Dead Person Switches - with or without trust

Lewis Westbury

Kellogg College

University of Oxford

Abstract

More commonly known as a Dead Man's Switch, a Dead Person Switch (DPS) is a system that can hold a secret on behalf of a person, and govern its release should they become unavailable.

Little formal work exists on this topic. A similar topic, time-release cryptography, is an active field of research and has several contributions that can be applied when considering a DPS.

A number of use cases exist for a DPS. This thesis focuses on the needs of an investigative journalist. The implementation is divided into three core sections:

1. Development of requirements
2. Evaluation of existing systems and components
3. Proposal and evaluation of new designs

Through desk research, study, and survey this thesis derives the following requirements for a DPS: Confidentiality, Awareness, Timing, Resilience, Affordability, Durability, Explainability, and Visibility.

Several existing solutions are evaluated against these requirements. Each has strengths and weaknesses, but none are shown to be appropriate for the chosen use case, or able to meet all requirements.

Several components that could meet the requirements of a DPS are discussed, and three designs are proposed and evaluated against the requirements:

1. A classic micro-services architecture
2. A distributed application (dApp) built with a smart contract and secret contract
3. An application of witness encryption

These represent distinct approaches to the design of a DPS, and are shown to meet more of the requirements. This thesis concludes that the dApp design is the strongest offering that can currently be implemented. The witness encryption design presents a strong hypothetical alternative, but requires further research.

Acknowledgements

First, I would like to thank everybody who took the time to contribute to this thesis by participating in the survey, and to the kindness of the volunteers at Bureau Local for permitting me to share information about it with your community.

Thanks to all my lecturers and classmates for being so generous with your knowledge and perspectives; to Anjuli Shere and Ivo Maffei for taking the time to talk to me about your brilliant research; to Ed Saperia, Dean of the London College of Political Technology, for helping to chart the wide range of communities of practise that exist in the field of investigative journalism; and to Professor John Barnden and Dr Huma Shah for sharing insights into their work and the Loebner Prize events.

Special thanks go to Dr Christopher Hargreaves for your patient and thoughtful supervision.

Finally, thanks go to my friends and family, whose encouragement, patience, and proof reading has finally been rewarded. I have completed something.

Contents

List of Tables

List of Figures

1 Introduction

More commonly known as a Dead Man's Switch, a Dead Person Switch (DPS) is a system that can hold a secret on behalf of a person, and govern its release should they become unavailable.

Use cases for such a system include final messages to loved ones, the distribution of assets after death, and the controlled release of a damaging secret to disincentivise an attack on the owner of a DPS (the subject). These are discussed in section 2.

This capability is of interest because it apparently contradicts a simple mental model of encryption:

- **Simple encryption:** A secret is encrypted with an encryption key, and the subject must provide the decryption key to restore the plaintext of the secret.
- **DPS:** Assuming the secret is encrypted, a DPS must be able to decrypt and release it in the *absence* of the subject.

It is not immediately obvious how the decryption key can be kept confidential but also made available to decrypt the secret when the user is not present.

This thesis contributes a set of requirements for a Dead Person Switch, a review of existing systems that purport to offer this capability, and several evaluated designs for a potential DPS.

In doing so, it addresses the following questions:

- What are the guarantees that a DPS system should provide to its users?
- How well do existing solutions deliver these requirements?
- Which existing information security tools could contribute to a working DPS?
- Which approaches could deliver working designs, and how well do those designs meet the requirements?

The remainder of this thesis is structured as follows:

- **Background** - Describing the user needs that DPS systems seek to solve.
- **Related work** - Exploring existing research and systems related to the field.
- **Methodology** - Structure and methods for the work conducted as the implementation of this thesis.
- **Implementation** - Split into 3 sections:
  - establishing a set of requirements for a DPS,
  - a review of existing systems and components, and
  - proposal of new designs that could meet requirements.
- **Evaluation** - Reflection on this thesis and methodology.
- **Conclusions** - Proposals for further research, and summary of findings.

A number of appendices are included which provide additional detail in support of this work:

Table 1: Appendices

Appendix	Description	Citations
A	Survey participant documents, for section 5.1.	
B	Threat Dragon model for a generic DPS, for section 5.3.	
C	Evaluation of existing solution tables, for section 6.	
D	Subjective logic - further detail, for section 7.1.2.2	[1] [2]
E	Paralysis proofs - further detail, for section 7.2.4	[3]
F	Trusted computing - further detail, for section 7.3.1.	[4] [5] [6] [7] [8]
G	Design and risk assessment tables, for section 8.1.3.	

2 Background

The need for a DPS is seen in many walks of life. A number of use cases are presented below, each prefixed with an illustrative user story.

2.1 Use cases

2.1.1 Whistleblower protection

> ***As a*** *whistleblower,* ***I want to*** *have a secret released on my death,* ***so that*** *I can disincentivise attacks on me.*

A whistleblower or investigative journalist (the subject), may grant control of the release of their secret to a DPS (the switch). The switch will then attempt to monitor the well-being of the subject. If they are unavailable for a determined period of time, it will release the secret. Provided the secret chosen is sufficiently damaging to the expected attacker (the threat), it can serve as a disincentive to physical attack.

In *A critical analysis of whistleblower protection in the European Union* [9], Popescu notes that "employees who reveal inside information are vulnerable to retaliation," and "without protection from retaliation, many would-be 'whistleblowers' will remain silent." Written in 2015, the paper notes that at that time only 4 EU members have legal protection frameworks for whistleblowers and their families. (Others were in the process of implementing one, or hadn't started yet.)

2.1.1.1 WikiLeaks insurance files In an attempt to protect their capability, WikiLeaks (an organisation devoted to publishing leaked data), published an 88Gb encrypted file and publicised it with the tweet illustrated in fig 1.

Figure 1: The 2016 WikiLeaks insurance files tweet [10]

Possible uses for this secret include:

- A disincentive for attacks on their members (ie. the data might be decrypted if members are attacked).
- Organisational continuity (ie. allowing the continued release of information even if members are attacked) - so making an attack futile.

2.1.1.2 Edward Snowden's DMS In May 2013, Edward Snowden leaked thousands of US intelligence documents before fleeing the country. He shared the documents with a couple of journalists, who were tasked with responsibly sorting and releasing evidence of wrongdoing. As reported in Wired that year [11], he also shared an encrypted copy of the entire cache with a number of people to act as a Dead Man's Switch:

> "Snowden also reportedly passed encrypted copies of his cache to a number of third parties who have a non-journalistic mission: If Snowden should suffer a mysterious, fatal accident, these parties will find themselves in possession of the decryption key, and they can publish the documents to the world."

Edward clearly perceived a risk to his own well-being, and recognised the potential of secrets he had obtained to disincentivise an attack on him.

2.1.2 Inactive Accounts

> ***As a*** *user of a digital service,* ***I want to*** *pass control of my account to chosen people in the event of my death,* ***so that*** *they can administer my account appropriately.*

Popular online service providers, such as Google [12], and GitHub [13], are increasingly responsible for the administration of accounts belonging to people who have died. These services have policies and techniques for passing control of those accounts to chosen colleagues, relatives or friends - and have developed techniques to resist attempts to fraudulently declare a person alive or dead.

2.1.3 Final wishes

> ***As a*** *mortal,* ***I want to*** *control the distribution of my assets after I die,* ***so that*** *my final wishes are enacted.*

Writing a will allows people to control the distribution of their assets after death. Which? Magazine notes that in 2018, 54% of UK adults did not have a will [14], risking loss of control of their assets.

This is a function that a DPS may fulfil. In *Fifteen Men on a Dead Man's Switch* [15], 2015, Lopp describes a process designed to meet that use case by storing and releasing a secret to a group of relatives or friends after death. This is explored in section 3.8.

Switches that operate in the realm of cryptocurrencies may also be able to distribute wealth automatically when triggered. Charitable giving is a special case of this, differing only in the chosen recipient.

2.1.4 Organisational continuity

> ***As a*** *member of staff at an organisation,* ***I want to*** *share the secrets I use to administer that organisation with chosen colleagues after my death,* ***so that*** *they may continue to administer the organisation.*

This use case is illustrated by the case of the Ivar Aasen Centre of Language and Culture, Norway. In 2002, the organisation posted an appeal: One of their archivists had passed away, taking with him the only password to a database of 11,000 titles. The password was recovered, however, in short order: The Slashdot community found it in just a few days [16].

This use case illustrates the importance of organisations developing access management policies that prevent unexpected deaths or staff leaving from locking vital resources. A Dead Person Switch is one way to achieve this, although not necessarily the most appropriate solution.

3 Related work

A number of components and solutions that claim to offer some of the properties of a Dead Person Switch are available either as consumer offerings, distributed apps (dApps), or open source projects. These solutions offer different sets of guarantees, and some come with disclaimers as to their suitability per use case. They are assessed in detail in section 6.

Academic research into DPS solutions is sparse, although many of the components that could contribute to a solution are well documented, or active areas of research. This thesis analyses relevant components, in the context of their application to a DPS, in section 7.

3.1 Hosted consumer DPS implementations

A number of systems exist today which purport to meet some the requirements of a Dead Person Switch. Each offer different capabilities, making claims about their ability to protect and distribute secrets.

Table 2: Hosted consumer DPS implementations

Product	Summary
deadmansswitch.net	A system designed to meet end of life needs - such as messages for loved ones. It checks for signs of life, and if not met, sends a number of emails to chosen recipients.
deadman.io	A system that checks for a response, and distributes documents by email if it does not receive that response.
deadmantracker.com	A service that automatically contacts your friends and/or family in the event that something happens to you.
Letters Cloud	A service that allows you to create messages to be sent, should you stop visiting your trigger link. (formerly komprom.at)

These implementations are presented and analysed in detail in section 6.1.

3.2 Distributed Applications (dApps)

Distributed Applications (dApps) are built to be run as one or more smart contracts, evaluated as a part of the blockchain for a cryptocurrency. dApps are, by nature, open source or represented by open protocols, and so there is a lot more information with which to reason about their capabilities.

Table 3: Distributed application (dApp) DPS implementations

System	Service description
KillCord	A tool to build resilient dead man's switches for releasing encrypted payloads.
Kimono	A digital time capsule built on the Ethereum blockchain.

These systems are presented and analysed in section 6.2.

3.3 Open source projects

A number of open source projects purport to meet some of the needs of users of a DPS, ranging from student or hackathon projects through to research based initiatives. These solutions often require that the user also find a solution to host and operate their system, and this can alter their security properties.

Open source solutions are evaluated in section 6.3.

3.4 A taxonomy of threats to journalists

At time of publishing, an authoritative taxonomy of threats to journalists is not yet available[1]. This thesis presents a simple risk model in the context of a DPS which lists relevant threat actors, and will return to the topic of threats to journalists in discussion of potential future work.

[1] The researcher is aware of current work to develop such a resource, as yet unpublished.

3.5 Summary of existing related work

There are a number of papers, techniques and tools for components relevant to the design and implementation of a DPS, discussed in detail in section 7.

Trust networks are discussed in section 7.1:

- K of N threshold schemes
- Shamir's Secret Sharing Scheme [17]
- Subjective logic [1] [2]

Aliveness checking is discussed in section 7.2:

- Classic authentication
- Biometrics
- Human judgement
- Paralysis proofs [3]
- Inactive account managers [12] [18]
- Validation of evidence [19]
- General Data Protection Regulations [20]

Confidential computation is discussed in section 7.3:

- Trusted Computing (TPMs, TEEs) [4] [5] [6] [7]
- Homomorphic encryption [21]

Guaranteed execution is discussed in section 7.4:

- Device hardening [22] [23]
- Distributed computing, science, smart contracts [24] [25]
- Durability

Publishing mediums are discussed in section 7.5:

- Public forums
- Trusted channels
- Distributed file systems

These cover components that can contribute to the design and build of a DPS. The following sections describe a number of works that relate more directly to Dead Person Switches.

3.6 Article: Decentralizing a Dead Man's Switch

The article *Tell No Tales? Decentralizing a Dead Man's Switch* [26] by Ainsley Sutherland, deserves special mention. This is an analysis of Dead Person Switches through the lens of a distributed confidential computing solution called *SECRET* (formerly ENIGMA), that could potentially meet several of the requirements of a DPS.

Exploring *KillCord*, an existing open source solution, Ainsley identifies the various properties of the system and also the missing component: the actual secret encrypted and distributed through IPFS[2]. The decryption key is held by a trusted third party, the "publisher." As a distributed app, KillCord can provide guaranteed execution, and a highly available aliveness check, but relies on that publisher to manage the secret itself.

Ainsley also explores *Kimono*, a solution which allows a user to divide secret information (for instance, a secret's decryption key) between a number of trusted third-parties, provides a countdown for those parties, and manages a financial incentive for them to retain their secret until the countdown completes through a smart contract. Ainsley and the Kimono team note that a system like this is vulnerable to collusion attacks (where these parties can cause early or late release of the information, provided they are willing to forgo the incentive).

Additionally, this solution must necessarily make assumptions about the resilience of these trusted parties against attack, persuasion or blackmail. If an attack on such a system can identify all the parties involved,

[2]InterPlanetary File System (IPFS) is discussed in section 7.5.4.

their capability to offer a more substantial reward could compromise the reliability of the group. This thesis presents reasoning about trust networks, such as this, in section 7.1.

Ainsley goes on to describe the unique component that the SECRET network offers: *secret contracts*. The security properties of the SECRET network allow a contract to exist and execute with guarantees about confidentiality that aren't traditionally available to smart contracts.

The paper proposes a "starting example" flow for a DPS built with a secret contract on the SECRET network, and a smart contract on the Ethereum blockchain. It is intended to stimulate conversation and feedback on a possible design for a DPS.

3.7 Article: Time-lock encryption

Time-lock encryption is a term for the group of techniques that permit control of the time at which a secret may be decrypted. There is significant overlap with the requirements of a DPS:

- Time-lock encryption allows the owner of the secret to determine when a secret may be decrypted, by control of the effort required to decrypt it. It is often described as "sending a message into the future."
- A DPS has a similar requirement, but must also permit the chosen time of decryption to vary - allowing recovery the plaintext secret should the owner be determined unavailable.

The article *Time-lock encryption* [27], written and updated by Gwern Branwen between 2011 and 2019, also deserves special mention. Branwen presents a history of time-lock encryption methods, and a thorough literature review of existing techniques.

This article serves as a comprehensive and useful starting point for further research, and features a number of techniques relevant to the design and build of a DPS, including:

- Witness encryption
- Bitcoin as a clock
- Distributed secret sharing with smart contracts

Distributed secret sharing by smart contract is a feature of Kimono, an existing solution, assessed in section 6.2.3.

