White Paper 2.0
A Decentralized Dead Man Switch

For Digital Asset Inheritance.

J.D. Bertron

November 2023

1. Introduction **3**
2. Market Need **3**
3. Previous Work **4**
4. Model **5**
4.1. Security Goals 5
4.1.1. Non Custodial 5
4.1.2. Decentralized, no Trusted Third Party 6
4.1.3. Forward Secrecy 6
4.1.4. Time integrity 6
4.1.5. Revocable 6
4.1.6. Uncensorable 6
4.1.7. Affordable 6
4.1.8. Safe 6
4.1.9. Private 7
4.1.10. Low profile 7
4.2. Overview 7
4.3. Preliminaries 7
4.3.1. Time Lock Puzzles 7
4.3.2. Verification and Rewards 8
4.3.3. Chained Puzzles 8
4.3.4. NuCypher 9
4.3.5. IPFS 10
5. Implementation **10**
5.1. User level secrets 11
5.2. Main Secret Encryption 11
5.3. Puzzles 11
5.3.1. Generation 11
5.3.2. Difficulty Adjustment 11
5.3.3. Puzzle Verification 12
5.3.4. Commit and Reveal 13
5.4. Condition Based Threshold Decryption 15
5.4.1. Network Access 15
5.4.2. Condition 15
5.4.3. Renewal 15
5.4.4. Revocation 15
5.5. Reward Management 15
5.5.1. Reward Storage 15
5.5.2. Reward Allocation 16
5.5.3. Decryption Reward 16
5.6. Interaction 16
5.6.1. Web3 Application 16
5.6.2. Wallet Integrations 17
5.7. Workflow 17
5.7.1. Setup 17
5.7.2. Recurring Interaction 18
5.7.3. Secret Replacement 19
5.7.4. Delivery Replacement 19

5.7.5. Notification Replacement 19
5.7.6. End of Life 19
5.8. Monitoring 20
5.8.1. Puzzle Farming 20
5.8.2. Expiration Notifications 20
5.8.3. NuCypher 20
5.8.4. Reward pricing 21
5.8.5. IPFS Replication Gaps 21
5.8.6. BqETH Puzzle Farming 21
5.8.7. Test Mode 21
5.9. Stability 21
5.9.1. Farming 21
5.9.2. NuCypher 22
6. Improvements and Future work 22
6.1. Quantum resistance 22
6.2. Biometric Device Integrations 22
6.3. Wallet Abstraction Integration 23
6.4. Wallet Tech 23
7. Conclusion 23
8. References 23
9. Appendix 25
Euler Totient Function’s Trapdoor. 25
ElGamal Encryption 25

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.

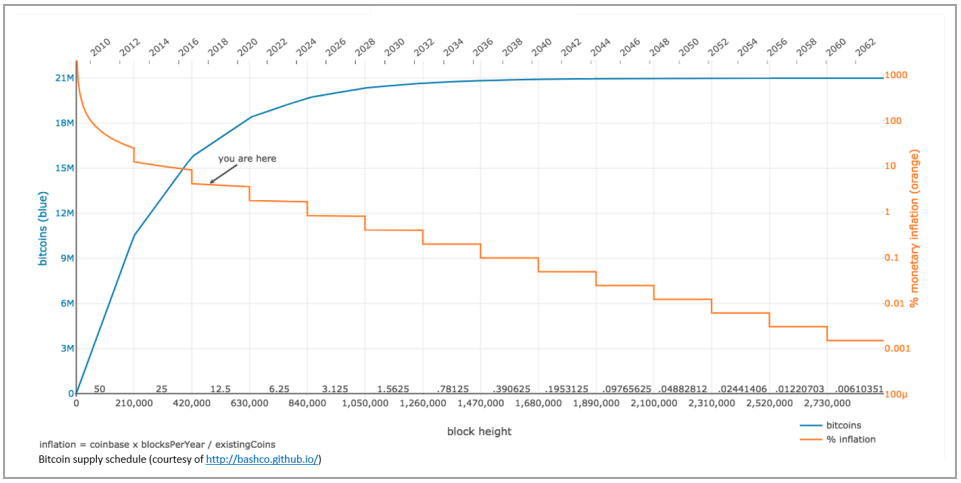
The paper is organized as follows: The Market Need section discusses the types of applications this system can enable and provides insight into the various ways it can be used. In section Previous Work we discuss previous attempts to provide similar functionality. The Model section discusses the overall 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. Finally the Implementation section discusses in detail how the system functions. A short discussion of Improvements and Future work then precedes the Conclusion and Appendix, which contains some of the mathematical proofs for reference.

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 crypto currency asset ownership on the rise, it is legitimate to wonder what would happen if a large portion of them will 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.



In this scenario, what will remain of the incentives to use the currency as a store of value, what would happen to society and the world economy, 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 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.

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, which must be part of a broader strategy.

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 EthIndia 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 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 CVLT 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 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.

[1] Julian Assange, Chelsea Manning, Seth Rich, Gary Webb, Edward Snowden.

On the commercial side, a Bitcoin Inheritance solution has been advertised by Casa. The system is a legal escrow service 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. Casa announced support for Ethereum in 2023. Other companies, such as FortKnoxter, Vault12, SafeHaven, SerenityShield and BeQuest have all proposed solutions for crypto asset inheritance leveraging multisignature, social recovery and at times the use of block-time to gate access, all with some kind of centralization of control.

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 much better designed than other solutions above, the system relies on multiple server nodes, 'archeologists' having control over the decryption of the payload and trusted to determine the conditions for decryption. Besides the vulnerability this creates if the 'archeologists' 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 whose long term value is questionable. Early in 2023, Sarcophagus changed their design to make use of Shamir Secret Sharing to remove the single node vulnerability.

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 exercise equipment. 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[2] 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 by chain spoofing or simulating a future time.

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. Non Custodial

BqETH does not want to hold your keys. We don't and any solution that does is a security risk, to you, to us and to your beneficiaries. It is also often a source of misunderstanding. So it is worth

[2] 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. In section Revocation we discuss our solution to this elusive problem.

repeating it. The BqETH blockchain contract does not hold, dispense, mix or otherwise control any of your crypto assets or keys. In fact we don't want to know what is locked up in your dead man switch because this makes us a risk.

4.1.2. 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, with blockchain technology, state transitions of a system are inevitable, predictable, unalterable and consistent.

4.1.3. 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 only the last secret a user encrypted should be decryptable.

4.1.4. 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 leverage Time Lock puzzles for this purpose.

4.1.5. 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.

4.1.6. 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 plays a key role for the user in ensuring the information cannot be suppressed before it is decrypted.

4.1.7. 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.

4.1.8. Safe

The safety of beneficiaries, recipients of the payload, has often been overlooked by previous solutions. Securing the payload while leaving the wallet identities of the people designated to receive them, for many years, in plain sight, or in the hands of vulnerable companies that can be pressured to reveal their identities, is a very significant threat. It should be obvious that with many years to prepare

an attack on an unsuspecting beneficiary, this can become the path of least effort in many threat models.

4.1.9. Private

Privacy is an important security goal which is often possible but difficult in many cases for Decentralized Applications. This goal should be to provide at least as much privacy as can be afforded by the blockchain, or better. Practically, this means that in view of the traceability of blockchain transactions on the Ethereum blockchain, no private information should be available in the contract state or transaction details.

4.1.10. Low profile

The system should present few attack surfaces to prevent its use or jeopardize its stability. In the Improvements section we will discuss areas of concern and obvious improvements that could be made in the future to further reduce the vulnerability of the set up. As with any such system, 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 in our system as it 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. Of crucial importance, as impossible as it may seem, we have ensured that no communication with BqETH is necessary for the operation of the Dead Man Switch, although, of course, for the user to benefit from the convenience of BqETH services, some amount of interaction will exist.

