their roles:



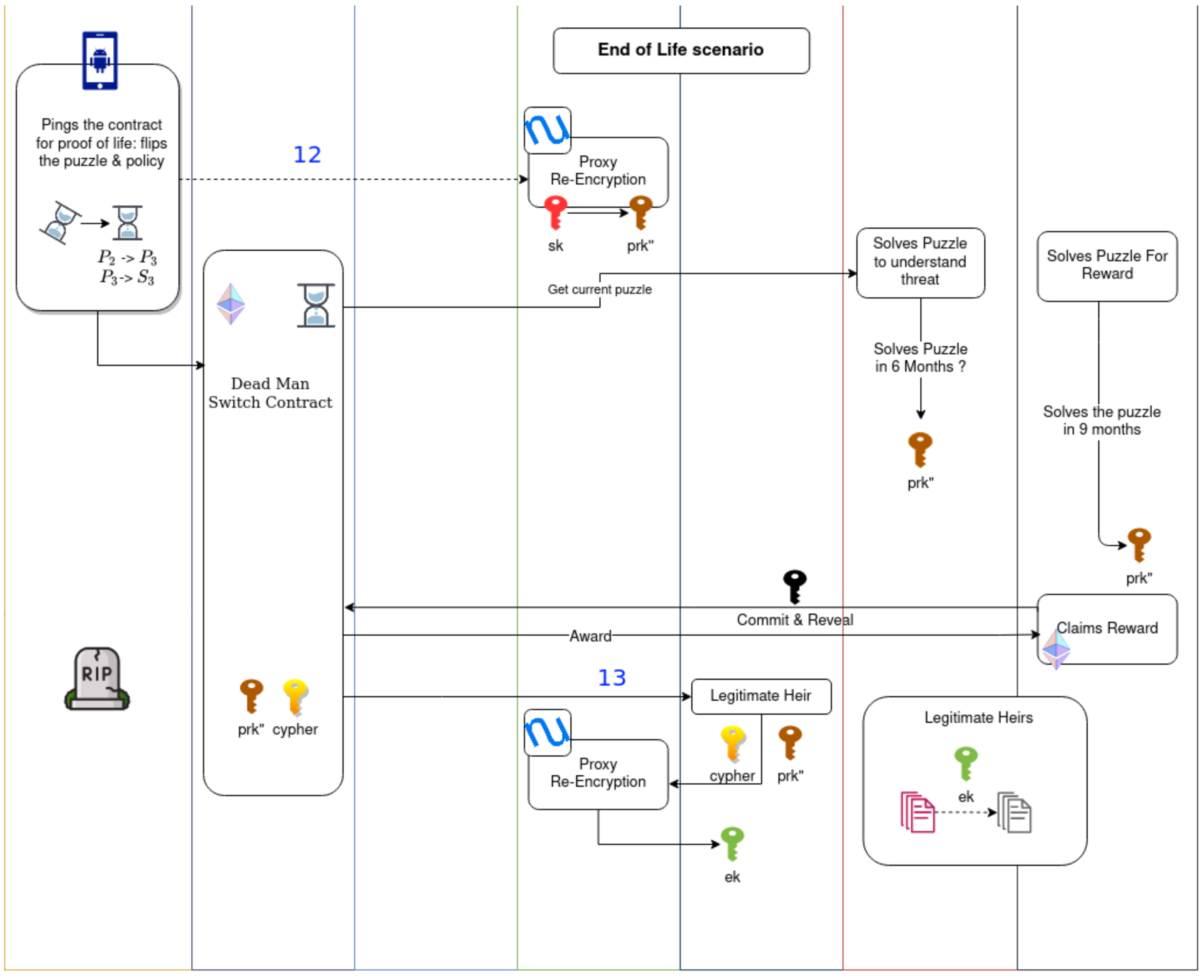
1. In the initial setup phase, the user has a secret key (sk) and public key (pk) pair that will allow them to use the Ethereum contract. Some symmetric encryption key (ek) is chosen by the user to encrypt their secrets.
2. Using their device, the user will pick random large safe primes p,q to obtain N and ϕ. They will also pick a random number to create the first puzzle P_1. The encryption key ek is also encrypted using the user's private key pk, which yields a cypher (message-kit). The Puzzle solution S_1 is also calculated privately.
3. From the application, connectivity with the NuCypher Network is initiated and a PRE policy is set up to allow re-encryption into a public key defined by S_1. Because the policy is linked to the puzzle solution, effectively no-one can currently request re-encryption of the cypher until the puzzle is solved.
4. The final setup step is for the user to call the Ethereum contract and publish the puzzle challenge, along with the rewards for it.
5. The puzzle is then ready for solving and anyone can request the parameters of the challenge.