The possible application of witness encryption and bitcoin as a clock are discussed in section 8.3 - a proposed design for a DPS that applies witness encryption.

3.8 Article: 15 Men on a Dead Man's Switch

In his 2018 article *15 Men on a Dead Man's Switch* [15], Jameson Lopp describes a process to prepare a secret that can be unlocked by a number of cooperative, trusted persons after a person's death.

This particular article is of note because it offers a practical implementation, including instructions for creating a data store on a physically isolated device that is encrypted with a secret.

It also illustrates a common mechanism found across several other designs: The fragmentation and distribution of a secret across a group of N participants using Shamir's Secret Sharing Scheme, a K of N threshold scheme described in section 7.1.1.

This solution is reasonably robust - it deals with the use case for passing on a legacy after death, permitting K of N participants to reconstruct the secret and, in the presence of the original laptop's store, decrypt a set of files (containing the user's legacy). It does have some limitations though:

- Jameson assumes that sufficient trusted parties will behave as expected, and that it will be difficult to compromise them. In a situation where the agents are family, this may be a reasonable assumption. However, if the assets being shared are sufficiently large, motives to misbehave may appear - which could lead to participants collaborating to cause an early release.
- The laptop described in Jameson's solution is a single point of failure. If it is destroyed (perhaps through malice or environmental factors leading to hardware failure) or if it is stolen, the secrets are lost.

- As described earlier, if families are in contact through remote means, it may be possible to fake the well-being of a subject by falsifying their presence on a call. This would effect a denial of service attack that could cause the participants to delay decryption of the secrets.

  "I can understand perfectly how the report of my illness got about, I have even heard on good authority that I was dead."

  Mark Twain, 1897

This use case likely assumes regular contact between family, and so it might be difficult to cause an early release by convincing participants that their relative has passed when they have not. Aliveness checks and the quality of human judgement are discussed in section 7.2.3.

4 Methodology

4.1 Objective

The objective for this thesis is to contribute evaluated designs for a number of possible implementations for a Dead Person Switch.

To do this, the implementation divides into several key sections:

1. **Gather requirements** for a DPS system.
2. **Evaluate existing systems** and components that relate to DPS capabilities.
3. **Propose and evaluate designs** for DPS systems that can meet established requirements.

4.2 Scope

A number of potential use cases exist for a system like a DPS, as described in the **Background**. They group reasonably well into the following categories:

1. Whistleblower / investigative journalist
2. Last wishes / charitable giving
3. Organisational continuity

These use cases are significantly different, have different actors, and different security properties.

This thesis focuses on the **investigative journalism** use case. This use case has a few well-defined actors:

- **Subject** - the owner of a DPS, either an investigative journalist, or whistleblower (their source).
- **Threat** - an organisation or person that poses a physical risk to the **subject**.

In this scenario, the **subject** holds a **secret**, and so is able to use its potential for release as leverage to disincentivise a physical attack by **the threat**.

There are a number of benefits to limiting the scope of the thesis to this use case:

- This is a well-defined problem with clear and motivated threats.
- Those threats are motivated and resourced - so allowing solutions that meet the criteria for this scenario to meet the needs of the other use cases, too.
- Investigative journalists are expected to be familiar with some information security concepts, and are considered more likely than others to have experienced threats to their safety.

The alternative scenarios (last wishes / charitable giving, and organisational continuity) are out of scope for this thesis:

- These scenarios don't define a motivated, well-resourced threat.
- A number of solutions for these use cases already exist[3].

4.2.1 Situations out of scope

More complex situations within the investigative journalism scenario, eg. those with 2 or more threats, are out of scope for this thesis. In these cases, complex interactions become possible. Decisions about how to group secrets in a DPS or the nuances by which they operate become more important.

For instance, suppose a subject owns two switches, corresponding to each potential threat. If the subject is attacked, further work would be required to determine how should each switch could assess whether it should activate.

Similarly, if a user has a single DPS containing all the secrets about all the threats, and is attacked - multiple threats are now affected. It might be possible to assume that with multiple threats a user's safety is reduced - but if they should become aware of each other[4] and the risk to their own secret that the other threats could pose, they may be incentivised to protect the user, or sabotage each other.

[3] Examples of solutions to help with last wishes include WeCroak: http://www.kkitcreations.com/wecroak-android-faq/, afternote: https://www.afternote.com/, BeRemembered: https://beremembered.com/, My Wonderful Life: https://www.mywonderfullife.com/

[4] In this scenario, the subject has notified each threat of the existence of their DPS, assuming they have a deterrent effect.

These uncertainties have the potential to quickly multiply the complexity of any solution. This thesis will return to them in discussion of potential future work.

4.3 Stage 1: Gathering requirements

As mentioned above, the implementation of this thesis is divided into 3 stages:

- Gathering requirements
- Review of existing systems
- Proposal and evaluation of new designs

The first of these stages is to develop a set of requirements for a DPS, identifying the needs of its users, understanding the threat landscape, and laying the foundations for risk analysis of existing and proposed solutions. As mentioned in section 3, no formal work exists that already does this. This is implemented in section 5.

4.3.1 Survey

Whilst it is possible to reason about these requirements from first principles, little is known about the needs of investigative journalists in relation to technologies such as a DPS. For that reason, this thesis incorporates a survey targeted towards investigative journalists.

Survey questions are intended to elicit the following information:

- Experience of threats to personal safety of journalists and their sources.
- Knowledge of DPS and DPS-like services that currently exist.
- How a journalist might choose to trust the operator of a DPS service.
- How journalists perceive the impact of losing CIA[5] properties for a DPS.

The survey takes the form of an online form, allowing users to participate anonymously, remotely, and without appointment.

Ethical approval to conduct the survey was sought and granted[6] through the Central University Research Ethics Committee[7] (CUREC) at the University of Oxford.

The researcher completed a number of modules from Research Integrity, provided as a self-learning resource by the University of Oxford Learning Facility[8].

The University of Oxford pre-approve 2 services, considered adequately secure, to host surveys:

- Microsoft Forms
- Jisc Online Surveys

The researcher attended Surveys: Introduction to Jisc Online Surveys and Microsoft Forms, from the University of Oxford IT Learning Centre[9].

The survey's consent form and main content were combined into a single survey hosted on Jisc, and subsequently advertised through various means:

- Open journalism communities, eg. Bureau Local[10].
- Twitter posts[11] with request for retweets and shares by popular investigative journalism accounts, including the graphic reproduced as Figure 2.
- A personal blog post[12] by the researcher.

The participant information sheet and schedule of questions is included as Appendix A.

[5] CIA properties: Confidentiality, Integrity, Availability

[6] Approval number: `CS_C1A_21_011_01`

[7] Central University Research Ethics Committee: https://researchsupport.admin.ox.ac.uk/governance/ethics#/

[8] Research Integrity course content, see: https://weblearn.ox.ac.uk/access/content/group/e0915ead-90fc-4c6b-a041-d43a7f491ade/2020/index.html

[9] Certificate of attendance available on request. *NB. "This certificate does not imply any specific competence"*

[10] Bureau Local: https://www.thebureauinvestigates.com/local

[11] Appeal through Twitter: https://twitter.com/instantiator/status/1395852175746686977?s=20

[12] Researcher's blog post: https://instantiator.dev/post/dead-mans-switches-appeal/

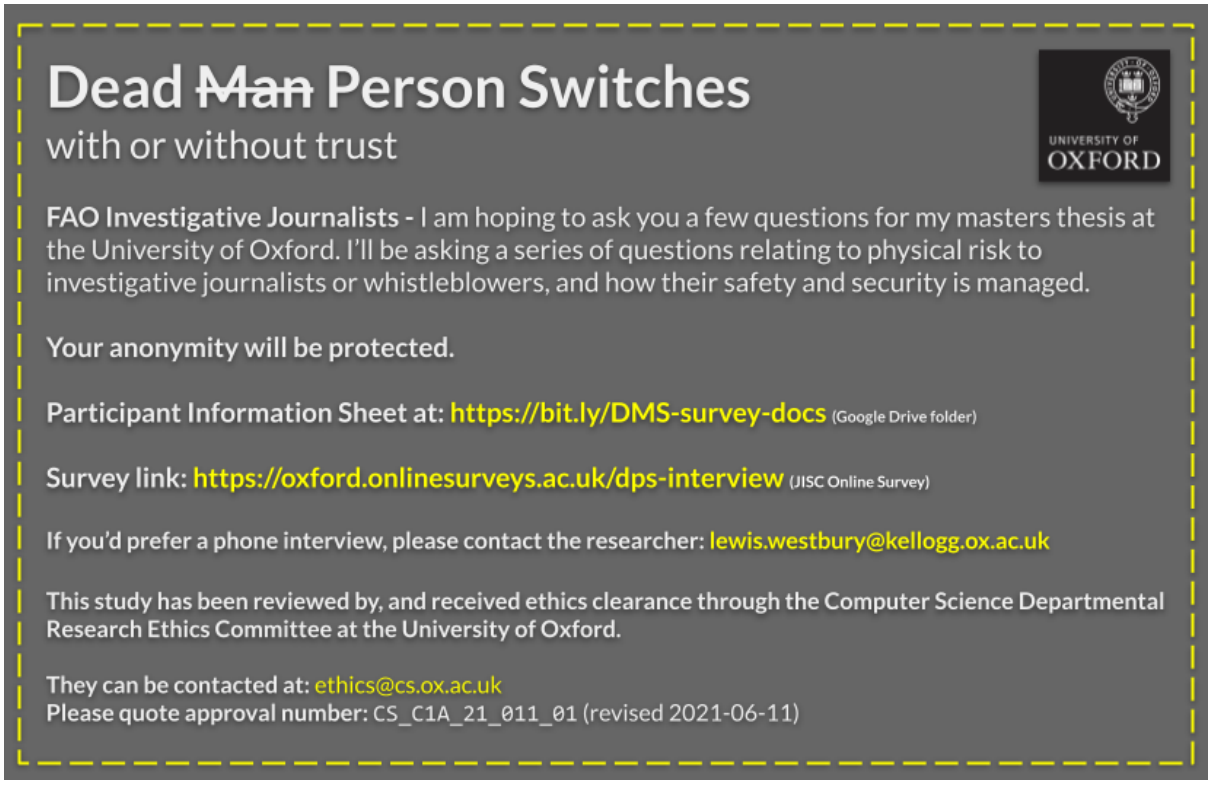


Figure 2: Survey appeal graphic

4.3.2 Synthesis

The resulting responses are reviewed qualitatively and combined with reasoning about the use case to develop a set of requirements for a DPS.

These outputs are used to guide evaluation of existing systems in section 6, and design proposals in section **??**.

4.3.3 Threat modelling

A high level threat model is presented using STRIDE-per-element, a technique developed at Microsoft [28]. This method is selected as the framework permits an analysis of components at any level of abstraction, and provides a set of threat classes that can be quickly applied to each element of a solution. It is also supported by freely available software.

STRIDE is a mnemonic: Each letter of STRIDE represents a different class of threat. The following table is a replication of the STRIDE chart, published by Adam Shostack in 2007 [29], which serves as a tool to understand each possible type of threat in this model:

Table 4: STRIDE-per-element threat classes, reproduced from Shostack [29]

Property	Threat	Definition	Generic example
Authentication	**S**poofing	Impersonating something or someone else.	Pretending to be any of `billg`, `microsoft.com` or `ntdll.dll`
Integrity	**T**ampering	Modifying data or code	Modifying a DLL on disk or DVD, or a packet as it traverses the LAN.
Non-repudiation	**R**epudiation	Claiming to have not performed an action.	"I didn't send that email," "I didn't modify that file," "I certainly didn't visit that web site, dear!"

Property	Threat	Definition	Generic example
Confidentiality	**I**nformation Disclosure	Exposing information to someone not authorized to see it	Allowing someone to read the Windows source code; publishing a list of customers to a web site.
Availability	**D**enial of Service	Deny or degrade service to users	Crashing Windows or a web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole.
Authorization	**E**levation of Privilege	Gain capabilities without proper authorization	Allowing a remote internet user to run commands is the classic example, but going from a limited user to admin is also EoP.

There are two freely available tools to assist building a data flow diagram and annotating it with STRIDE threats.

Table 5: Freely available threat modelling tools

Tool	Comparison
Microsoft Threat Modeller (MTM)	MTM is a closed-source tool, which aims to deliver a lot of threat analysis automatically. It does this with a rich database of components that you might add to your design.
OWASP Threat Dragon	Threat Dragon is an open source tool (still in development). It provides a much simpler model - and generates prompts for threats according to the STRIDE model, rather than attempting to fill them all in. As described by Mike Goodwin in 2020 [30], it is designed to encourage users to think about the threats.

This thesis presents a high level threat report using Threat Dragon, as it permits generic components rather than specifics.

4.3.4 Risk analysis foundations

A risk model for a given DPS depends on its specific implementation. However, it is possible to reason about a number of properties upfront. For instance, the threats can be inferred based on our understanding of the scenario, and it is possible to discuss the impact of exploitation affecting the various informational assets. This section presents tables to represent:

- Informational assets in a generic DPS
- Threats to informational assets in a DPS
- Impact of compromise for information assets in a DPS

4.4 Stage 2: Review of existing systems and components

The second stage toward the objective of this thesis is a review of existing systems and components.

- Existing systems are reviewed in section 6.
- Components are discussed in section 7.

A number of systems exist that purport to meet the requirements of a DPS. In order to establish where opportunities exist for new designs, section 6 presents and evaluates those systems against the requirements developed for a DPS.

A search conducted with relevant terms through Google and GitHub, and combined with references uncovered through research, gives a set of systems. This is further filtered by relevance to the topic of this thesis.

Existing systems can be categorised as:

- hosted consumer systems,
- distributed applications (dApps), or
- open source solutions.

Where possible[13], an analysis of the limitations and vulnerabilities in each system[14] is conducted to help determine which have the potential to be developed further, and which fall short of the requirements. These analyses are presented, and summarised.

Section 7 presents a review of components and techniques with application to the following capabilities:

- Trust networks
- Aliveness checks (and proxies for aliveness)
- Confidential computation
- Guaranteed computation
- Publishing mediums

Each capability lends itself to one or more of the requirements of a DPS derived in section 5.2, and contributes to the design of a reliable DPS.

4.5 Stage 3: Propose and evaluate designs

The third stage of this thesis is the proposal and evaluation of designs for a DPS that can meet the requirements derived in section 5.2, and that are not currently available through existing systems reviewed in section 6. This is implemented in section 8.