4.2. Overview

At a very high level, our Dead Man Switch consists of setting up a Time Lock Puzzle, whose solution is only known within the application immediately after its creation, yet cannot be discovered by others unless some linear computation is performed (i.e. some deterministic amount of time has passed).
We leverage the puzzle's unbreakable nature to secure the secret by letting the Blockchain contract to only allow its decryption once the puzzle has been solved. This is accomplished using a Condition Based Decryption (CBD), decentralized service.

Ordinarily, the user can "**flip the hourglass**" - i.e. restart a a new puzzle, that we represent as a Timer, delaying the possibility of decryption. The decentralized, CBD service will guarantee that the payload can be decrypted only when the contract approves it. By design, the Condition Based Decryption service does not require trust in a third party and meets all of our security requirements.

If the user does not renew the Time Lock Puzzle in time, the CBD service will be able to decrypt the last collection of secrets - which we call 'payload' authorized by the contract. A system of incentives provides rewards for individuals who dedicate CPU cycles to solving puzzles and submitting decryption requests and proof of decryption. The rewards must be funded ahead of time and constitute the dominant 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 out of 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. More recent developments have circled around the use of Supersingular Isogenies which secure the primitive against quantum attacks while making them easier to compute. The very first instance of a time lock puzzle was proposed by Rivest, Shamir and Wagner (RSW) 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 initial value x . For anyone without the knowledge of the factorization of N , this can only be done by repeated modular squaring. The parameter t provides control of the puzzle difficulty.
For the puzzle creator, the value y can be computed efficiently knowing Euler's Totient function $\phi(N) = (p-1)(q-1)$ by noticing that $y = x^{2^t} \pmod N = x^{2^t \pmod{\phi(N)}} \pmod N$. The details of this simplification can be found in the Appendix.

4.3.2. Verification and Rewards

Recent research papers by Wesolowski and Pietrzak pioneered ways to provide a succinct 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, in environments in which one cannot afford to reveal the factorization of the modulus. It is obviously desirable to spare the verifier from having to solve the puzzle themselves, especially in a decentralized environment, where the verifier is a smart contract.

In the case of our Dead Man Switch, the outcome of solving a puzzle is used by different actors. First there is the puzzle solver claiming a reward for publishing the solution. Note that claiming the reward must be a two-step process, 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. [3] Second, there are the Conditional Based Decryption nodes, who will query the blockchain contract to obtain permission to decrypt. Third, when a user's last puzzle is solved, a reward can also be claimed for providing proof of decryption by publishing the decrypted payload.

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 to be 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. The Stability section will discuss this dynamic in more detail.

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, twelve distinct puzzles could be

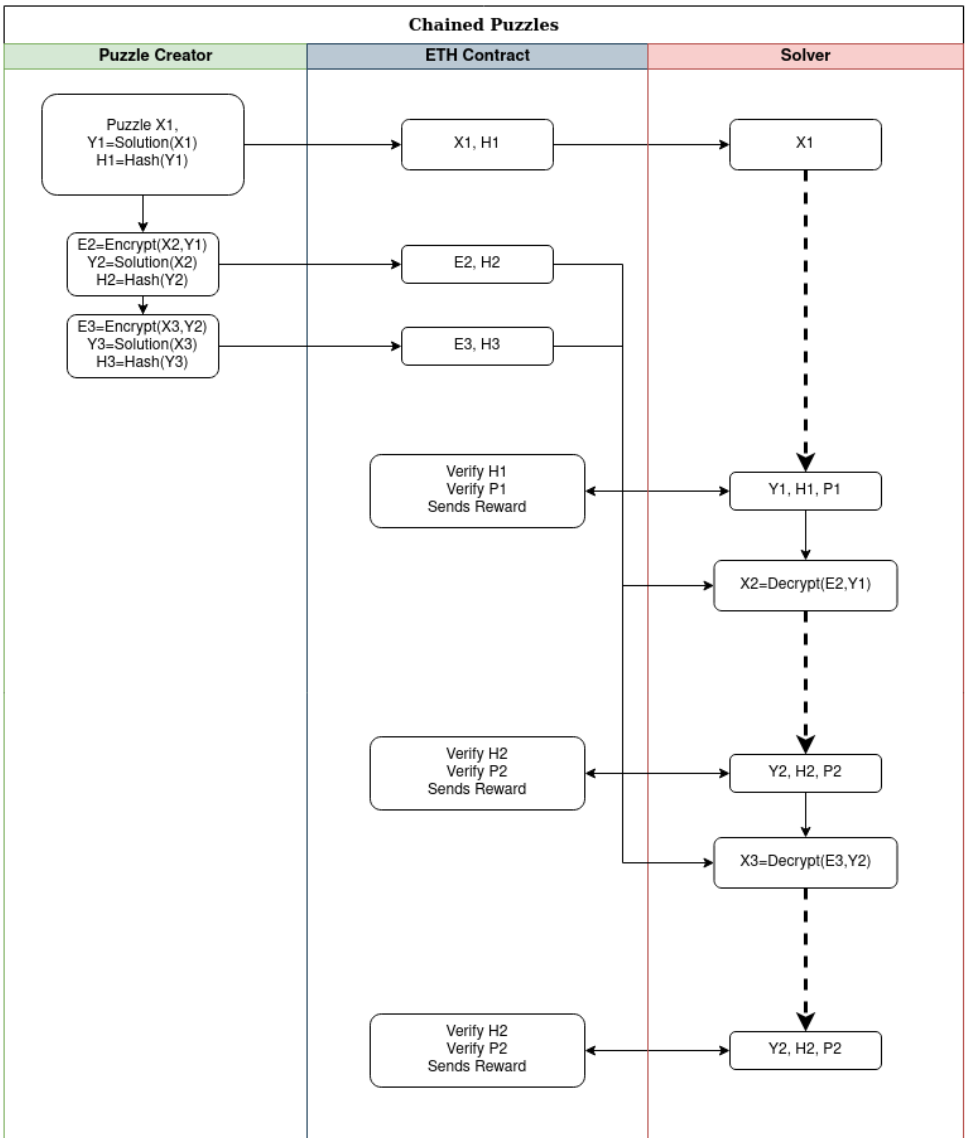
[3] Simply providing the solution 'in the clear' could expose the puzzle solver to cheating from other participants, and simply granting the reward on the submission of the proof alone could not guarantee that the solution would be made public. This method is explored in the Commit & Reveal Puzzles section.

generated, lasting a month each 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 even allows for a market in partial puzzles to develop. The extra storage on the blockchain is significantly higher but not prohibitive since a puzzle is entirely defined by its starting point and an exponent if all puzzles in a chain share the same modulus.

To set up a chain, 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 by solving each puzzle sequentially. This technique can allow various puzzle difficulties (time parameter) to be combined. 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 solve time estimate used at puzzle creation. Shorter chained puzzles will also provide a disincentive for puzzle farmers to wait too long before claiming their rewards.

In the current embodiment of the system, BqETh uses chains of 32 puzzles.



4.3.4. NuCypher

NuCypher is a Threshold Condition Based Decryption service, not associated in any way with BqETH. It is currently our selection for best-in-class for this function. The service is part of an array of cryptographic solutions revolving around Threshold technologies, and offered by a company of the same name. To avoid confusion, we will continue to refer to it as NuCypher.

The technology relies on several cryptographic primitives which are beyond the scope of this paper to explain: Distributed Key Generation, Shamir Secret Sharing and Threshold Decryption.