In the next picture, we examine what happens during the lifetime of this contract, and the workflow that takes place to 'flip the hourglass', substituting both puzzle and policy.



6. In this step a new puzzle P_2 is created and its solution S_2 is calculated.
7. The NuCypher network is once again contacted, to cancel the old re-encryption policy as well as to establish a new one, allowing re-encryption from sk to prk' defined by S_2.
8. The new puzzle P_2 is published to the Ethereum contract as the new 'active' puzzle.
9. This flow illustrates that the NuCypher Network will at this point deny re-encryption to anyone presenting the correct credentials prk, derived from the old puzzle solution S_1.
10. Solving the older puzzle still allows a puzzle farmer to contact the Ethereum contract to claim their Ethereum reward.
11. The puzzle solution does not help a greedy heir to obtain access to the documents.

The final illustration shows what happens when the user passes.



12. Assuming the last puzzle P_3 has been properly generated, its solution S_3 has been used to generate a policy and interested parties are actively working on solving the Time Lock Puzzle.
13. Because the user dies and is unable to ‘flip the hourglass’ one last time, some puzzle solver will be able to claim the puzzle reward and in doing so, reveal the puzzle solution. At this point anyone can access the policy and request of NuCypher that the cypher be re-encrypted into the public key, defined by the puzzle solution S_3. Thus in this case, the Proxy-ReEncryption will re-encrypt the cypher for key prk’’ allowing anyone to decrypt the symmetric key ek.

Clearly, for most secrets, other information will be used as blinding factors to only allow certain participants to access the secrets. For example, instead of encrypting the secrets with ek, a private salt only known to the intended recipients could be used, thus preventing everyone from decrypting them.

5.8. Monitoring

This section discusses things that should be monitored, by BqETH or others. Many parameters, suggested by the application and perhaps enforced by the contract will vary. A compromised app may choose to ignore such parameters and therefore some reasonable limits will need to be implemented in the contract. For example, we can force the contract to only increase its estimates of the values of **t** necessary to create 3-month puzzles, and so on for longer durations. We know this because the computing power is not expected to ever fall.

5.8.1. Puzzle Farming

Puzzle farming will need monitoring in several ways. First and obviously, the ‘best of breed’ solving speed will be available from the expected duration of puzzles compared with the actual check-in times. Another important metric will involve the expected check-in times against the actual check-in time of puzzle solvers. This may provide insight into how much computing power is made available and switched in order to discourage other puzzle farmers. A recent paper suggests such repeated squaring markets may become decentralized as well, providing better insight into both costs and expected performance at a global level.

5.8.2. Expiration Notifications

When a puzzle is nearing completion and if the user is alive, it becomes important to ‘flip the hourglass’ to protect the secrets. Another option is to simply cancel the PRE policy without re-generating the puzzle, effectively canceling the contract. Finally a more drastic option of course is to rug-pull the information that is being protected. In all cases however, the Ethereum blockchain, as a public ledger, will provide insight into how many and which of these puzzle contracts face imminent solving. Such cases can be handled by BqETH for its customers, and attempt to locate or notify them. As a service, BqETH can assist in recovering the secrets locked by the policy, before it expires, for the benefit of heirs and interested parties.

5.8.3. NuCypher

Monitoring of the health of the NuCypher network will also become important if more resources must be dedicated to providing a reliable service. Also, monitoring of the NuCypher policy parameters **m** and **n** will inform future users on how to balance their security with the costs associated with setting up the policy.