Proposals are chosen and developed to represent distinct strategies. Each design incorporates existing information security research and components, which are presented and explained to support the reasoning behind the design.

Sources for components and research are sought through a combination of:

- Study - Components covered in the Software and Systems Security taught MSc are selected for relevance.
- Search - Related terms are used to uncover further work using tools such SOLO[15], and Google Scholar.
- Word of mouth - The search is widened by reaching out to researchers and practitioners.

Each component, technique, or paper is evaluated in the context of its application to an aspect of a working DPS.

A number of designs are proposed. For each design:

- The premise is explained.
- Relevant information security research and components are presented.
- The design is described.
- Diagrams are presented to aid visualisation.

The first design, proposed in section 8.1, serves to illustrate a base case using a classic micro-services architecture. A full risk analysis is conducted, highlighting strengths and weaknesses in the design. This serves as a point of reference.

The designs described in sections 8.2 and 8.3 describe a distributed application, and an application of witness encryption (respectively). Both rely on the existence of one or more blockchains, comprised of nodes without constraints on hardware or operating system, and multiple versions and implementations of client software, which rapidly complicate a risk analysis. The boundaries of these systems encompass complex networks, and so a risk analysis of this magnitude is expected to exceed the limitations of an MSc thesis.

For these reasons, more pragmatic approaches to analysis are adopted. Each design is evaluated against the requirements for a DPS:

- Findings for each requirement are presented.
- Potential mitigations for found risks are offered.
- Further considerations are discussed.

[13] Some closed-source systems deliberately obfuscate their code and mechanism.

[14] Either exploring the code provided, or by reviewing the system's available documentation.

[15] Search Oxford Libraries Online - an index with access to all digitised documents held by libraries across the University of Oxford.

Finally, the proposed designs are compared to discuss strengths and weaknesses in the context of all proposed systems.

4.5.1 Risk analysis process

A qualitative method for risk analysis, scoring values as Low, Medium and High, is best suited to this thesis. This permits evaluation of the relative importance of risks without needing to know the costs or frequencies involved to a high degree of precision.

A **threat's** likelihood of success is calculated as a combination of a **vulnerability level**, and **threat capability**. This, in turn, is used to calculate the **threat level** in combination with frequency of attempts made by a given threat. Finally, the **risk level** itself is calculated from the **threat level** combined with the **impact** if it compromises an asset, as shown in figure 3.

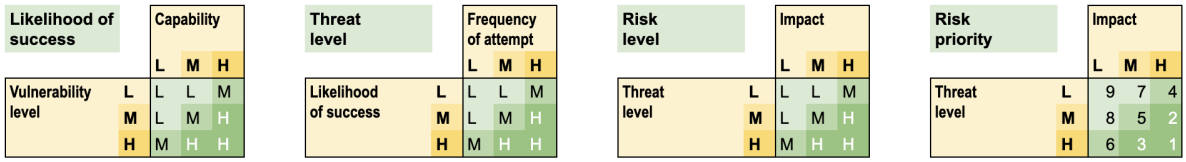


Figure 3: Risk analysis combination tables

- Likelihood of success = Vulnerability level x Capability
- Threat level = Likelihood of success x Frequency of attempts
- Risk level = Threat level x Impact valuation

These tables are unbiased - showing a balanced rule for combining these estimates. (Although the prioritisation table, far right, slightly favours impact over threat level.)

In the context of an organisation, they would be designed with the business and biased towards business priorities.

The result of this analysis is then used to select responses to each threat. A number of controls are presented for each threat.

5 Implementation - Stage 1: Gathering requirements

This section presents insights from the survey conducted, and combines them with reasoning to determine a number of requirements for a DPS. A data flow diagram and STRIDE-per-element threat analysis are used to evaluate a number of possible attacks against a DPS. Finally, stakeholders, assets, and threats are inferred and presented as the basis for a risk analysis in section 8.

5.1 Survey

As described in section 4.3.1, the survey presented here contains answers provided by investigative journalists, and seeks their opinions about if and how they would trust a DPS with their secret.

The participant information sheet and schedule of questions is included as Appendix A.

The survey is qualitative, seeking inputs that can help to describe user needs and inform reasoning about the use cases for a DPS.

5.1.1 Insights

As shown in figure 4, 68 participants viewed the opening description of the survey, and a further 5 undertook to complete the survey.

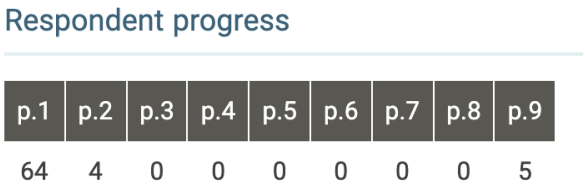


Figure 4: Survey respondent progress

A possible inference is that, despite precautions[16], many potential participants were also unable to trust the survey process itself to protect their anonymity.

5.1.1.1 Trusting an operator Results from the survey indicated that selecting and trusting an operator for a DPS is challenging for investigative journalists, with high standards to meet.

Such an operator will need to work hard to earn their trust. Answers ranged through requirements for transparency of motive, building an interpersonal relationship, and the security protocols and mitigations put in place to protect the system.

Table 6: Answers regarding trust of DPS operators

Q: How would you decide whether to trust the operator of a DPS?
"Transparency of funding, motives, background, and encryption"
"By meeting them in person if possible at least twice."
"Word of mouth/personal experiences with him"
"They would have to use the same security protocols as used by UK Police Forces and Interpol"

This caution is corroborated by advice found in *A Journalist's Resource for Safe and Ethical Reporting, Chapter 4: Digital Safety* [23], from the RSF - which offers advice for choice of digital communication tools.

A choice to make use of a DPS may rely on the ability to show why a subject should trust the service to keep their secrets reliably, or how they can trust the motives of the system's operator. This informs a property of a DPS: It should be possible for the target subject to understand and trust the system.

This is included as the **explainability** requirement in section 5.2.

[16] Precautions include: Ethical review from the University of Oxford, careful explanation of the anonymity protections provided, and choice of survey tool approved by the University of Oxford.

5.1.1.2 Systems already in use

Participants in the survey indicated overwhelmingly that they do not use any existing DPS systems.

Table 7: Survey answers regarding use of a DPS system

Question	Yes	No
Have you ever employed a DPS or similar system?	20% (1)	80% (4)
(If no.) Do you consider that you might if the need arose?)	50% (1)	50% (1)

This, combined with their answers to subsequent questions about how they might establish trust in a system, suggests that either:

1. investigative journalists do not perceive a need for such a system (perhaps because they do not believe that release of a particular secret is sufficient disincentive against attack), or
2. they do not trust any existing systems to provide the required resilience.

Both possibilities are plausible. The second is corroborated by a number of answers about the resilience of Dead Person Switches.

5.1.1.3 Resilience

The survey asked participants about the assurances they would need to know that a DPS was resilient against a number of different types of attack:

- Physical or digital assaults to make it unavailable
- Physical or digital assaults to retrieve its stored secrets or alter its behaviour
- Bribery or incentives to retrieve its secrets or alter its behaviour

In each case, answers varied - with many suggesting that it is impossible to build a fully resilient system:

> "both are always susceptible"
>
> "That's unlikely to be possible. It would always be a huge risk."
>
> "probably an impossible risk to take"

A few options were suggested by survey participants:

> "Written assurances"
>
> "A trusted programmer, perhaps."
>
> "Unlikely that either will be foolproof, ideally a decentralised system is much safer imo"

The application of decentralised systems to this problem is explored in section 6.2 (distributed apps).

This supports the **resilience** requirement in section 5.2.

5.1.1.4 Deterrent effect

Some threats to journalists are found to be extremely resilient against informational threats. Answers from the survey suggest that investigative journalists do not believe a DPS can serve as sufficient deterrent to an attack.

Table 8: Survey answers regarding deterrent effect of secrets

Question	Yes	No
In your professional opinion, can secrets obtained through whistle-blowing or other investigative means serve as a deterrent against reprisals?	0% (0)	100% (5)

For example, despite a UN finding that "the Kingdom of Saudi Arabia is responsible" for the "extrajudicial killing" of journalist Jamal Khashoggi, and that there is " 'credible evidence' to warrant an investigation into Prince Mohammed," Saudi Prince Mohammed bin Salman has successfully avoided investigation, trial or consequences [31].

This position contrasts with evidence that Wikileaks and Edward Snowden have both employed the threat of release of secret information as an "insurance" against attack - showing their belief that it could serve as an effective deterrent.

However powerful the deterrent effect is, this does inform a property of a DPS: It should be possible for a DPS to indicate that it is working and *could* release a secret, to support any deterrent effect.

This is included as the **visibility** requirement in section 5.2.

5.1.1.5 Cost One respondent indicated that the resilience required for a safe DPS would raise the cost beyond practical means:

> "I don't think the concept of a DPS is viable for many reasons... it would be a resource too expensive to use."

This informs an additional property of a DPS: It should be affordable for the target subject to establish and operate a DPS without compromising its security properties. This is included as the **affordability** requirement in section 5.2.

5.1.2 Failure states

There are two main ways a DPS can fail, although a number of different threats may have different motives for attempting to cause these failures:

Table 9: Failure states for a DPS

Fail state	Description
Early release	An attacker causes the DPS to release the secret before it is required. This may have a number of different effects on the physical safety of the user by removing their ability to disincentivise an attack.
Denial of service	An attacker causes the DPS to never release the user's secret, or to release it sufficiently late, so that it no longer offers sufficient disincentive to physical assault.

In each case, respondents to the survey indicated that the tool itself would lose the confidence of its users, and in both cases some answers suggested that the outcome could be serious:

> "It could compromise journalist and subject safety."

> "Potentially very serious"

> "The DPS involved would immediately lose the confidence of all clients... and be dead in the water as an untrusted system"

All answers regarding the risk of losing confidentiality of the DPS secret suggested high risk to the journalist and their source:

Q: What are the consequences of losing confidentiality of a secret stored in a DPS?
Very very serious, could blow cover of sources/risk of injury or death
Potential physical or digital harm to source and journo
Potentially very serious
It could compromise journalist and subject safety.

5.1.3 Summary

5 of 73 possible participants chose to complete the survey. This is a small sample, but offered a number of qualitative insights into the needs of investigative journalists and their perceptions of the risks related to a DPS.

Answers indicate a high level of caution from participants, requiring strong assurances that the tool they are using is resilient against attack, or suggesting that they could never trust a third party in this way. These answers inform a number of the requirements described in section 5.2.

5.2 Synthesised requirements for a DPS

Reasoning about the background, use case, and user needs for a DPS indicate a number of user stories. A DPS requirement can be inferred from each user story.

Table 11: Reasoned user stories, mapped to requirements

User story	Inferred requirement
As an investigative journalist, **I want** to keep my secret confidential from others, **so that** it is still valuable as a deterrent.	**Confidentiality**
As an investigative journalist, **I want** to be able to indicate to the system that I am still alive and well, **so that** it will not release the secret if I have not been attacked.	**Awareness**
As an investigative journalist, **I want** my secret released when I am unavailable (not before, not long after), **so that** there is a direct relationship between an attack on me and the release of the secret (supporting the deterrent effect).	**Timing**
As an investigative journalist, **I want** my secret to be stored and protected in a system that is resistant to attacks on its confidentiality, integrity or availability, **so that** it is available should it need to be released, and can be released without interference.	**Resilience**
As an investigative journalist, **I want** my secret to remain protected for as long as I am alive, **so that** no matter when, if I am attacked, it can be released.	**Durability**

Analysis of the survey, targeting investigative journalists, corroborates these and provides supporting evidence for several further requirements.

Affordability, **Explainability**, **Visibility**

Table 12: User stories inferred from the survey, mapped to requirements

User story	Inferred requirement
As an investigative journalist, **I want** to be able to afford to use a DPS, **so that** cost does not limit my use of this protective measure.	**Affordability**
As an investigative journalist, **I want** to understand the system and how it protects and releases my secret, **so that** I can develop confidence in it.	**Explainability**
As an investigative journalist, **I want** the system to indicate that it holds a valuable secret, **so that** threat actors believe it is real and active.	**Visibility**

These requirements are summarised in the following table with full descriptions.

Table 13: Requirements for a DPS

Requirement	Description
Confidentiality	A DPS must keep a user's **secret** confidential until it determines it is appropriate to release the **secret**.
Awareness	A DPS must have a mechanism to check the **subject's** well-being, and a method to release the **secret** if the subject fails this test.
Timing	A DPS should not release the **subject's secret** early (ie. whilst they are still alive and well), late (ie. too long after the subject has become unresponsive), or never.
Resilience	A DPS must assume the existence of, and protect against, attempts by hostile **threats** to compromise the **secret's confidentiality** and **integrity**, and be resilient against attacks intended to compromise its **availability** (CIA properties).
Affordability	A DPS should use affordable technologies to provide its functionality, not relying on resources beyond the means of the target subject group.
Durability	A DPS should be expected to remain operational for a significant amount of time (eg. the lifetime of a **subject**), with continued maintenance (including security patches), support, or upgrade pathways if the technologies in use become obsolete.
Explainability	A DPS must be presentable as a model that the target subject group can understand and trust.
Visibility	A DPS must be able to present evidence that it is operational, to contribute to its deterrent effect.

5.3 Threat analysis

This section presents reasoning about the stakeholders, components and threats in a high level generic DPS system. This provides a foundation for considering threats in section 8.

5.3.1 Stakeholders

The key stakeholders in a DPS system are the **subject** (owner of the secret), **operator** and **platform provider** - controlling the software and platform the switch runs on and, dependent on the design, a number of **participants**.

Table 14: Stakeholders in a DPS system

Actor	Role
Subject	Owner of a specific DPS. Inputs the initial secret, provides evidence of own well-being. Depends on a reliable release of the secret if they are not. Depends on the reliability of the DPS. Incentivised to protect their own safety, and to take steps to ensure the reliability of the DPS.
Operator	Owner of the DPS system itself, incentivised in some way to manage the resilience of the various components in the system.
Platform provider	Owner of the hardware and infrastructure that the DPS system operates on. Incentivised to provide a reliable operating system and hardware on which the DPS system can run.
Participant	(Optional. Required by some designs.) Member of a group tasked with determination of the state of the **subject's** well-being. Incentivised to protect their own safety, and the safety of the subject. Motivated towards appropriate behaviour in some way (eg. through loyalty to the subject, or a reward scheme).

NB. It is possible for the **subject** and **operator** (and sometimes also the **platform provider**) to be the same actor.