For details on this amazing technology we refer you to NuCypher's documentation. At a high level, NuCypher allows one party to initiate a decentralized ritual for key generation, resulting in multiple key shares being distributed among a set of specialized nodes, along with threshold parameters for its use. The generated key pair is shared among nodes using Shamir's Secret Shamir method, which prevents any one of the nodes from using their share. The threshold parameter dictates how many nodes are necessary to perform decryption and typically take the form of 2:3, meaning 2 out of 3. Encryption is similar to traditional public key cryptosystems and uses a single public key.

In contrast to other systems requiring multiple participants who must be trusted to 1) hold key shares, 2) not collude and 3) agree to perform decryption and 4) agree on the conditions for performing decryption, the NuCypher system relies on anonymous, decentralized, staked and incentivized nodes to perform the decryption.

As for the agreement surrounding the conditions for decryption, the NuCypher system allows such conditions to be programmed, along with the encrypted payload, such that decryption nodes can all evaluate the conditions independently. Originally designed to facilitate decryption based on conditions such as NFT ownership or other demonstrable attributes, the conditions can involve the evaluation of a call to a blockchain contract. We use this feature to allow our contract to authorize the decryption of payloads.

4.3.5. IPFS

The storage of large secrets, i.e., over a size considered affordable on the blockchain,if desired, is relegated to the Inter Planetary File Systems (IPFS). 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. Therefore a mechanism will exist in BqETH to allow participating nodes to provide the replication.

5. Implementation

At the core of the system, an Ethereum smart contract manages the publishing of puzzles, including the required intermediate checkpoints and difficulty, the allocation of puzzle rewards, verification of solutions and unlocking of rewards, as well as distribution of decryption rewards.
A Web3 application, capable of linking to Web3 wallets such as Metamask provides 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.
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 heart of a user's puzzle generation is the prime composite N, product of two safe primes. From the analysis of VDF papers, a minimum security parameter of 112[4] is used, implying that N should be at least 2048 bits. Recommendations from Albrecht, Massimo, Paterson and Somorovsky (13) relating to the generation of large prime numbers in adversarial environments, has been included in the Web3 application so there can be no leakage of the factorization. In particular, we leverage the Baillie PSW state of the art method for selecting large primes, which is a combination of Miller Rabin tests and Lucas tests and making sure both primes are *safe* primes by making sure $p \mod 4 \neq 3$, and $q \mod 4 \neq 3$.

To date, no false prime has ever been found to pass these two tests. In addition, recommendations from Blum, Blum & Shub have been selected to ensure the combination p & q generates a large enough cycle and make factoring N more difficult than solving the puzzle.

The factorization of N can be discarded after the computation of the totient factor ϕ has been used to calculate solutions to their puzzles. Once a user has set up their puzzle and encrypted their

[4] 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/ .

secret, all private information can be discarded and no user information storage is necessary on the part of the contract or application. This satisfies much of the privacy requirements laid out in the Security Goals.

5.2. Main Secret Encryption

We have not yet discussed the primary purpose of the Decentralized Dead Man Switch, which is to secure a secret. This payload, 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[5], will be encrypted using the Threshold public key, along with the Condition crafted to prevent the decryption until the user no longer has an unsolved or inactive (implying death) puzzle. . The Ethereum contract provides limited storage for such a payload or for the location of the payload on a decentralized file system.

In this encrypted state, the payload cannot be decrypted by anyone, not even the user.

5.3. Puzzles

5.3.1. Generation

As explained in Preliminaries, puzzle generation involves picking a random 2048 bit seed s, which will allow the creation of the first puzzle. Recall that the puzzle itself, published to the contract, is simply the root value $x_0 = 2^s \mod N$, for which the challenge is to compute $y = x_0^{2^t} \mod N = (2^s)^{2^t} \mod N = (2^{2^t})^s \mod N$ and which has solution $y = (2^{2^t \pmod \phi})^{s \pmod \phi} \mod N$ which is extremely fast to compute for the user, knowing the private values ϕ and s but very slow to compute for everyone else, as a sequence of modular squarings $y = (x_0)^{2^t} \mod N$.

We implemented cryptographic limitations to ensure the starting point belongs to the set of Quadratic Residues Modulo N. According to Blum Blum Shub, one of these measures is to ensure the seed is co-prime to both p and q.

The initial puzzle generation is assumed to occur under controlled conditions such that the randomness of the first value is not a concern. Our app does not require contact with BqETH, and only with the Ethereum Network when setting up the timer.

5.3.2. Difficulty Adjustment

The Application uses a value, published in the blockchain contract, for the fastest known solving speed, as a benchmark for generating new puzzles with a difficulty that guarantees a minimum solving time. Initially, BqETH will be responsible for updating this value, but we plan that the contract will be able to update the value on its own. This is to avoid the possibility that a State actor could pressure BqETH to shorten the life of the next puzzle for users using default values.

The value is read directly by the application from the blockchain with no interaction with BqETH necessary.

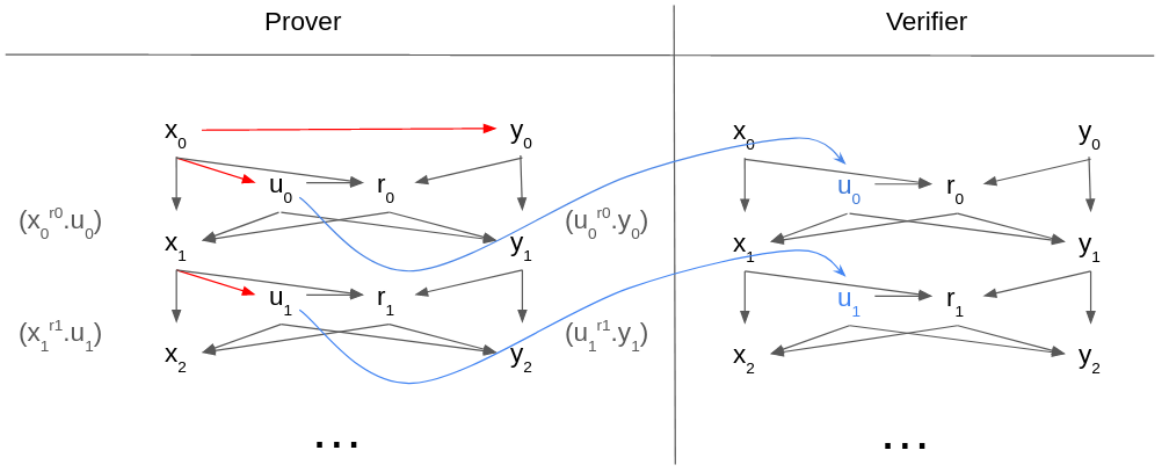
With this framework, if the puzzle solving community finds faster ways to solve puzzles, the expected solving time can be maintained for every user's next puzzle. For more drastic changes in solving speed, notifications to "flip the hourglass" - i.e. restarting a new puzzle - will be broadcast by BqETH, and can be performed by the user in the Application.

---

[5] The user must plan carefully to ensure such transactions are not accidentally invalidated by the user by spending the UTXOs linked to the P2SH transactions.