5.8.4. Reward pricing

Another metric that may prove useful is the amount of ‘actively worked puzzles’ so that advice can be produced to reduce the number of outpriced puzzles that generate little interest from puzzle farmers. Monitoring the number of puzzles actively being solved and their rewards will provide insights into the supply and demand curves of the ecosystem, to advise users on what rewards are appropriate for each length of time. The prices will depend on the supply and demand of puzzles as well as the opportunity costs of tying up a CPU Core, and of course, the value of the rewards against other crypto currencies.

5.8.5. Escrow Alerts

Monitoring the escrow balances will also provide insights that will be useful to BqETH to alert its customers who need to fund their contract. Extracted from publicly available information on the Ethereum blockchain this information will be available for any user, including anonymous users.

5.8.6. PRE Policy Mismatches

The two edge cases detailed in the improvement sections will also lead to monitoring of how often puzzles expire before the PRE expires as well as how often a PRE policy expires without the puzzle being solved. The product of this monitoring will inform users and BqETH how to adjust certain buffer parameters between the puzzle expected expiration and the policy expiration.

5.8.7. IPFS Replication Gaps

Because some users will want to store lots of information, in sizes that do not belong on the blockchain, both for cost and forward security reasons, a free form field will be provided in the contract for a link to information stored on IPFS, FileCoin or similar service.

BqETH can effectively monitor the existence of replicated versions of the encrypted files, for some of its customers. In addition, it can detect that replication gaps have occurred and notify their customer or prevent such gaps from occurring by replicating them further. As discussed in other sections, there is a case to be made for the information to be located in various geographies.

5.8.8. BqETH Puzzle Farming

We envision BqETH will embrace Puzzle Farming as an open and profitable source of revenue for itself, at least until a competitive market develops. For its clients, BqETH may offer the guarantee that a puzzle is always being solved, to ensure their heirs will always be able to access the decrypted information. This offer can also come with the assurance that the puzzle will be solved in locations around the world that are appropriate to the sensitivity and maturity of the puzzle. In the event that the puzzle is solved by BqETH, an arrangement to refund a portion of the puzzle reward toward the escrow may be a possibility.

6. Conclusion

There are several business opportunities that arise from this work. First is the creation of a reliable and decentralized system for a digital deadman's switch.
Consulting and teaching opportunities exist in the setup phase of the dead man switch use. This includes the business opportunity of inheritance planning for use of the system. It also includes the coaching of heirs in the use of the system post mortem. A substantial source of revenue can also come from Estate Attorneys' continuing education requirements.

A second business opportunity consists of providing services to clients wishing to be notified of expiring puzzles, reward escrow levels, suggested set-up parameters, file replication, puzzle solving guarantees and other details relevant to their particular set-up.

Finally, some revenue may come from the collection of puzzle rewards, at least initially, since BqETH will have an interest in engaging in puzzle farming to provide the system with stability and security until external actors figure out it is profitable.

In order to make these business opportunities accessible, even though the software will of necessity be largely open source, the new company will offer classes both in group settings and to solitary students (in person or online) and consultations to make sure the setup is properly implemented.

7. References

1. Casa: Comprehensive Bitcoin Inheritance
2. Goldin, Mike. Dead Man Switch https://github.com/skmgoldin/dead-mans-switch
3. Ronak, Doshi Whistle https://github.com/Ronak-59/Whistle-dApp
4. Seres, Shlomovits and Tiwari: CryptoWills. https://eprint.iacr.org/2020/283.pdf
5. Whineman, John: Living Proof https://github.com/jwineman/livingproof
6. Zhang, Daian, Bentov. Paralysis Proofs. https://eprint.iacr.org/2018/096.pdf
7. Rivest, Shamir, Wagner. Time-lock Puzzles and Timed-release Crypto (1996)
8. Pietrzak. Simple Verifiable Delay Functions (2018)
9. Boneh, Bunz, Fisch. A Survey of Two Verifiable Delay Functions (2018)
10. Attias, Vignery, Dimitrov. Implementation Study of Two Verifiable Delay Functions (2020)