5.3.2 Components

By reasoning about individual functions that could comprise a generic DPS, a number of independent components can be inferred. Each component is responsible for one task, based on information provided by other components.

Figure 5 illustrates these components, in relation to each other, and describes their roles.

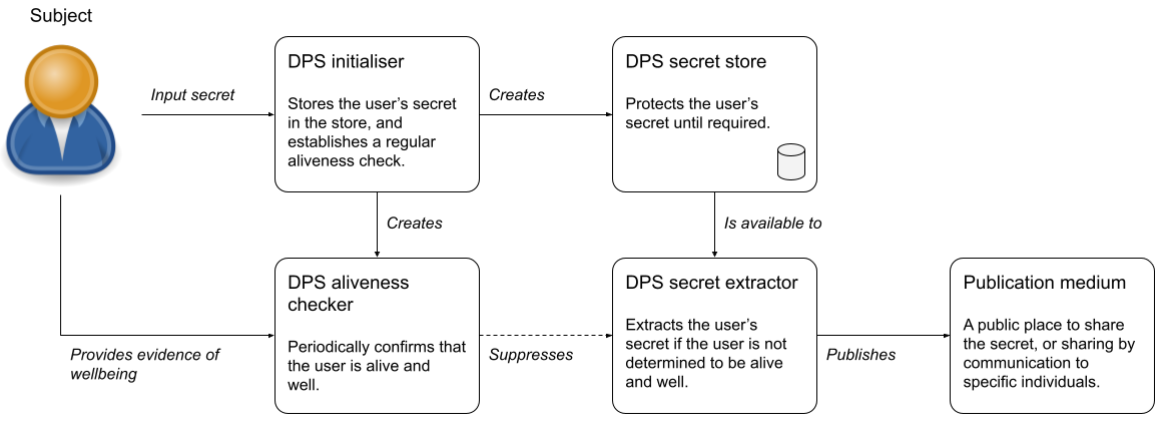


Figure 5: Components and roles for a DPS

Table 15: High level components in a generic DPS

Component	Role
Initialiser	Component that handles the **secret** initially provided by the **subject**. Arranges safe communication and storage of the **secret**, and establishes a method for the DPS to check the well-being of the **subject**.
Secret store	Component that stores the **subject's secret** in such a way that it can be extracted only when the right conditions are met.
Aliveness checker	Component that can monitor **evidence** of the **subject's** well-being, and is able to activate or suppress the **secret extractor** dependent on the **subject's** state.
Secret extractor	Component that can retrieve the **secret** from the **secret store**, and share it to the **publishing medium**.
Publishing medium	Either a public medium where the **secret** can be published, or direct communication of the **secret** to specific recipients.

Stakeholders and components as described above may change as the model becomes more specific. Cryptography, distributed networks, or groups of people could be substituted for some of the roles, functions or incentives.

A high level data flow diagram and threat analysis follows to examine the various ways each component can be compromised, and the impact of these failures.

5.3.3 High level data flow diagram

Figure 6 illustrates a generic DPS as a data flow diagram, modelled using Threat Dragon.

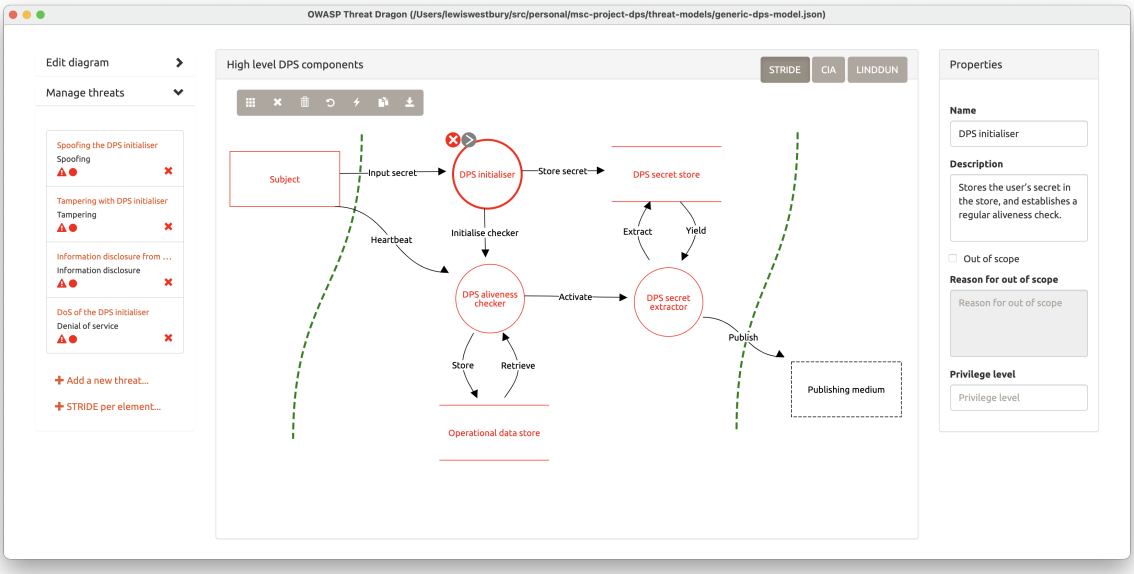


Figure 6: Generic DPS data flow diagram, modelled in Threat Dragon

5.3.4 STRIDE-per-element threat analysis

The threats that apply to this model are listed in the report generated from the high level data flow diagram using Threat Dragon. The output from Threat Dragon is included as Appendix B.

The high level model represents generic components. This facilitates a quick analysis of the various ways each component can be compromised, and the impact of these attacks.

This analysis reveals a number of areas of significant vulnerability. These will require special consideration when evaluating designs.

Table 16: Areas of significant vulnerability, generic DPS components

Component	Spoofing	Tampering	Repudiation	Info. disc.	DoS	Elevation
Subject	×					
Initialiser	×	×		×	×	
Secret store		×		×	×	
Aliveness checker	×	×			×	
Secret extractor		×				×

The impact of compromise of each of these components is discussed in the Threat Dragon model, Appendix B. Some key risks are summarised below.

If either the subject or aliveness checker is successfully spoofed, the aliveness check may fail, as either:

- an attacker can force a user to report their well-being to the wrong place (causing an early release), or
- an attacker can provide false proof of well-being (causing a denial of service or late release).

Table 17: Spoofing attacks on the subject and aliveness checker

Spoofed element	Early release	Denial of service
Subject		An attacker spoofs the subject and submits false evidence of their well-being, preventing release.
Aliveness checker	An attacker spoofs the aliveness checker, causing the subject to submit their evidence of well-being to the wrong place, leading to early release as the real aliveness checker believes the subject to be unavailable.	

Internal components with no public-facing endpoints (the initialiser, secret store, secret extractor) are less at risk of spoofing, but may face tampering, information disclosure, or denial of service attacks - effectively causing the switch to fail to take an action, or disabling the aliveness check.

In the absence of an aliveness check, specific designs of switches may have their own preferred action. Some designs will consider an attack on themselves the same as an attack on the subject, and so release their secret. Others, that manage many subjects and secrets, may not know which secret to release.

The other key threat relates to the secret extraction capability. If attacked, this may disable the switch - as even if the attack were detected, without the secret extractor the switch cannot release the subject's secret.

An escalation of privileges within the secret extractor could also result in the early release of the subject's secret, or potentially all_the secrets held in the secret store.

5.4 Risk analysis foundation

A comprehensive risk model for a given DPS depends on its specific implementation. However, it is possible to reason about a number of properties upfront.

This section presents inferred assets, stakeholders, threat actors, and some reasoning about the impact of compromise of those assets.

5.4.1 Informational assets

The informational assets of a generic DPS can be inferred from the components. A DPS must process and protect a number of different informational assets, listed here:

Table 18: Informational assets in a DPS

Asset	Name	Description
A001	Subject identity	A representation of the identity of the subject, used to evaluate the evidence of well-being presented, to ensure it matches the identity of the user that owns the DPS.
A002	Secret	Representation of the subject's secret. Stored and protected by the secret store.
A003	Secret extraction information	Information that may be used to extract the secret - eg. a decryption key.
A004	Operational information	Additional information relating to the operation of the DPS - eg. a period for recurring aliveness checks, chosen behaviour should the checks fail (eg. a cooling-off period before activating the secret extractor, and information about the Publishing medium).

More specific implementations for a DPS may introduce additional informational assets required to check the **subject's identity**; manage storage, encryption and decryption of the **secret**; or for onward publication.

Some techniques (for instance, application of forms of time-delay or witness encryption) could also remove the need for a DPS to store and protect **secret extraction information** - instead relying on work done by *any* actor to release the secret at the appropriate time.

5.4.2 Known threat actors

Knowledge of the threat landscape, and historical risks to investigative journalists, helps to inform a set of actors that threaten the informational assets managed by a DPS.

No single list of threat actors exists, and no single scale exists to help assign values for the capability of each - as these values are dependent on context. Here capabilities have been reasoned, and assigned according to a simple scale:

- **Very high:** Threat actor has access to unlimited resources, expertise, and time; may not be constrained by law enforcement.
- **High:** Threat actor has access to top of the range consumer resources, a department of experts, and unlimited time. May be confident in avoidance of law enforcement, and may have access to strong political influence.
- **Medium:** Threat actor has access to standard consumer tools, a degree of training, and determination.
- **Low:** Threat actor has access to standard consumer tools, limited resources, and limited training.

Table 19: Known threat actors

Threat actor	Description	Capability
Nation states (extra-legal means)	Nation states and their intelligence agencies are considered to have limitless resources with which to attack systems - making 0-day vulnerabilities, and expensive physical or social engineering attacks available.	Very High
Nation states (legal means)	Nation states may choose to subject the operators or platform providers of DPS services to legal pressure to halt their service or reveal stored secrets.	Very High[17]
Organised crime	OC gangs may have various motivations for attacking a switch - either to prevent information about themselves being released, or to cause the release of information about their own adversaries. Criminals may also act unpredictably, displaying behaviours such as revenge.	High
Activists	Some activists may not approve of practices that withhold incriminating information for any reason. They may perceive it as cowardly, irresponsible, or even blackmail. To uncover and release this information, activists may use technical or social engineering attacks. Activism is not believed to be as well funded as organised crime or national security organisations.	Medium
Law enforcement	Considered separate from nation states, law enforcement bodies that perceive DPS activity as blackmail (or another activity at odds with the law) may attempt to shut down such systems, or arrest those operating it. Some law enforcement bodies have a limited range of legal activities that they can engage in, or may be called upon to enforce decisions made by local justice systems.	Medium

Threat actor	Description	Capability
Script kiddies	A high volume threat on the internet, script kiddies are people who, for reasons such as boredom, curiosity, or malice, continuously test endpoints for vulnerabilities (without much thought to the nature of the endpoint) - exploiting them when found.	Low

Assumptions:

- This list does not contain mention of internal threat actors, such as disgruntled staff. Specific implementations that rely on an operator[18] may need to add additional threat actors.
- Competing organisations (such as rival journalism outlets) are not included in this model, and not considered a threat[19].

5.4.3 Impact to informational assets

Information from the survey, threat model, and reasoning inform an understanding of the impact of compromise to each asset's CIA[20] properties:

Impact level is difficult to measure, as there are very few cases where loss of CIA properties of a DPS do not lead to loss of the deterrent effect (so possibly enabling an attack on a subject). The following scale is used:

- **High:** Could lead to the loss of deterrent effect, enabling an attack on the subject.
- **Medium:** Could be used to enhance an attack on a subject or DPS. The secret remains safe.
- **Low:** Undesirable, but does not compromise the reliability of the DPS, or confidentiality of the secret.

Table 20: Impact to informational assets

Asset	Asset name	CIA	Impact	Level
A001	Subject identity	C	Loss of confidentiality of the subject's identity may represent a risk, as the subject may have chosen not to reveal the details of their switch (or how to locate it), to prevent it becoming an attack surface. However, to act as a deterrent, the existence of a switch must be declared - and so the positive angle is that this information may constitute evidence that the switch is operational.	Medium
A001	Subject identity	IA	The subject identity is essential for determining the well-being of the subject. If this is altered or made unavailable, the switch will not have enough information to determine the state of the subject. In this case, it must follow the safest course of action. This is not a trivial decision.	High
A002	Secret	C	Loss of confidentiality of the secret held in the switch itself can have a high impact. If the secret is released early, it can no longer be used as a disincentive to physical attack. It may also inform the attacker just how much (or how little) impact the secret could have, or may allow them to put mitigations in place to plan for the secret's publication. These all weaken the switch as a protective measure.	High
A002	Secret	IA	Alteration or deletion of the secret is a powerful attack and this grants an attacker the ability to 'defuse' the switch entirely, removing its disincentive to physical attack, and rendering it useless.	High
A003	Secret extraction information	CIA	Confidentiality, Integrity, Availability | The information required to extract the secret from the switch should remain entirely under the control of the switch. If this information becomes known, it offers insights into the secret itself (see the impact of loss of confidentiality for the secret). If it can be tampered with, it directly affects the availability of the secret too. If the switch loses the ability to extract and publish the secret it holds, it is rendered useless.	High

[17] Nation states are considered to have very high capability where they have jurisdiction.

[18] This is the first suggestion that some designs may not rely on an individual or organisation to administer the DPS. It is explored further in section 6.2.

[19] Large communities of investigative journalists exist that transcend organisational boundaries. It is reasonable to assume that the notion of harming another investigation is sufficiently abhorrent to the community that this is a negligible risk.

[20] CIA: **C**onfidentiality, **I**ntegrity, **A**vailability

Asset	Asset name	CIA	Impact	Level
A004	Operational information	C	Operational information, such as the frequency and chosen publishing details of the switch are of some value to an attacker. They show the intended frequency of aliveness checks, and details of the recipients (or medium of publication). In turn, this information could be used to improve the quality of an attack against the switch, subject, or recipients - increasing the risk that individuals involved will be attacked, or the switch disabled. However, as with the subject identity, releasing information to the effect that the switch is operational may contribute to the switch's deterrent effect.	Medium
A004	Operational information	IA	Ability to alter or remove the operational information could render the switch inoperable (or impractical - for instance, if the period for aliveness checks were increased beyond a desirable limit).	High

The impact of loss of CIA properties of all information assets is evaluated as **high** except for confidentiality of the metadata assets (**subject identity** and **operational information**), both evaluated at **medium**.

As seen when considering the confidentiality of **subject identity** and **operational information**, some thought should be given to a means of publicising the switch's active state (if not the details) to support the switch's deterrent effect. This should be balanced against the risks of advertising the switch's location, and so presenting a visible attack surface.