5.3.3. Puzzle Verification

Puzzle verification proceeds using Pietrzak's method mentioned in the Preliminaries section. Pietrzak's approach was to split the problem in two halves: $y = x^{2^t}$ can be re-written as $u = x^{2^{t/2}}$ and $y = u^{2^{t/2}}$. Combining the terms of these expressions on each side $u \cdot y = (x \cdot u)^{2^{t/2}} \pmod N$ allowed him to introduce the Fiat Shamir challenge r for a zero knowledge verification as $u^r \cdot y \pmod N = (x^r \cdot u)^{2^{t/2}} \pmod N$ which is a new statement of the form $y' = x'^{2t/2} \pmod N$ in need of verification but with half the difficulty of the first.
The method then proceeds by halving the exponent from until reaching 1, at which point the expression $y'' = x''^2 \pmod N$ is trivial to verify.
The method is diagrammed below, using index 0 for the original puzzle expression with challenge x_0 and solution y_0, introducing the midpoint u_0. Subsequent indices correspond to the expression in need of a proof at a new, smaller exponent: $y_i = x_i^{2t/2^i} \pmod N$



In the diagram above, substantial repeated modular squaring operations take place and are depicted

i	$\overline{x}'_i$	$\overline{\mu}_i$	$\overline{y}_i$
1	1	$2^{T/2}$	2^T
2	$r_1 + 2^{T/2}$	$r_1 \cdot 2^{T/4} + 2^{3T/4}$	$r_1 \cdot 2^{T/2} + 2^T$
3	$r_1 \cdot r_2 + r_2 \cdot 2^{T/2} + 2^{T/4} \cdot r_1 + 2^{3T/4}$	$r_1 \cdot r_2 \cdot 2^{T/8} + r_2 \cdot 2^{T5/8} + r_1 \cdot 2^{T3/8} + 2^{T7/8}$	$r_1 \cdot r_2 \cdot 2^{T/4} + r_2 \cdot 2^{3T/4} + r_1 \cdot 2^{T/2} + 2^T$
⋮	⋮	⋮	⋮

with red arrows. The u_i terms do not present significant extra work to be performed, as they are computed from combinations of powers of x_0. The figure below from Pietrzak's paper illustrates the sequence of powers required for each u_i in log_{x_0} basis, with $\mu_i = log_{x_0}(u_i)$.
For example this yields: $u_3 = (x_0^{t/8})^{r_2 r_1} (x_0^{3t/8})^{r_1} (x_0^{5t/8})^{r_2} (x_0^{7t/8})$.

Pietrzak's method spares the verifier the cost of calculating the terms u_i by sending them over as the proof $\pi = u_i$ and defining the terms r_i as $H(x_i + y_i + u_i)$ so the verifier can recreate them. This is illustrated above by the blue arrows in the above figure.
The contract holds a temporary 'credit' associated with any Ethereum address which successfully submits the correct X2+H1 combination that Hashes to H3. (see Commit and Reveal) The puzzle solver is then 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.

For use in the BqETH Application, code to generate puzzles and chain them, solve them and produce the Pietrzak proof, then verify the proof was implemented in Typescript, Python, GoLang, and Solidity and some of this code is available on GitHub.

5.3.4. Commit and Reveal

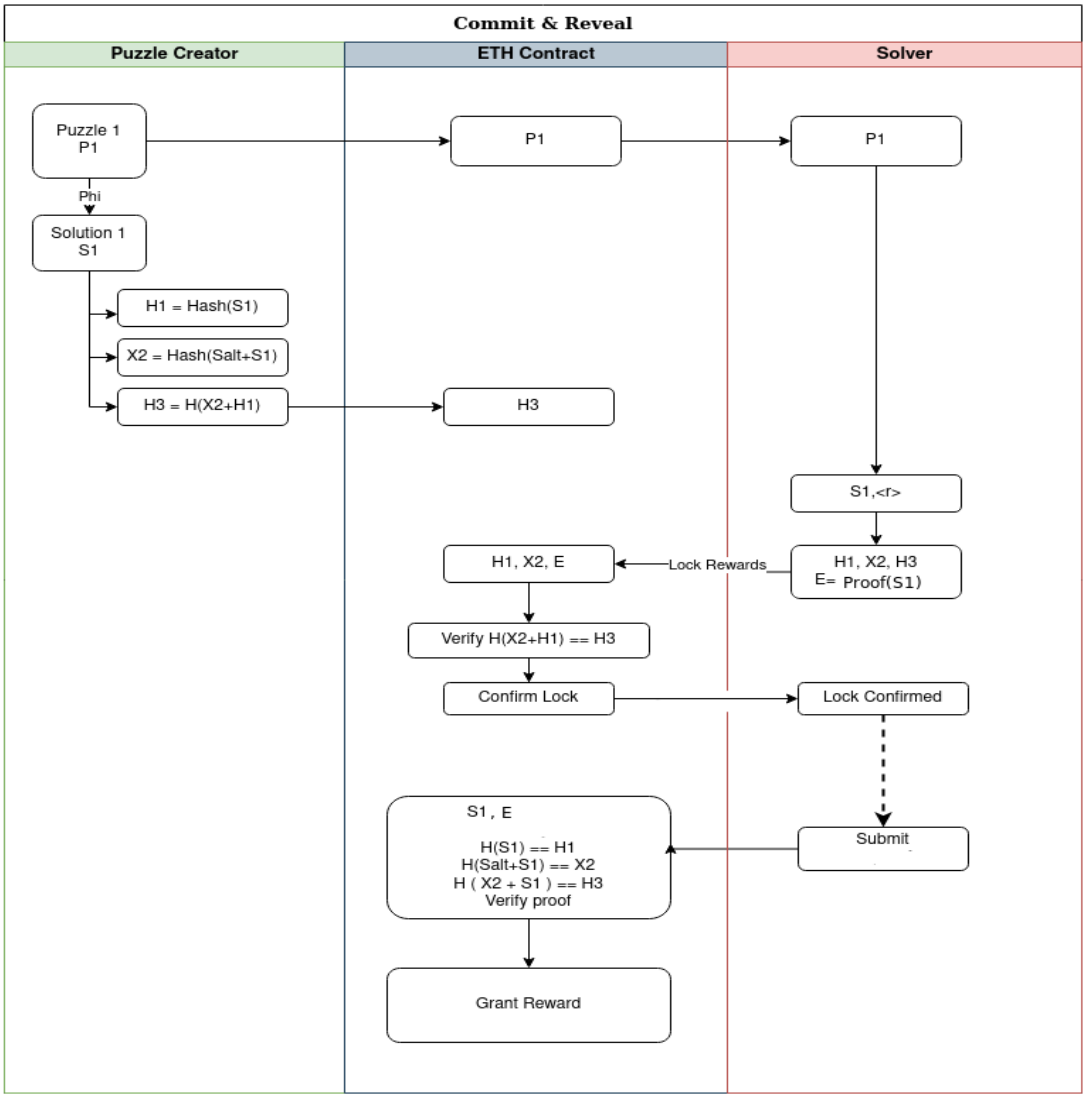
Because puzzle solvers must receive the rewards for their work without allowing an enterprising Ethereum miner to substitute their own address, we must provide a way for them to prove they are the legitimate solver of the puzzle, before providing the solution and its proof.

The way this is typically implemented is as follows. The application 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 application computes the hash of the solution S_1 as $H_1 = Hash(S_1)$, and then the hash of the combination of these results is calculated as:
$H_3 = Hash(Hash(Salt + S_1) + Hash(S_1)) = Hash(X_2 + H_1)$
The application submits H_3 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 first submit $X_2 + H_1$ to prove they have found a value that hashes to H_3 and then create a Pietrzak proof of the work they did to obtain S_1, and send it as E to the blockchain contract 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 values $X_2 + H_1$ from being able to claim the reward without providing proof that they know S_1 later on. When the puzzle solver submit the proof E, the contract can verify the proof S_1 and confirm that the puzzle solver who initially submitted 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. Also note that this commit/reveal method of claiming rewards is not specific to chained puzzles, and therefore will be used to submit other solutions. Also note that a malicious Ethereum node has no way to 'grab' values from the requests to claim the puzzle solution for themselves. The blockchain contract can enforce a delay of at least one block between these interactions.