6 Implementation - Stage 2: Review of existing systems

This section presents a number of existing systems, each of which purports to meet the requirements of a DPS:

Table 21: Existing systems reviewed

Section	Category
6.1	Hosted consumer DPS systems
6.2	dApp DPS implementations
6.3	Open source DPS projects

Each system is analysed in the context of the requirements for a DPS, established in section 5.2 and reproduced here.

Table 22: DPS requirements reproduced

Requirement	Description
Confidentiality	A DPS must keep a user's **secret** confidential until it determines it is appropriate to release the **secret**.
Awareness	A DPS must have a mechanism to check the **subject's** well-being, and a method to release the **secret** if the subject fails this test.
Timing	A DPS should not release the **subject's secret** early (ie. whilst they are still alive and well), late (ie. too long after the subject has become unresponsive), or never.
Resilience	A DPS must assume the existence of, and protect against, attempts by hostile **threats** to compromise the **secret's confidentiality** and **integrity**, and be resilient against attacks intended to compromise its **availability** (CIA properties).
Affordability	A DPS should use affordable technologies to provide its functionality, not relying on resources beyond the means of the target subject group.
Durability	A DPS should be expected to remain operational for a significant amount of time (eg. the lifetime of a **subject**), with continued maintenance (including security patches), support, or upgrade pathways if the technologies in use become obsolete.
Explainability	A DPS must be presentable as a model that the target subject group can understand and trust.
Visibility	A DPS must be able to present evidence that it is operational, to contribute to its deterrent effect.

6.1 Hosted consumer applications

A number of consumer services exist that claim to meet some of the requirements of a DPS. Some come with a number of disclaimers regarding the use cases they suggest for their users.

The following services are presented in this section, and evaluated in the context of the requirements for a DPS, established in section 5.2:

Table 23: Hosted DPS systems evaluated

#	Product	Summary
1	Dead Man's Switch	A system designed to meet end of life needs - such as messages for loved ones. It checks for signs of life, and if not met, sends a number of emails to chosen recipients.
2	Dead Man	A system that checks for a response, and distributes documents by email if it does not receive that response.
3	Dead Man Tracker	A service that automatically contacts your friends and/or family in the event that something happens to you.
4	Letters Cloud	A service that allows you to create messages to be sent, should you stop visiting your trigger link. (formerly komprom.at)

These applications are then compared and contrasted to determine if they meet sufficient needs for the investigative journalism use case.

6.1.1 Properties of consumer applications

Each of these systems takes responsibility for the safe storage and distribution of the user's secret. This makes each service a single point of failure for confidentiality, integrity and availability properties.

Most of these services do not document their design, and this makes it difficult to understand the risk a person takes when trusting it with their secret. These systems face a number of threats, ranging from casual hackers, through organised crime, motivated attackers (such as those named in the user's secret), and extremely well-resourced threats such as nation states. Users must make a trust decision about the service - both regarding its intent to keep their secret, and its capability to do so.

6.1.2 Hosted DPS 1: Dead Man's Switch

Table 24: Hosted solution summary: Dead Man's Switch

	Dead Man's Switch
URL	deadmansswitch.net
Privacy policy	N/A
Released	2017
Last social post	2020-06-13
Pricing model	Freemium[21]
Aliveness check channels	Email, push notification, Telegram
Publishing medium	Email

Dead Man's Switch, shown in figure 7, is a service provided by Stochastic Technologies.

6.1.2.1 Mechanism The user creates a number of messages to be emailed to individuals if they do not respond to prompts sent through the supported channels. The user responds to prompts by clicking a unique link to activate a check-in on the site. (Signing in to the site is also considered proof of aliveness.) If the user misses a predetermined number of prompts, their messages are sent.

> "Your switch will email you every so often, asking you to show that you are fine by clicking a link. If something were to... happen... to you, your switch would then send the emails you wrote to the recipients you specified. Sort of an 'electronic will,' one could say."

6.1.2.2 Use cases Content on the site states that the recommended use case is for end of life needs, such as a message to loved ones. The site also contains a number of disclaimers:

> "Dead Man's Switch is provided without any guarantees of anything, not even that it will do its job properly"

> "this service is meant for casual use by the average person. Please don't use the service if you need strong guarantees of privacy, e.g. if you are a whistleblower or any similar life-and-death situation. It is NOT meant to safeguard against high-value messages."

This suggests that the service is designed with casual threats in mind. It is an acknowledgement that it may prove vulnerable to well-resourced threats. Stochastic Technologies are based in the Cayman Islands, and subject to UK law.

6.1.2.3 Evaluation Details of the design and technologies supporting Dead Man's Switch are not available, however the information available does allow reasoning about the service in the context of requirements:

[21] A freemium model offers basic services for free, attaching a charge for advanced features.

Figure 7: Hosted DPS 1: Dead Man's Switch (deadmansswitch.net)

Table 25: Requirements vs Dead Man's Switch (hosted DPS 1)

Requirement	Comments
Confidentiality	It is reasonable to assume the secret is treated confidentially. However even if it is encrypted, the service retains a key that can be used to decrypt it - (notwithstanding issues of configuration, about which information is not known) making it confidential to all except the service itself, and any system administrators with access to the service's keys. No privacy policy is available.
Awareness	The system can check aliveness through a number of channels: web push notification, telegram, email, sign in. This improves the quality of the aliveness check, but represents an attack surface for threats wishing to falsify signs of life.
Timing	A disclaimer states that the switch does not guarantee "anything, not even that it will do its job properly." Whilst likely written this way as a legal defence, this reduces confidence that the switch can reliably meet this requirement. No data on uptime, and no SLA are provided.
Resilience	The disclaimer on the site clearly states that the service is not intended for high-value messages. The switch is unlikely to prove resilient to well-resourced threats to the confidentiality of the secret, integrity or availability of its service.
Affordability	The switch offers a free tier, and a $50 single payment premium service. Both are considered affordable to a journalist or an organisation they work for.
Durability	The only paid tier is a lifetime single payment. This means that the service must steadily gain customers to balance income against maintenance costs. If new membership slows sufficiently, the switch may become expensive and the operator may choose to stop the service.
Explainability	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.
Visibility	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.

6.1.3 Hosted DPS 2: Dead Man

Table 26: Hosted solution summary: Dead Man

	Dead Man
URL	deadman.io
Privacy policy	N/A
Released	2012
Last social post	2016-01-13
Pricing model	Free
Aliveness check channels	Email, SMS, phone call
Publishing medium	Email, SMS, phone call

Dead Man, shown in figure 8, was created by Jesse Lovelace for the LSRC Hackathon in 2012.

6.1.3.1 Mechanism Dead Man stores a number of email recipients, messages, and files. It contacts the user at predetermined intervals by email, SMS, or phone call and if they do not respond it will then send the stored messages and files to the chosen recipients.

6.1.3.2 Use cases The site advertises a number of use cases for its service:

- Whistleblowers
- Wilderness excursions (missing persons)
- Shut-ins
- End of life needs
- The unknown

If taken at face value, these use cases might indicate a design that's intended to be resilient against sophisticated threats (eg. for whistleblowers). However, this should be balanced against knowledge that the service was created at a 2012 hackathon, and should be considered more a proof of concept.

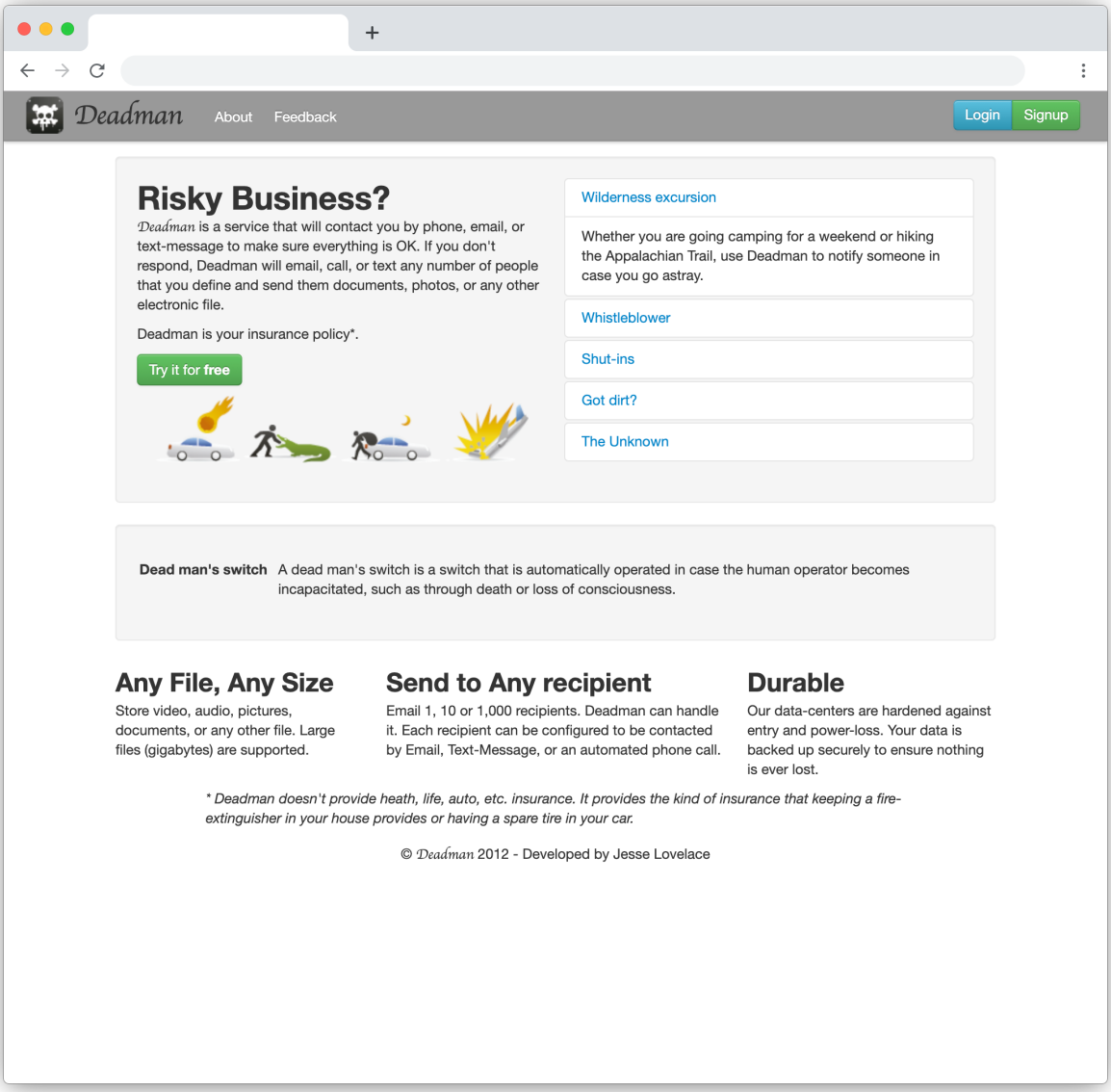


Figure 8: Hosted DPS 2: Dead Man (deadman.io)

6.1.3.3 Evaluation

Dead Man relies on a number of technologies:

- Hosted on Google App Engine
- Written in Python
- Integrations with Twilio & SendGrid (providing SMS, phone call, and email capabilities)

The service comes with a disclaimer regarding its use of the word insurance:

> "Deadman doesn't provide heath, life, auto, etc. insurance. It provides the kind of insurance that keeping a fire-extinguisher in your house provides or having a spare tire in your car."

Attempts to sign up using syndicated Google account were met with a warning (see: figure 9), suggesting that the service has not been updated since changes to account integration were introduced. Account creation through all the offered means failed.

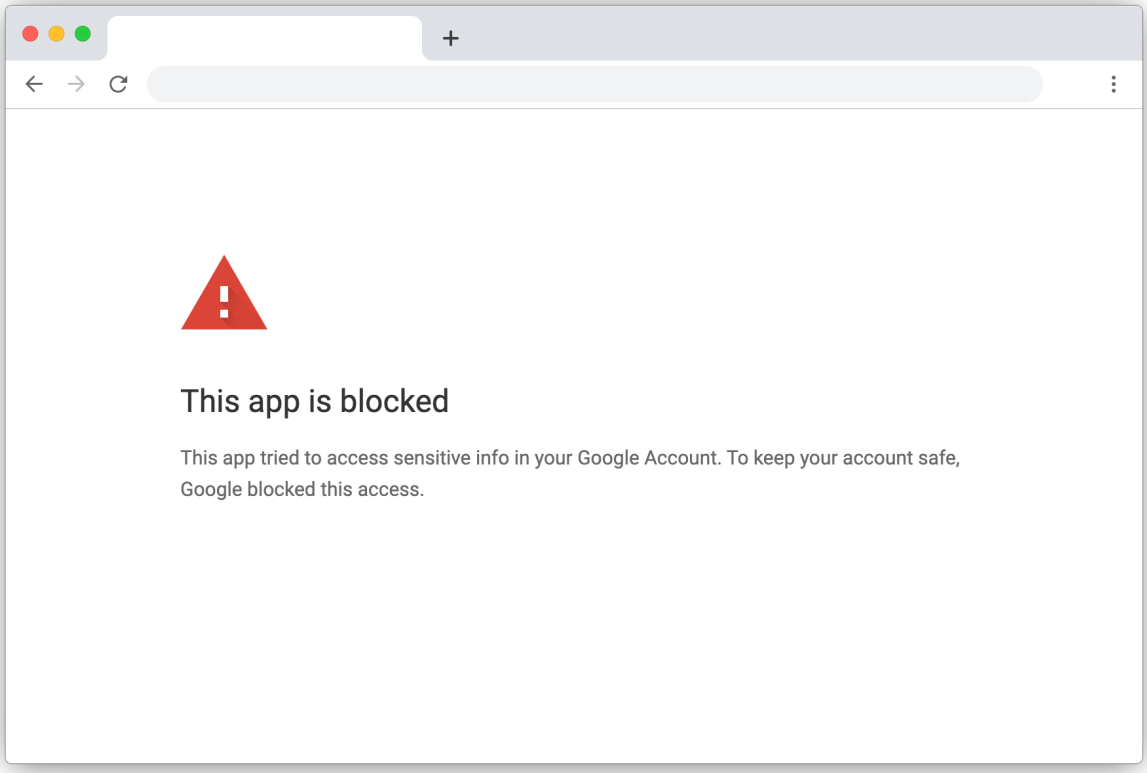


Figure 9: Dead Man service (hosted DPS 2) blocked

Despite this the service, if it were as described, should be able to meet some of the requirements.

Table 27: Requirements vs Dead Man (hosted DPS 2)

Requirement	Comments
Confidentiality	The system is hosted on Google App Engine, and protections for that platform apply (dependent on good configuration). Although Dead Man does not have a privacy policy, Google's is substantial and GDPR compliant. It is reasonable to assume that, notwithstanding a misconfiguration[22], the system is designed to restrict user secret visibility to the service and its administrators.
Awareness	The system can check aliveness through a number of channels, supported by Twilio and SendGrid: email, SMS, or phone.
Timing	The system is based on reliable technologies (Google App Engine, Twilio, SendGrid) and so is expected to be able to deliver messages reliably.

Requirement	Comments
Resilience	Although composed from several services considered to be resilient (Google App Engine, Twilio, SendGrid), the service itself was developed at a hackathon and is likely best considered a proof of concept. No information about a threat model is available, and so it is reasonable to assume the service is not resilient to well-resourced threats.
Affordability	The site advertises a free service.
Durability	The service was created in 2012, and does not seem to have been updated for a while (as indicated by the fact that it is no longer possible to sign in or create accounts through the various mechanisms offered). With no apparent source of funding, it is likely that after initially creation, this service has not been maintained.
Explainability	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.
Visibility	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.

6.1.4 Hosted DPS 3: Dead Man Tracker

Table 28: Hosted solution summary: Dead Man Tracker

	Dead Man Tracker
URL	deadmantracker.com
Privacy policy	deadmantracker.com/privacy-policy
Released	2019
Last social post	2021-05-19
Pricing model	Freemium
Aliveness check channels	Email, app push notifications
Publishing medium	Email, SMS, phone

Dead Man Tracker, shown in figure 10, operates as an inexpensive consumer service. The operator responded to requests for information, and this informs the following commentary.

6.1.4.1 Use cases Initially created for the operator, as a safety device to notify friends and family of accidents when travelling, the service has since evolved to incorporate additional use cases:

- Professionals working alone (such as roofers)
- Elderly people going on walks alone
- Parents that want their children to check in
- People walking home after a night out

Additionally some domestic abuse charities have asked the operator for a discreet method, and the mobile app provided has a mode that appears to be a game with a hidden panic button for victims.

6.1.4.2 Mechanism Users can respond to emails or push notifications from the service (if they have the app installed). If they do not, it sends messages to chosen recipients by email, SMS or phone call.

6.1.4.3 Evaluation Dead Man Tracker relies on the following services:

- Hosted on Amazon Web Services (AWS)
- Stripe / Google Pay / Apple Pay for payments

It appears to be maintained, and the operator is responsive.

[22] For example, in 2020 thousands of Android applications were found to be leaking data through misconfigured Firebase databases [32].

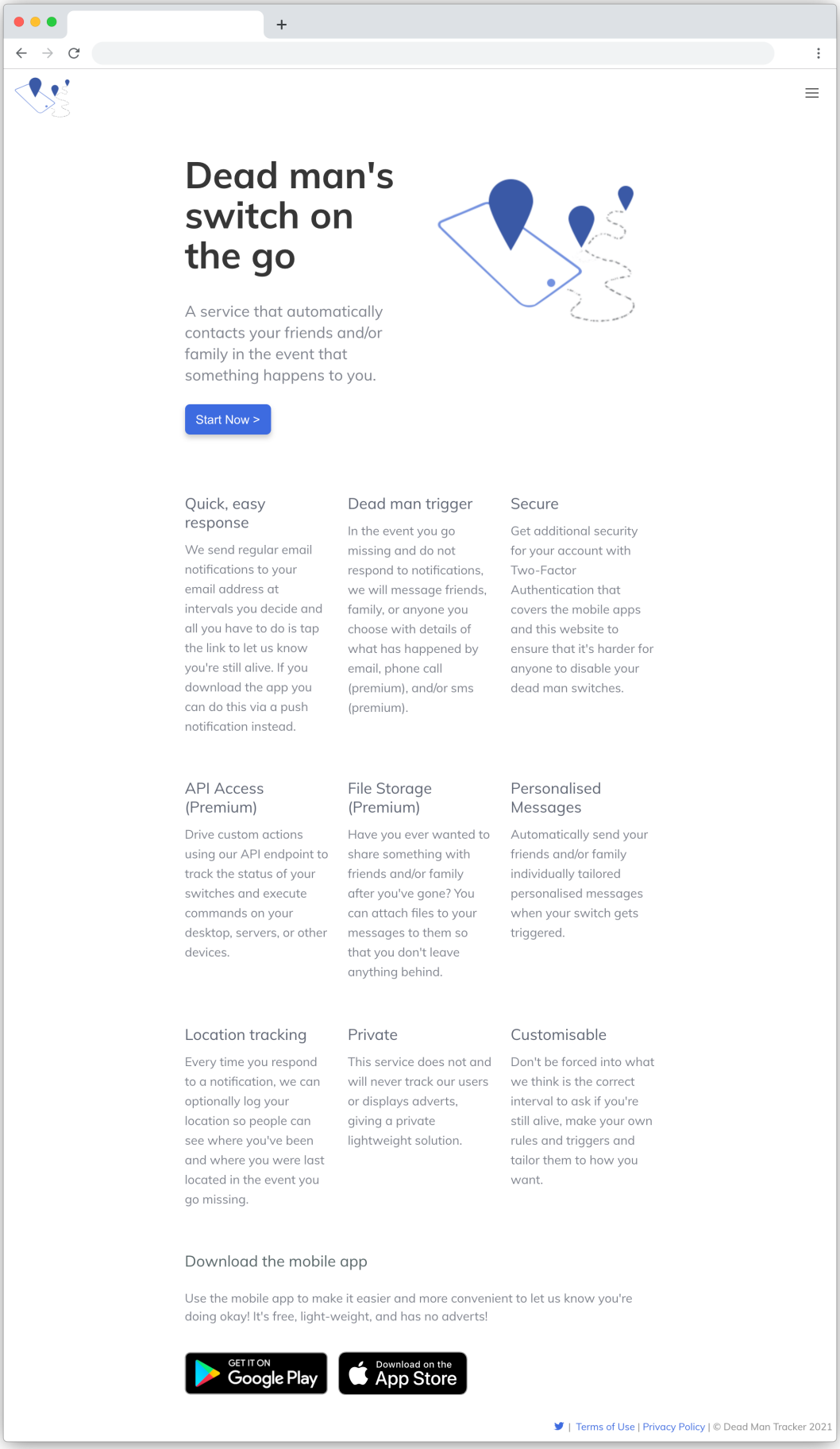


Figure 10: Hosted DPS 3: Dead Man Tracker (deadmantracker.com)

Table 29: Requirements vs Dead Man Tracker (hosted DPS 3)

Requirement	Comments
Confidentiality	This is the only system surveyed that has a full privacy policy. The system is hosted on AWS - and protections for that platform apply (dependent on good configuration). Data is encrypted at rest, although the service must retain a key to decrypt it - making the user's secret confidential to all except the service itself, and any system administrators with access to the service's keys.
Awareness	The system can check aliveness through email or push notification.
Timing	As an AWS hosted service, the system is based on reliable technologies, so is expected to be able to behave reliably.
Resilience	The creator has shared priorities for a threat model in response to questions, and has placed high value on system availability, and resilience to casual threats. Hosting on AWS provides that high availability, and the design has separated the trigger mechanism from the rest of the service - to prevent a denial of service attack on the website from affecting delivery of secrets. The creator acknowledges some personal risk, noting that it is probably easier to attack the operator or individual users of the service, rather than the infrastructure.
Affordability	The service has a free tier (offering a single switch), or a $10/year subscription service with unlimited switches, file storage, API access, and multiple publishing mediums. Both are considered affordable to a journalist or an organisation they work for.
Durability	The service is currently maintained, and the creator is responsive. The subscription model allows for time and costs to maintain the service.
Explainability	The existence of a privacy policy goes some way towards documentation that a subject could use to determine whether they trust the system with their secrets. This system is *assumed* to follow a classic design. Communication with the operator revealed that it is hosted on AWS, and that subject data is encrypted at rest. However, as with other hosted solutions the architecture and code is not published - making it difficult to evaluate.
Visibility	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.

6.1.5 Hosted DPS 4: Letters Cloud

Table 30: Hosted solution summary: Letters Cloud

	Letters Cloud
URL	letters.cloud
Privacy policy	N/A
Released	2020
Last social post	N/A
Pricing model	Freemium (implied)
Aliveness check channels	Passive (trigger link URL)
Publishing medium	Email, Twitter

Originally known as komprom.at, little is known about Letters Cloud, shown in figure 11. The site states it was created by "a team of security professionals, programmers and web developers who like privacy."

The operator, going by the name of Anomia Security, did responded to requests for information - but provided only a little information:

- They chose not to share any information about the system's architecture.
- They indicated that the system is compartmentalised, giving it resilience to denial of service attacks and exploits: "There was never a successful attack."
- They indicated that they had chosen to remain anonymous as a measure to protect themselves from attack as individuals, and acknowledged that this limited the credibility of the app.

6.1.5.1 Use cases The site itself simply states that it can be used to send messages "when you can't do it in person any more, for any reason."

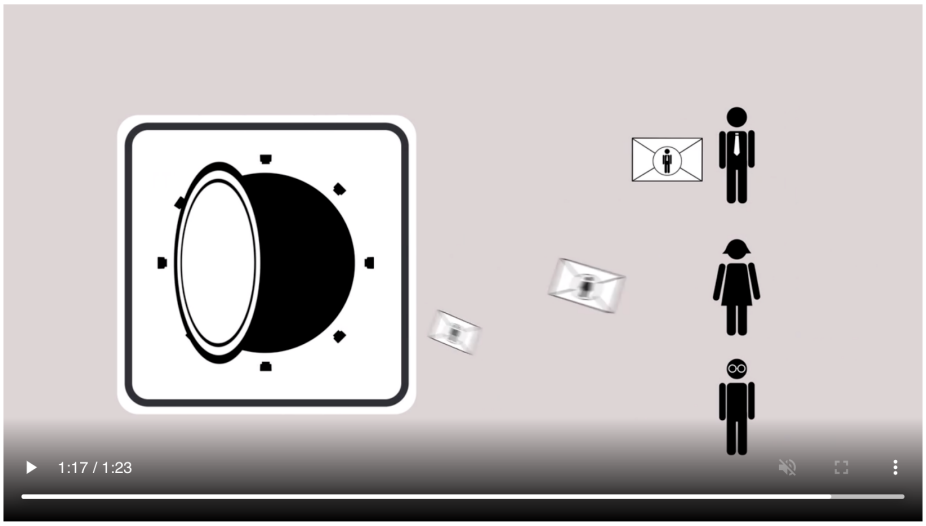
In a description on its website, it mentions two possible scenarios:

- A mechanism to pass on messages to loved ones after death.

Register Log in Contact

Letters Cloud
==========

This is a place where you can leave a digital message to be delivered to someone when you can't do it in person anymore, for any reason.



Introduction

You might have seen this in a movie: someone lost a family member or a loved one. After the funeral, a stranger approaches them and passes a sealed envelope with a message, or they receive that message online, from a secret server.

Or maybe that one, where the bad guys catch the hero and threaten him to give up the secret he/she has stolen? And the hero replies that he/she had stored the secret somewhere on a server and it will be posted all over the Internet if the bad guys kill the hero?

This is the place, this is the secret server.

How does it work?

You can leave your messages to be sent in four easy steps:

1. Create your account. You can also use your social media accounts or Yubikey to register an account!
2. Leave your messages at your account.
3. You account gives you a special URL address called **trigger link**. Set it as the home page of your phone or desktop browser.
4. As long as you visit your trigger link regularly, the messages won't be sent.

This is the four steps you have to do. The fifth step is done automatically.

5. The messages will be sent after you stop visiting your trigger link, and the extra time margin specified in your account settings passes.

Basic account is free and allows you to send one e-mail message and one Tweet.

Privacy

We don't care about your name or where you are from. You can pay using Bitcoin. You don't have to give up your email address.

We don't use any tracking technology on the page ale the only cookies we use are session and form protection cookies that are not bound to your personal data. We do not collect any personal data about you. And we don't share with anyone.

Note for TOR users: we tried to avoid that as much as possible, but you need to enable javascript for this app.

About us

We're a team of security professionals, programmers and web developers who like privacy.

Figure 11: Hosted DPS 4: Letters Cloud (letters.cloud)

- A whistleblower, who uses the service to disincentivise an attack on them.

6.1.5.2 Mechanism The user defines messages to be sent to a particular recipients. The service gives the user a URL (a *trigger link*) they should visit regularly (by making it their home page in a web browser, for example). If they do not, the service then sends messages to the specified recipients by email or tweet.

This is the only consumer service reviewed that works passively in this way, allowing users to send them a periodic signal rather than prompting for signs of life through traditional communication channels.

6.1.5.3 Evaluation The security consciousness of the operators, about whom very little is known, might be considered reassuring. In this case, however, it also makes it difficult to learn about the security properties of the service - and this could make it difficult for journalists to trust the operators.

Table 31: Requirements vs Letters Cloud (hosted DPS 4)

Requirement	Comments
Confidentiality	Not a lot of information is available. A short statement on the site's page indicates how little information is gathered by the system. Developers have taken pains to ensure their own identities are not listed. Whilst indicative of respect for privacy, this could make it more difficult to trust the system's operators.
Awareness	The user is given a URL to visit regularly, called the trigger link.
Timing	Nothing is known about the service's reliability with regard to timing, but the description suggests that a missed check-in followed by a determined amount of time will result in messages being sent.
Resilience	The trigger link given to the user has interesting security properties. Instead of regularly reaching out to the user with a reminder the service expects the user to visit the trigger link. The trigger link is considered a secret to be kept and used by the user. This alters the challenge for an attacker, making it more difficult to learn about the user's activity with Letters Cloud - as common communication methods (email, SMS, etc.) are not used. Other information about the service's security properties isn't available.
Affordability	The service provides a free tier, and content suggests a freemium pricing model - however no information about the cost of the premium service is available, other than the intention to make it possible to pay with cryptocurrency.
Durability	The service seems to have fallen into disrepair. Difficulties experienced signing up to the service make it difficult to test.
Explainability	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.
Visibility	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.

6.1.6 Comparisons

These implementations each meet different combinations of the requirements.

6.1.6.1 Confidentiality Some services fared better than others. Only Dead Man Tracker offers a privacy policy (although Letters Cloud have a strongly worded sentence or two about respect for privacy). In these cases, it is difficult to learn more (and so difficult to develop user confidence) without access to designs for the systems.