5.4. Condition Based Threshold Decryption

5.4.1. Network Access

During the initial set-up the application needs to access the NuCypher network as well as an Ethereum network provider through the use of a Web3 wallet, for publishing to the smart contract. Subsequently, every time the timer is renewed, the application needs to connect to the user's Web3 wallet in order to update the contract.

In the current architecture, the NuCypher network provides the public key for encrypting secrets. It is possible that in the near future, such communication might not be necessary. The application will not issue an Ethereum network request to publish and activate a new puzzle until this is complete.

5.4.2. Condition

A simple condition requiring that a user's 'instance' has expired is sufficient to guarantee that the nodes in charge of decrypting the secret payload will not authorize the decryption unless the contract says so. The user's wallet address (account) which is used to index their instance within the smart contract, is hard-coded into the condition. At a high level, the condition is an instruction to check with the smart contract that the user's instance has expired. As explained in the NuCypher section, each Threshold node in possession of a key share will execute a call to the contract to verify the condition before agreeing to decrypt.

5.4.3. Renewal

When the user decides to 'flip-the-hourglass' timer and create a new puzzle, the condition will not change. Only the last published puzzle is considered "active". In this way, users can renew a timer early, or change out their timer for a longer period of time.

5.4.4. Revocation

Because it is to be expected that people, in the course of their lives, might want to change beneficiaries, allocations or their will or modify their revelations, users can update their payload and publish the new encrypted version to the contract at any time. To ensure forward secrecy from the Security Goals section, we opted to bind the conditions to the last published payload. This ensures a past payload can never be decrypted.

As the payload is published to the contract, a simple signature of that message is kept as a unique identifier by the contract and used as an extra comparison when crafting the condition to its decryption. When NuCypher nodes call the contract to verify approval to decrypt, only those messages matching the identifier will be approved for decryption. This is our unique solution to the problem of 'erasing' data on public ledgers

5.5. Reward Management

5.5.1. Reward Storage

The Ethereum contract holds Ethereum tokens with allocation details for each of the puzzles, in each of its instances, for each user. A planned feature is to allow users to contribute to the escrow fund of any account instance. This means a user wishing to remain anonymous can obfuscate the origin of their funds.

Once the user has flipped the hourglass and published a new puzzle, older puzzles remain active until their reward has been claimed, but have no effect on the release of the payload..

5.5.2. Reward Allocation

Reward allocation is initially set by the user on a time-duration basis. The BqETH contract will also store and make available the market rate for puzzle solving rewards, so that users can make informed decisions about the reward they wish to attach to their timers.

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[6], 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 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.

5.5.3. Decryption Reward

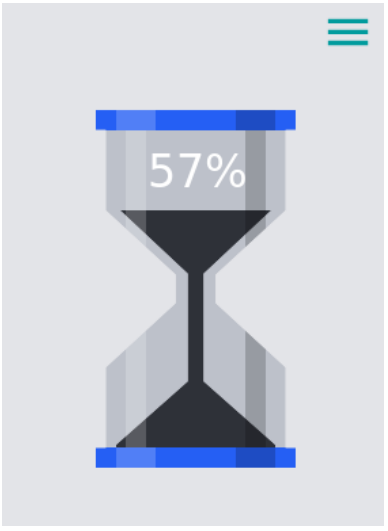
A small portion of the Timer's reward will be set aside in the contract to reward the proven decryption of a secret. This is an incentive to make sure secrets are decrypted in a timely fashion and to ensure they are published. The contract uses a Commit/Reveal scheme similar to the one used for puzzle reward claiming. When the secrets are small, the complete clear text message can be provided as proof of decryption, and in the case of larger payloads, Merkle proofs can be provided for the decryption and storage of the decrypted payload on IPFS.

5.6. Interaction

5.6.1. Web3 Application

The Web3 Application allows a user to check on the status of their current timer and the expected date by which they are expected to 'flip the hourglass'. We are only supporting a browser app at the moment since we have security concerns with centralized phone platforms, their toolkits, and app censorship. All applications will eventually be open-source and audited by BqETH.

The user application, because security is very important, features a simple interface allowing very few customizations and exposing very few parameters. In the future the application will include other notifications for clients of the monitoring services.



5.6.2. Wallet Integrations

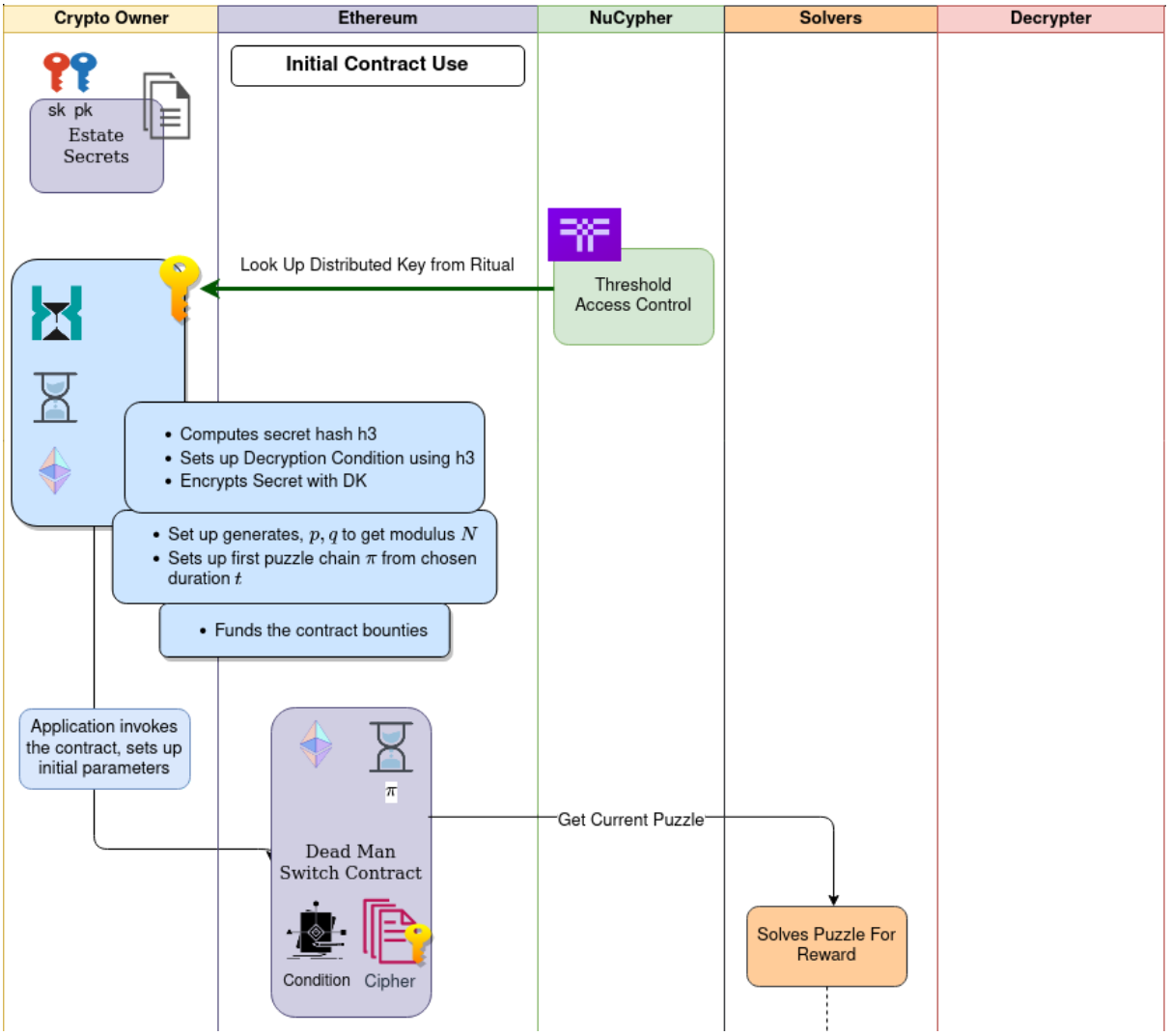
Since funding of the Timer Reward of each instance may be performed from any address, the application integrates with several well known crypto wallets such as Metamask, Coinbase as well as with portals such as WalletConnect and Hardware wallets.

[6] 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. See for example: https://www.gwern.net/docs/www/blog.janestreet.com/68ce637aa08a052399f781ddaf8e7f2fcb45e693.html

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 Solvers.

5.7.1. Setup

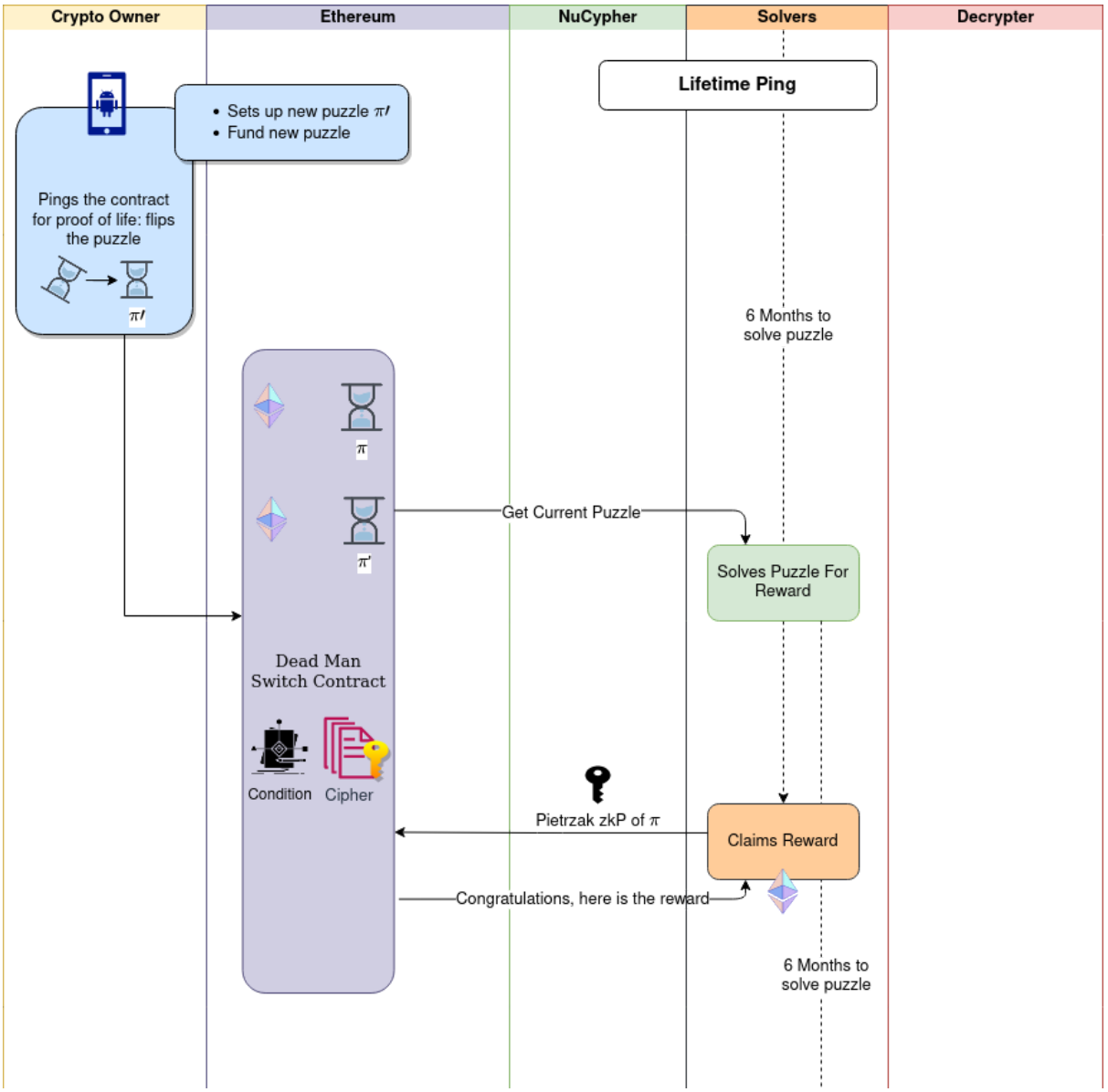


- In the initial setup phase, the user has a secret key (sk) and public key (pk) pair that will allow them to use the Ethereum contract.
- Using their device, the Application will pick random large safe primes p,q to obtain N and ϕ. They will also pick a random number to create the first puzzle π.
- From the application, connectivity with the NuCypher Network is initiated and the Distributed Encryption Key is accessed to encrypt the secret
- A condition is created for decryption of the payload. Because the condition is false as long as the use has an active puzzle, effectively no-one can currently request decryption of the cypher until the puzzle is solved.
- The final setup step is for the user to call the Ethereum contract and publish the cipher and puzzle challenge along with the rewards for it.

- The puzzle is then ready for solving and anyone can request the parameters of the challenge.

5.7.2. Recurring Interaction

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.