6.1.6.2 Awareness Some services offer a wide range of channels for aliveness checks. This can improve the quality of the check (and assure a user that they can find a convenient method). However, there is a balance to be found: More channels make it harder to cause early release through denial of service, but also represent an attack surface for threats wishing to falsify signs of life and cause a denial of service.

Letters Cloud is unique in offering a passive means of communicating aliveness. Whilst possible to spoof a user's aliveness by learning their trigger URL, an attacker must do so by compromising the user's personal device, obtaining a record of their web browsing history, or sniffing their traffic.

6.1.6.3 Affordability All services reviewed fell into the affordable category - either having a free service, or having what is determined to be a low cost premium solution.

6.1.6.4 Durability Of the services reviewed, only Dead Man's Switch and Dead Man Tracker are operational. Both services have a freemium model. Dead Man's Switch offers a single lifetime payment as the premium tier, whereas Dead Man Tracker operates a subscription payment. Services with a subscription payment model are more likely to endure, having incentivised themselves to keep the service operating (and having made it possible to continue paying for hosting, patching, and software services).

6.1.6.5 Resilience Neither operational service has rated themselves as suitable for use in situations with sophisticated attackers (such as the investigative journalist use case). The operator of Dead Man Tracker notes in a response to questions that individuals are likely easier to attack than the service itself - and that attempting to attack and deactivate a subject's switch is easier than attempting to disrupt the service itself.

6.1.6.6 Explainability and Visibility With the exception of Dead Man Tracker's privacy policy, these systems offer very little to support either of these requirements.

6.1.7 Summary

Evaluation and comparison of these solutions suggests that none of the consumer offerings are appropriate for the investigative journalism use case.

The solutions above are summarised in Appendix C table 01, evaluated in table 04, and scored in table 05. These scores are reproduced in section 6.4.

6.2 Distributed Apps (dApps)

A number of distributed apps have been proposed that meet some of the requirements for a DPS system.

The following are presented in this section, and evaluated in the context of the requirements for a DPS, established in section 5.2:

Table 32: dApps evaluated

#	System	Service description
1	KillCord	A tool to build resilient dead person switches for releasing encrypted payloads.
2	Kimono	A digital time capsule built on the Ethereum blockchain.

Although Kimono is a time-lock solution, it is included for discussion as it could be adapted for use as a DPS.

6.2.1 Properties of dApps

Cryptocurrency networks distribute smart contracts for execution across their network, relying on consensus to determine the result. For example, the Ethereum cryptocurrency network will execute smart contracts written in a number of languages (Solidity, Vyper, Yul are supported) in an environment known as the Ethereum Virtual Machine (EVM).

The decentralised nature of these networks impacts a number of security properties of any system that relies on them.

- The size of cryptocurrency networks and use of consensus gives smart contracts strong availability and integrity properties.
- To prevent them from executing correctly, an attacker must attack the whole network and gain control of a majority share of the nodes.
- However, in their current form, smart contracts are not equipped to hold secrets as the security properties of the EVM cannot be guaranteed across an untrusted network.

These properties are discussed in detail in section 7.4.2.2.

6.2.2 dApp 1: KillCord

Table 33: dApp 1 summary: KillCord

	KillCord
URL	killcord.io
Source code	github.com/nomasters/killcord
Released	2018
Last updated	2021-07-16

KillCord is a dApp designed to meet the requirements of a DPS. It assists a subject to publish an encrypted secret to IPFS[23], and sets up a smart contract they can use to track their well-being. In the case that they fail to check-in to the smart contract, the system then releases the decryption key for the subject's secret.

KillCord comes with a disclaimer:

> "WARNING This software is in early alpha. Please do not rely on this with your life. Though great care has been taken to ensure that this code is as well structured and straight-forward as possible, it has not undergone proper peer-review and could have both minor and major bugs that undermine the integrity of the system."

6.2.2.1 Mechanism

KillCord has a number of components:

- **An Ethereum smart contract** to hold the application state and track the subject's aliveness.
- **A client CLI**, written in Go, meant to run on a personally controlled system that sets up the DPS, and enacts check-ins.
- **A hidden publisher**, written in Go, that communicates with the Ethereum smart contract and publishes the key for the encrypted payload in the event that the subject stops checking in.
- **IPFS** is used to store and share the subject's encrypted secret.

In its default configuration, KillCord uses Ethereum to host the smart contract, and IPFS to store the encrypted secret. Both are distributed networks and offer high availability.

Illustrated in figure 12, the subject uses the client CLI on their personal device regularly to check-in to the smart contract. If they fail to do so, the smart contract indicates this, and a hidden publisher application publishes the key to the encrypted payload.

6.2.2.2 Evaluation

Table 34: Requirements vs KillCord (dApp 1)

Requirement	Comments
Confidentiality	KillCord's smart contract stores the secret as an encrypted file on IPFS. As the smart contract cannot protect the decryption key, KillCord relies on a hidden publisher - which must be able to decrypt the subject's secret to release it. The confidentiality of the publisher is linked to the confidentiality of the user's secret, and relies on the security properties of the system it runs on.
Awareness	The KillCord smart contract tracks the subject's aliveness: The user's use of Ethereum is considered a proxy for aliveness. and it will only accept evidence of wellbeing as a call to the smart contract made with the identity of the subject's cryptocurrency account.
Timing	The KillCord smart contract is guaranteed to run by the Ethereum network, however if the publisher can be located and attacked, it may be possible for an attacker to prevent release of the secret at the appropriate time.

[23] InterPlanetary File System (IPFS) is discussed in section 7.5.4.

Requirement	Comments
Resilience	The smart contract is resilient against attempts to compromise its integrity and availability through the properties of the cryptocurrency network. The user's cryptocurrency account is used as the identity to submit well-being check-ins to the smart contract. If located, the publisher (carrying the decryption key for the secret) could be attacked to deny service, alter the integrity of the secret during publishing, or obtain the decryption key and release the secret early. IPFS is used for the release of the encrypted secret, and its decryption key, offering high availability and integrity.
Affordability	Use of KillCord requires a small amount of *gas* (Ethereum currency to be spent on distributed computing) for the smart contract, aliveness submissions, and final publishing. This has points of comparison to a subscription model.
Durability	Once a DPS is established, provided the small amounts of gas required are paid for, the network itself takes care of maintenance. Durability depends on continued support for EVM smart contracts, the Ethereum blockchain, and IPFS.
Explainability	KillCord is fully described in articles and papers, and its source code repository. Although it relies on a number of complex components, including a smart contract, IPFS, and a hidden publisher - it should be possible to explain in simple terms. However, it is also anticipated that most investigative journalists have not developed sufficient expertise to create and configure a DPS using the command-line tools provided. This, combined with the requirement to learn about several complex aspects of the system before trusting it, may make it difficult to trust without advice from a trusted person.
Visibility	The encrypted secret on IPFS may serve as a visible indicator that a KillCord switch is in use, provided it were advertised by the subject to indicate they had created it.

Use of Ethereum and IPFS provide the integrity and availability properties required but, as mentioned in section 7.4.2.2, smart contracts cannot hold secrets. KillCord's solution is to operate a separate, hidden, publisher - which stores the decryption key, checks the state of the smart contract to determine aliveness, and does the work to decrypt and publish the secret if the user is no longer available. In turn, this publisher becomes responsible for the confidentiality of the subject's secret.

The gas[24] required to operate the smart contract is expected to be within the reach of journalists or the organisations they work for, but may depend on the computational capability of the network. ie. If the cost of Ethereum varies, so will the cost of operating the DPS.

6.2.3 dApp 2: Kimono

Table 35: dApp 2 summary: Kimono

	Kimono
URL	kimono.network (unavailable)
Source code	github.com/hillstreetlabs/kimono
Released	2018
Last updated	2018-05-29

Kimono is a time-lock system, using a countdown timer to determine when it is appropriate to release the user's secret.

NB. Kimono no longer seems to be supported. (The last change to the github repository was recorded in 2018, and the project's site, kimono.network, is no longer available.)

6.2.3.1 Mechanism Kimono, described by Paul Fletcher-Hill in 2018 [33], allows a user to distribute fragments of a secret key amongst a group of trusted participants using Shamir's Secret Sharing Scheme (SSSS), a K of N threshold solution (described in section 7.1.1). The fragments are distributed by IPFS (described in section 7.5.4).

It operates a countdown timer as an Ethereum smart contract. When this contract's timelock completes, trusted participants must publish their fragment.

[24] Gas refers to the cost of performing a transaction on the Ethereum network.

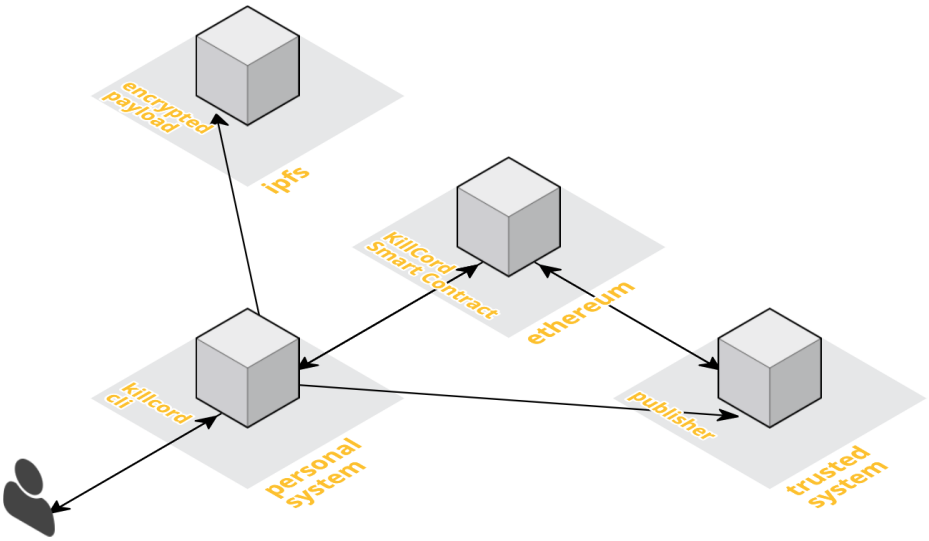


Figure 12: KillCord relational diagram, reproduced from github.com/nomasters/killcord

Kimono provides a motive for participants to behave appropriately, issuing a reward for publication at the appropriate time. The K of N strategy offers some resilience against participant failure.

6.2.3.2 Evaluation

Table 36: Requirements vs Kimono (dApp 2)

Requirement	Comments
Confidentiality	Kimono first breaks the user's secret into SSSS fragments, across a network of N peers to share with the public only once the time-lock is complete. Each fragment is encrypted for its recipient and distributed by IPFS. No individual can reveal the secret, and K or more of the participants would need to collude to release the secret early.
Awareness	N/A Kimono is a time-lock solution. It would need to be adapted to serve as a DPS.
Timing	The smart contract behaves reliably with high availability - it is a good mechanism for determining when to release the secret. (As mentioned, Kimono is a time-lock solution, though.) The participants are incentivised to release their fragment on time, and will be rewarded by the smart contract for doing so.
Resilience	At the heart of the solution is a K of N network of peers. Resilience of the system relies on the appropriate behaviour of at least K participants. This protects the confidentiality, integrity and availability of the system. Secret fragments are distributed by IPFS - considered highly available and with controls for integrity.
Affordability	There is a balance between affordability and timing/resilience of the system: Assuming the only incentive for participants is financial reward, the subject must select a high enough reward to make their network resilient against bribes.
Durability	Durability depends on continued support for EVM smart contracts, the Ethereum blockchain, and IPFS.
Explainability	Kimono is described in a formal paper, but as its site is no longer available simple explainers are not available. It ought to be possible to describe but as with KillCord, it relies on a number of complex components and this, further complicated by the need to trust a group of participants, may require a subject to spend time and effort understanding it before they could trust it.

Requirement	Comments
Visibility	The encrypted secret on IPFS may serve as a visible indicator that a Kimono time-lock is in use, provided it were advertised by the subject to indicate they had created it. The distribution of SSSS fragments to a number of participants may also serve to indicate that Kimono is in use, provided a party were sufficiently interested to discover this on IPFS.

Kimono approaches the problem of decryption and release of the subject's secret by distributing fragments of the decryption key amongst a trust network of N participants. SSSS ensures that K participants are required to reconstruct the secret.

By providing a financial incentive for participants to behave appropriately, Kimono balances risk of inappropriate behaviour amongst the network against cost. It is not immediately obvious how much it would cost to persuade K people from a trust network to reveal their .. Although some may take their task seriously, others may not in the face of a sufficient bribe[25].

Although Kimono is designed to behave as a time capsule, a scenario where the timelock itself were exchanged for an aliveness check would meet the requirements for a DPS. This is a small modification.

6.2.4 Summary

Both solutions rely on different solutions to manage and release of the subject's secret:

Table 37: dApp approaches to manage the subject secret

Solution	Approach
KillCord	Relies on a hidden, trusted publisher application to store and release the decryption key.
Kimono	Distributes SSSS fragments of the decryption key amongst a trusted group of participants, who are incentivised to release at the right time.

Both exploit the non-repudiable and highly available nature of smart contracts to ensure that the aliveness signal is reliable and visible to the secret extractor (either the publisher application, or the participant group). As discussed in section 7.1, choice of participants and selection of values for K and N will provide assurances to users of Kimono that the system is going to behave as required.

Smart contracts rely on cryptocurrency nodes to execute their code. This impacts the durability properties of each DPS. Should the underlying network be abandoned (through loss of confidence in the cryptocurrency, for example), or the virtual machine that they run on be deprecated, the switch itself could fail to operate.

Cryptocurrencies that depend on proof-of-work mining activity consume a vast amount of energy. Unchecked, this level of consumption will continue to contribute to devastating climate change [34]:

> "It is unequivocal that human influence has warmed the atmosphere, ocean and land."
>
> "Global surface temperature will continue to increase until at least the mid-century under all emissions scenarios considered."

This will inevitably affect the durability of any solution that ought to be available over the course of a human lifetime (and may also shorten the expected lifetime of the subject).

Solutions built to depend on wasteful cryptocurrencies carry the risk that their supporting infrastructure will be abandoned in favour of less computationally intensive and environmentally damaging trading systems.

The impact of this, and planned improvements to the Ethereum network are discussed in section 7.4.3.1.

The solutions above are summarised in Appendix C table 02, evaluated in table 04, and scored in table 05. These scores are reproduced in section 6.4.