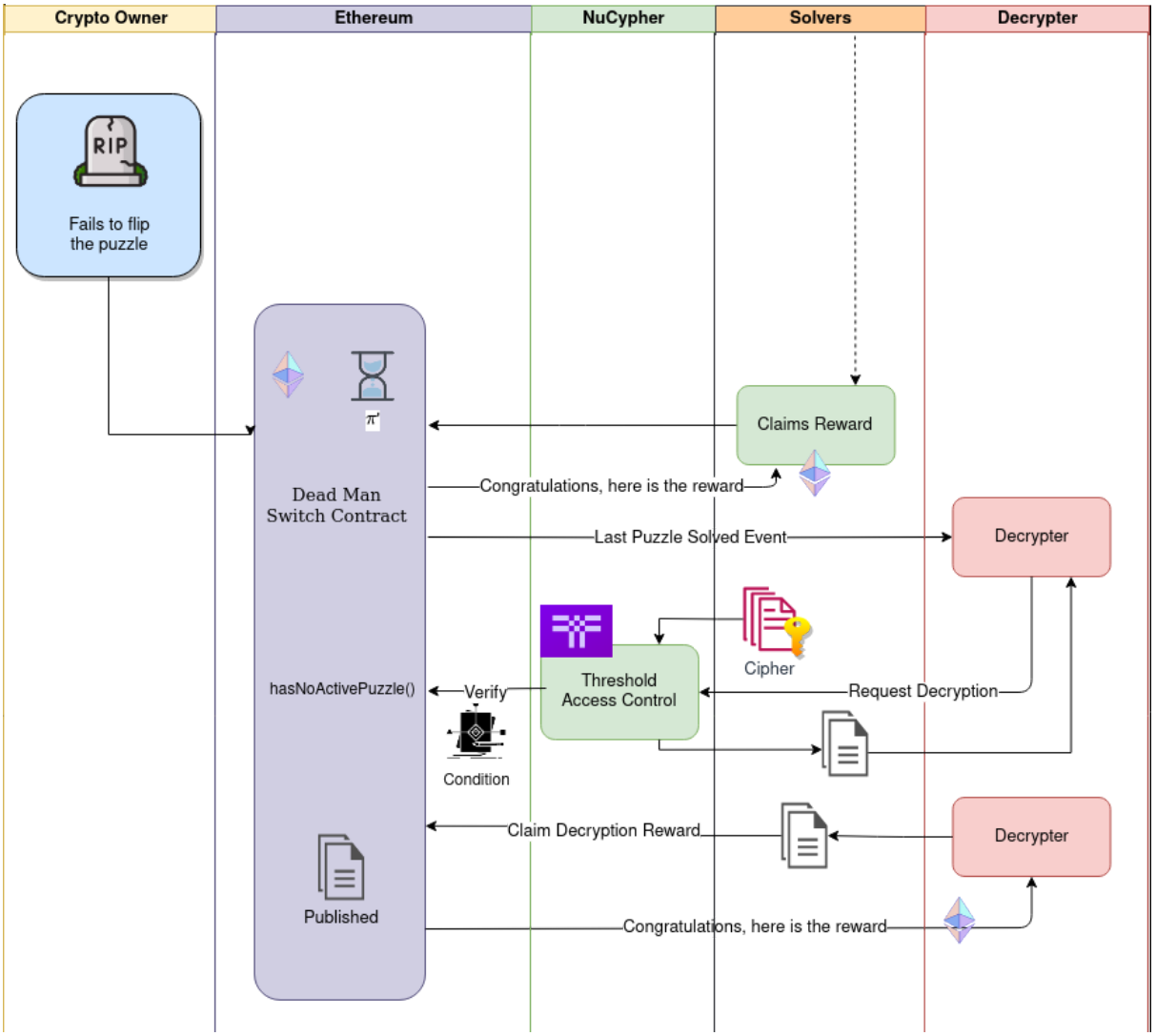
- In this step a new puzzle π' is created.
- There is no interaction necessary with the NuCypher Network.
- The new puzzle π' is published to the Ethereum contract as the new 'active' puzzle.
- Solving the older puzzle π still allows a puzzle farmer to contact the Ethereum contract to claim their Ethereum reward.

5.7.3. Secret Replacement

Replacing the secret, as is expected for clients who need to amend or alter their wills, is simply a matter of re-encrypting a new payload. The decryption condition associated with the payload is updated to only allow decryption of the last secret.

5.7.4. End of Life

The final illustration shows what happens when the user passes.



- Assuming the last puzzle π' has been properly solved as the user was unable to replace it, an event is emitted from the blockchain contract notifying anyone interested in decrypting the message.
- Decrypters can access the Cipher and request its decryption by NuCypher. The NuCypher network nodes will call upon the contract to verify that the Decryption Condition is satisfied. If so, a threshold number of nodes is assembled to decrypt the message which is then sent back to the decrypter.
- The decrypter can compute proof of decryption and submit the plain text and proof to the contract for publishing in a Commit/Reveal interaction to claim the decryption reward.

5.8. BqETH Services

Among the services BqETH will provide to clients of the Dead Man Switch, Notification and Delivery are two important features that leverage the ability to monitor the blockchain contract for events. BqETH Notification services will reach out to its clients when unusual situations emerge. For example, when a timer nears completion, or if reward levels or puzzle solving speeds have changed significantly since the timer was set-up. BqETH provides in-app suggestions for reward amounts when setting up the timer, but over long periods of time, these may change.
BqETH Delivery services will provide the decrypted payload of its clients when a timer expires, to the destination of their choice. Other services, such as puzzle farming, IPFS replication and payload design, will be provided but are outside the scope of this white paper .

Naturally, the BqETH's Application makes a distinction between the Secret Payload , i.e the information that is to be delivered in the clear upon death, the Delivery details, in essence, who should be getting the decrypted message and the Notification details, how to alert the owner.

The Secret Payload is encrypted only using the NuCypher Threshold Public Key and stored. The Notification information is encrypted with BqETH's public key only and sent through the contract as an event, but not stored[7]. The Delivery details are encrypted for BqETH's public key first, then encrypted using the Threshold public key. This ensures that Delivery information is unknown to BqETH until expiration of the timer, and remains private to BqETH after decryption, to protect the identity of beneficiaries or attorneys.

The payload, notification and delivery details can all be replaced independently in the application.

This section discusses things that will be monitored by BqETH in order to provide these services. Many parameters, suggested by the application and enforced by the contract will vary. For example, we can force the contract to only increase its estimates of the values of the solving difficulty 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.

It is also worth noting that the application features a test mode. With this test mode, puzzles use a modulus known to BqETH , allowing their solution to be found at no cost. The timers in this mode are limited to a very short period of time. This enables BqETH to conduct training sessions with the live application but with non resource intensive puzzles.

5.8.1. Puzzle Farming

Puzzle solving needs 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 to start solving puzzles and then switched to slower processing in order to discourage other puzzle farmers.

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 switch out the secret without re-generating the puzzle, effectively canceling the contract.

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 detected

[7] Events are stored and can be found, but do not take 'storage' space in the contract.

by BqETH for its customers, and BqETH services will notify them using the settings provided during set-up.

5.8.3. NuCypher

Monitoring of the health of the NuCypher network may also become important if more resources must be dedicated to providing a reliable service. Also, monitoring of the NuCypher Distributed Key will be important to make sure it remains funded. NuCypher will provide an API for approving Encrypters and sponsoring the cost of the Distributed Key for its node operators. Authorization of encryptors is only necessary before decryption and can be done after encryption has taken place, therefore this can be performed asynchronously from encryption.

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 under-rewarded 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. The application provides a suggested reward based on a value maintained in the contract.

5.8.5. 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 file upload feature is available in the application, which saves the encrypted information on IPFS, the decentralized file storage system.

The locator for the encrypted payload, known as "CID", can be stored in the smart contract as-is. Using this BqETH can effectively monitor the existence of a replicated version 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.

5.8.6. BqETH Puzzle Farming

BqETH will conduct Puzzle Farming as a profitable source of revenue for itself, initially until a competitive market develops. For its clients, BqETH this provides a guarantee that a puzzle is always being solved, to ensure their heirs will always be able to access the decrypted information.

5.9. Stability

5.9.1. Farming

Because puzzle-farming will rely on a system of incentives, it is important to discuss the various forces at play that will hopefully lead to the emergence of a stable system.
What will lead someone with a spare computer to choose to dedicate CPU cycles to farming BqETH puzzles ? What will influence them to stop and what sort of reward will need to be offered to them for solving the puzzles are important questions.

We think that a thriving market will develop that will allow solvers to sell each other partially solved puzzle chains. The system will operate in a world in which there is an opportunity cost to doing computation, for example doing crypto currency mining or validation, or lending CPU cycles for example with iExec or Golem. Companies such as Amazon or Linode sell CPU access that will put a

soft ceiling on the reward that must be offered[8]. It is important to keep in mind that the decision to farm puzzles for Ethereum rewards will also depend on the value of Ethereum relative to other crypto currencies.

Since older, expired puzzles with uncompetitive rewards will surely exist, new uses might be discovered for older hardware in the form of puzzle farming. As computing power is expected to fall, this may seem surprising, but the opportunity cost is what drives these decisions.

Will there be a reason to load-balance the solving of puzzles based on their known status (new, half-done, near completion, expired) ? Certainly, the risk of physical destruction of puzzle farming hardware would have a much bigger impact on the decision to farm new puzzles versus expired puzzles. Floods, fires or simple power loss could impact such operations.

What if the existing reward on a puzzle gets too uncompetitive ? We have considered the possibility that solving the last active puzzle could carry the reward for all inactive puzzles -all puzzles with no probability of being 'worked', as an incentive to clear-out older puzzle chains. This is not implemented but could soon be a feature to incentivize puzzle solvers to focus on the most recent puzzle, the one with the most value for our user.

5.9.2. NuCypher

The stability of the NuCypher network will also be of significance. The NuCypher network is secured by the threat of slashing nodes from their stake in NuCypher tokens if they fail to provide the service. It is natural to wonder if the value of such a stake may become a vulnerability should the NuCypher token become less valuable than they are now.

6. Improvements and Future work

6.1. Quantum resistance

The first obvious improvement to the system, and the most common question we get is how to address Quantum Computing. The simplest is to switch the puzzle math to one that uses Elliptic Curve multiplication. Some VDFs have already been partially tested with Elliptic Curve operations and the Pietrzak VDF can also be adapted for Elliptic Curve operations.

But this only makes the system Quantum Frustrating. For Quantum Resistance, a conversion of the CVDF to Supersingular Isogenies will be necessary. What is important to note is that the conversion can be entirely transparent to users of the dead man switch.

6.2. Biometric Device Integrations

For some clients, ensuring the smooth operation of their instance might generate interest in some integrations with biometric devices which can confirm their identity or detect operation under duress. Because there are many projects leveraging the blockchain for securing identities, there is little doubt we will investigate the possibility of leveraging and integrating them into the app to streamline authentication.

[8] Rewarding a puzzle above this level will almost guarantee an arbitrage profit for someone willing to rent an AWS computer, and therefore jeopardize the possibility that many solvers would compete for the reward. As a result, this level of reward would be counterproductive to ensuring the eventual decryption of their secret, should they die.

6.3. Wallet Abstraction Integration

One of the biggest technological advances of 2022 and 20223 in blockchain technologies has been the advent of Wallet Abstraction technologies: Magic and others have allowed users to on-ramp into Crypto using traditional Banking systems and have provided features such as social recovery, while avoiding the need for users to 'remember' a seed phrase. While it seems odd, on the surface, that someone would want to lock-up crypto secrets while at the same time avoiding them by using wallet abstraction technology, it is actually a likely scenario. Our system has applications for management of multisignature keys, or securing of non-crypto related secrets, or custodial situations involving law enforcement or probate courts. In these situations, wallet abstraction technologies allowing the use of biometrics and other identification methods may prove useful.

6.4. Wallet Tech

Perhaps it is not obvious, but a useful scenario for the dead man switch is that of a wallet. Wallet technologies have typically focused on permitting immediate access to the owner and no-one else. However the dimension of time is largely lost, while the focus is restricted to the security of a physical device, often re-centralizing what should remain decentralized.

However, the use case of a piggy-bank, which cannot be opened until a future date is equally interesting. If the idea that crypto is a safe asset to cross borders is important, then an easier solution than to remember twelve words might be to use BqETH with a 'burner wallet' to set up a timer which expires in a few months and ensures you will be delivered your own seed phrase.

This concept of a time-capsule wallet will be worth exploring.

Another application is that of management of multisignature systems in which many parties cannot afford to lose a quorum of signing shares. A Proof of Paralysis proposal was put forward in 2018 that solved this problem for corporate boards who need to recover from death or dementia situations. Our system is very suitable for solving these kinds of problems.

7. Conclusion

Many people have tried to create a truly decentralized general purpose dead man switch for digital information. Pamela Morgan calls it the holy grail of computing. While this is an important achievement that many have naively thought they had solved, we believe it is a natural solution arising out of the many technologies being created every day by very smart crypto enthusiasts. We will continue to improve it. Its utility is evident, and will become obvious and mandatory to a growing audience as the value of crypto currencies increases against the value of fiat currencies, and as the bureaucratic controls become more prevalent.

The system is designed to give users ultimate control over their assets, to be decentralized in the best way possible, eliminating the threat of censorship or denial of service, providing perfect forward secrecy, offering privacy and time integrity. The cost of operation is as low as possible to guarantee time integrity. The system is non custodial and protects beneficiaries with the same level of protection as the assets they will inherit.

8. 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. Blum, Blum, Shub https://crypto.junod.info/bbs.pdf (1986)
9. Rabin, Thorpe. Time Lapse Cryptography (2006)
10. Liu, Jager,Kakvi, Warinschi. How to build time-lock encryption (2015)
11. Wesolowski . Efficient verifiable delay functions (2018)
12. Pietrzak. Simple Verifiable Delay Functions (2018)
13. Ning, Dang, Hou, Chang. Keeping Time-Release Secrets through Smart Contracts (2018)
14. Albrecht, Massimo, Paterson, Somorovsky. Prime and Prejudice: Primality Testing Under Adversarial Conditions (2018)
15. Boneh, Bunz, Fisch. A Survey of Two Verifiable Delay Functions (2018)
16. De Feo, Masson, Petit, Sanso. Verifiable Delay Functions from Supersingular Isogenies and Pairings (2019)
17. Ephraim, Freitag, Komargodski, Pass. Continuous Verifiable Delay Functions. (2019)
18. Malavolta, Thyagarajan. Homomorphic Time-Lock Puzzles and Applications (2019)
19. Attias, Vignery, Dimitrov. Implementation Study of Two Verifiable Delay Functions (2020)
20. Burdges, De Feo. Delay Encryption. (2020)
21. Krishnan, Gong, Bhat, Kate & Schröder: OpenSquare: Decentralized Repeated Modular Squaring Service (2021)

9. Appendix

Euler Totient Function's Trapdoor.

Euler's Totient Function $\phi(n)$ allows the trapdoor simplification of the RSW Puzzle because of Euler's Theorem: $a^{\phi(n)} \equiv 1 \pmod{n}$

As follows: write $e = 2^t$ and assume $e = c + d\phi(n)$ for some d we can write $e \equiv c \pmod{\phi(n)}$ then

$$a^e \bmod n = a^{[c+d\cdot\phi(n)]} \bmod n$$
$$= a^c \cdot (a^{\phi(n)})^d \bmod n$$
$$= a^c \cdot 1^d \bmod n \quad \text{per Euler's Theorem}$$
$$= a^c \bmod n$$

Since $e \equiv c \pmod{\phi(n)}$ or equivalently $c = e \bmod \phi(n)$ we have

$$a^e \bmod n = a^{e \bmod \phi(n)} \bmod n$$

Carmichael's function divides Euler's totient function, so the trapdoor can be simplified to use it, and in many cases this makes sense, because $\phi(n)$ could be much larger. But because Carmichael's function is slightly more difficult to calculate, there is no advantage in using it here, since the solution is rarely calculated.
Carmichael's function is defined as $\lambda(N) = lcm(\lambda(p), \lambda(q)) = lcm(p-1, q-1)$.

ElGamal Encryption

Recall that to encrypt a secret m, ElGamal encryption creates two values:
$c_1 = g^r \bmod P$ and $c_2 = p_k^r \bmod P \oplus m$ with $p_k = g^{s_k} \bmod P$, with r a random value and P a prime.

Decryption by the holder of s_k is done with $m = c_2 \oplus (c_1^{sk}) \bmod P$ since it simplifies to $m = (p_k^r \oplus m) \oplus (g^r)^{sk} \bmod P = (g^{sk})^r \oplus m \oplus (g^r)^{sk} \bmod P$.

Alternate versions exist in which terms are multiplied by the modular inverse of g^{sk} and a similar formulation exists for the Elliptic Curve version.