[25] "More than 70% of people would reveal their computer password in exchange for a bar of chocolate, a survey has found." BBC News, 2004: http://news.bbc.co.uk/1/hi/technology/3639679.stm

6.3 Open source solutions

A number of open source implementations exist, each attempting to meet some of the needs of users of a DPS. The following is a subset of those solutions, selected for relevance to this thesis. Each has a unique approach to the problem.

Table 38: Open source DPS projects evaluated

#	Repository	Description
1	skickar/DeadManSwitch	A python script that can be used to encrypt a user secret (and delete the original) if a keyphrase is not tweeted from a given account in a certain amount of time. (Possible use could be to conceal damning evidence.) It could also be adapted to decrypt a user's file if the keyphrase is not tweeted - so releasing a secret.
2	h313/dead-mans-switch	A python web service that accepts a POST of a predefined password. If this has not been received after a day, it emails a predefined message to a given email address.
3	deadmenswitch/dms	A web server wraps access to a smart contract (DMSContract). This stores a map of switch configurations.
4	EsmailELBoBDev2/Dead-man-s-switch	A script that accepts a password. It is checked against the OS keyring service, and if correct, the date for release of the user's secret is pushed back by a day. If the user misses a day, the secret is sent by SMTP to a predetermined recipient.
5	dmp1ce/DMSS	An application to assist in the SSSS fragmentation of a symmetric key used for encryption / decryption of a secret, the distribution of those fragments to trustees (participants), and tracking the user's aliveness - notifying the trustees if the user is not responsive.

6.3.1 Properties of open source solutions

Generally speaking, code-only projects require the user to operate and host a solution. (In the case of those that employ smart contracts, they require the user to fund and place the contract on a cryptocurrency's blockchain.)

This is often a limiting factor when considering mitigations that require substantial resources. (For instance, those that require software to be hosted in multiple jurisdictions, or in locations with physical protection, quickly become prohibitively resource intensive for individuals unless they can rely on an existing network such as a cryptocurrency blockchain.) It also places full responsibility for operational security on the user - including the encryption and safe deletion of the original secret's plaintext.

6.3.2 Evaluations

Appendix C presents evaluations of all open source projects listed against the requirements for a DPS.

Several of these projects are short scripts, designed to fulfil a partial need for an individual. Others are hackathon projects, or projects created by small teams, for systems that could potentially meet a few more of the requirements for a DPS.

Open source solution 3, deadmenswitch/dms, stands out: It is an example of a simple smart contract, able to store and release a secret when queried. It is passive, requiring the **subject** to check-in and any **recipients** to check for a message. This solves a common issue with smart contracts: they cost money to execute, and if they are caused to poll regularly to run an aliveness check, they could create a significant bill over time.

This solution is incomplete, however. Smart contracts run in the EVM, which cannot hold secrets. This means that the time of release for the content of a DPS built with this system cannot be controlled, and such a solution cannot meet the *confidentiality* requirement as any determined attacker is able to extract the subject's secret.

The open source projects each had unique approaches to the problem - and each has their own strengths and weaknesses. These solutions are summarised in Appendix C table 03, evaluated in table 04, and scored in table 05. These scores are reproduced in section 6.4.

The evaluations and scoring show that none of the open source solutions are suitable for the investigative journalist scenario.

6.4 Scoring

All DPS systems described in sections 6.1 (hosted consumer applications), 6.2 (dApps), and 6.3 (open source solutions), are evaluated and scored by the researcher against the requirements for a DPS[26] in Appendix C, table 05 according to the following criteria:

Table 39: Scoring criteria for evaluated DPS solutions

Score	Description
0	Does not meet requirement, or no information available.
1	Shows awareness of the requirement, minimal steps towards a solution, with significant room for improvement.
2	Partially meets the requirement, with some room for improvement.
3	Meets the requirement comprehensively, with little or no room for improvement.

This scoring matrix is reproduced as figure 13.

Type	Solution	Confidentiality	Awareness	Timing	Resilience	Affordability	Durability	Explainability	Visibility
Hosted	DeadMansSwitch	1	2	2	1	3	1	1	0
Hosted	DeadMan	1	2	3	1	3	0	1	0
Hosted	DeadManTracker	2	2	3	2	3	3	2	0
Hosted	Letters Cloud	2	2	2	2	3	0	1	0
dApp	KillCord	2	3	3	2	3	3	2	2
dApp	Kimono	3	0	3	2	3	3	2	2
dApp	SilentDelivery	3	0	3	3	3	3	2	2
OSS	skickar/DeadManSwitch	1	1	2	1	3	1	2	0
OSS	h313/dead-mans-switch	0	1	2	1	3	1	2	0
dApp	deadmenswitch/dms	1	3	2	1	3	3	2	1
OSS	EsmailELBoBDev2/Dead-man-s-switch	0	1	2	0	3	1	2	0
OSS	dmp1ce/DMSS	2	0	2	2	3	2	2	0

Figure 13: Scoring matrix for existing DPS solutions

Some classes of solution show strengths in specific areas:

- All solutions showed strong *affordability* - none suggesting that they would be prohibitively expensive for investigative journalists or the organisations they work for, regardless of the charging model.
- The dApp solutions are expected to show strong scores for *resilience* and *durability*, relating to the guarantees of availability and integrity made by blockchain networks, and financial incentives that exist for miners and nodes to ensure that the network endures.

[26] Requirements are defined in section 5.2.

- However, smart contracts struggle to provide guarantees of *confidentiality* and must rely on a third party to contain the actual secret (or decryption key for the secret), so creating an additional attack surface and limiting the *resilience* score attainable.
- Very few solutions scored well on the *visibility* requirement. Those that distribute their secret by IPFS, or operate a smart contract in the public domain, were considered to be able to demonstrate activity.
- *Explainability* is difficult to measure, but in particular the hosted solutions struggled to score highly here as none were willing to share information about how they work in sufficient detail to develop trust for the solution.

As described in the evaluations, no individual solution meets all requirements for a DPS to a sufficient level for the investigative journalism use case.

7 Implementation - Stage 2: Review of components

The previous section presented existing systems. This section presents a number of components relevant to the design and implementation of new DPS systems:

Table 40: Supporting research for DPS capabilities

Section	Capability
7.1	Trust networks, and subjective logic
7.2	Aliveness checks, and proxies for aliveness
7.3	Confidential computation
7.4	Guaranteed execution
7.5	Publishing mediums

7.1 Trust networks

Various implementations of DPS rely on networks of people to assess aliveness of an individual, or to hold and release the user's secret. This section discusses a number of techniques that help to manage participant groups, and to manage their risks.

7.1.1 Shamir's Secret Sharing Scheme

As mentioned in "15 Men on a Dead Man's Switch" in section 3.8, Shamir's Secret Sharing Scheme (SSSS) is a commonly used K of N threshold scheme. A secret can be divided into N fragments, of which only K are required to reconstruct it.

A number of implementations of SSSS exist, including a unix tool which simplifies the process of dividing a secret into fragments. Jameson also suggests using a passphrase, and as the unix tool supports application of SSSS to secrets of up to 128 ASCII characters, this should suffice for a strong passphrase.

In the example below, SSSS is being used to split a secret (`X marks the spot`) into 5 fragments, which can then be reconstituted using only 3 of them:

```
$ ssss-split -t 3 -n 5
Generating shares using a (3,5) scheme with dynamic security level.
Enter the secret, at most 128 ASCII characters: X marks the spot
Using a 128 bit security level.
1-a07cfafadfb8c949e10043063dd77bf9
2-c8aca4294241312eae985d2480ffa9f7
3-b94432ce93edd7d2d360939fc0f4b641
4-08426b9a2a8d5f0cde7c1b33734c35bc
5-79aafd7dfb21b9f0a384d58833472a18
```

```
$ ssss-combine -t 3
Enter 3 shares separated by newlines:
Share [1/3]: 3-b94432ce93edd7d2d360939fc0f4b641
Share [2/3]: 5-79aafd7dfb21b9f0a384d58833472a18
Share [3/3]: 1-a07cfafadfb8c949e10043063dd77bf9
Resulting secret: X marks the spot
```

7.1.1.1 Explanation SSSS relies on the notion that K points can define a polynomial of degree $K-1$.

Table 41: Polynomial degrees

Points	Polynomial	Degree
2	Line	1
3	Parabola	2
4	Cubic curve	3

Defining a curve as the secret, a subject can select N points from it, and give those to N trusted contacts. A minimum of K points are required to reconstruct the original curve. The degree of the polynomial defines K.

7.1.2 Choosing participants

A number of factors will help to decide which people or services to include in a K of N threshold solution. It is helpful to understand how the addition of a person will affect the overall behaviour of a group of participants.

- Is a given participant capable and willing to perform the actions requested of them?
- How likely is it that a given participant will engage in inappropriate behaviour in a given time period?
- How vulnerable is a given participant to threats that could alter that behaviour?

K of N systems are resilient, but still vulnerable to possible attacks:

Table 42: Failure states for a K of N system

Attack	Description
Denial of service	Participants are caused to indicate that the user is alive and well when they are not (or participants are prevented from participating).
Early release	Participants are caused to agree (under duress or not) that the user is no longer alive and well, causing an early release of the secret.

These factors may need to be regularly reevaluated, and confidence in the system recalculated as time passes. Some systems described earlier have incentives built into them - rewarding participants for appropriate behaviour. When managing a group, a person will often do this instinctively - selecting participants with good incentives (such as friends and family), or arranging payment for satisfactory behaviour.

7.1.2.1 Naive probability A simple probability can be calculated to represent the likelihood of K participants in a network of N people behaving appropriately, using the formula:

$\sum_{i=K}^{i=N}(\binom{N}{i} \cdot P(success)^i \cdot P(1-success)^{N-i})$

Where:

- N = total number of participants in the group
- K = number of participants required to behave appropriately for the group to behave correctly
- $P(success)$ = probability of a member of the group behaving appropriately when required

This approach is considered naive for a number of reasons:

- It attempts to estimate the likelihood of human behaviour with simple probability.
  - There may be little or no data to help inform the initial probability values[27].
  - Each person should really be understood and evaluated individually (ie. assess their intent and resilience to attack separately).
- It may be necessary to enrol the services of people unknown to the subject, people who are recommended, or agencies whose reputation and resilience are not yet evaluated.
- It doesn't incorporate other attackers into the model (other than through the probability of good behaviour per person).
- It assumes that each participant will behave independently. However, if a person chooses to behave inappropriately, they are also likely to attempt to change the behaviour of enough people to subvert the system (eg. $N-K$ people)[28].

[27] Psychological studies could assist here, although it may still prove difficult to make predictions about individuals without a lot of data, and a good understanding of their attitudes. Many variables can quickly change a person's attitude, eg. factors influencing motivation and loyalty of participants (including a common cause, common enemy, family ties, and friendships), incentives built into the DPS, changing personal circumstances (including financial pressures, threats, bribery and persuasion).

[28] Interactions between people in the network significantly complicate the matter, and are considered out of scope for this thesis.

- It assumes that each period is independent - whereas if a person expresses inappropriate behaviour in one period, it may also influence their behaviour in future periods. (Or perhaps they will be removed from the network.)

7.1.2.2 Subjective logic The naive probability model has room for improvement. One solution is to represent participants in a trust network using subjective logic.

As described in 2006 paper *Trust Network Analysis with Subjective Logic* [1], Jøsang, Hayward, and Pope show how subjective logic can be used to represent trust relationships, and networks of trust relationships.

These relationships can be classified as:

- **Direct functional trust** in a proposition (ie. A's belief that participant B will behave appropriately)
- **Indirect functional trust** (ie. B informs A how likely they believe it is that C will behave appropriately)
- **Direct referral trust** (ie. A's trust in B's advice about C)

These are often described with a simple example:

> Alice (A) needs a mechanic, and asks Bob (B) for his opinion about Eric (C). Bob holds a direct functional trust opinion about Eric, and Alice has direct referral trust opinion about Bob. Alice uses both of these opinions to form her own indirect functional trust opinion about Eric.

This is illustrated in Figure 14.

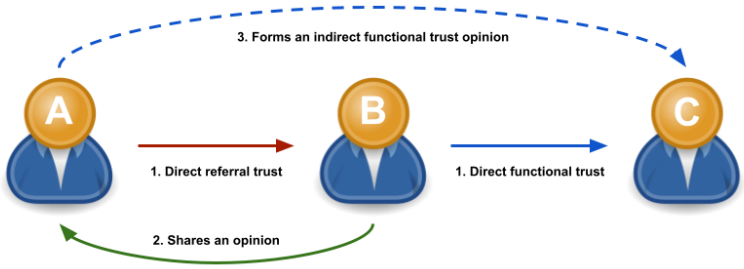


Figure 14: Alice forms an indirect functional trust opinion of Eric

Further exploration of Jøsang's 2016 book, *Subjective Logic* [2], shows how trust opinions and subjective logic can be used to represent a network, and how it can be applied to reasoning about system reliability. Further detail of this is provided in Appendix D.

The application of subjective logic is an interesting path to follow, but there are good reasons to treat it as a limited, theoretical solution:

- Everything discussed assumes that each person in the participant network behaves independently.
- It offers no practical way to measure, determine, or estimate values for the trust opinions required to reason about a network of people.

When building participant networks for a DPS, the user in the scoped scenario for this thesis is primarily a journalist, and unlikely to apply complex mathematics to the problem. Rather, they are considered likely to select people they know and trust to a high degree, or select an alternative system that does not rely on people.

Regardless of these weaknesses, subjective logic is recognised as *one way* to reason about trust in a network of people. It may not yet offer results that can compete with a subject's intuition, but it is an active area of research.

If K of N threshold solutions must rely on unknown persons, it may help to develop processes that can identify weak members of a participant network, and to calculate how a given person's trustworthiness can affect the overall safety of the solution.

7.2 Aliveness checks

Systems that can determine whether a person is alive or dead must also contend with attackers who wish to provide evidence disputing the truth.

The failure states for a DPS system, laid out in section 5.1.2, are:

- early release
- denial of service

Attempting to confuse the system by persuading it that the user is alive when they are not, or dead when alive and well can lead a system to exhibit these failure states.

A variety of signals can serve as a proxy for the user's well-being.

7.2.1 Classic authentication

Authentication with a secret (ie. a password), is a common solution to determining if the user is who they should be.

In some DPS systems, ability to authenticate is considered a proxy for being alive and well. This means that if a person fails to authenticate within a given time frame, they can be considered unavailable and their secret should be released.

Authentication is a security task, and often interferes with what people are trying to do. In *The Compliance Budget: Managing Security Behaviour in Organisations* [35], Beautement, Sasse and Wonham describe the compliance budget as an amount of effort that people are prepared to spend on security tasks.

Provided the effort required to demonstrate that the user is alive is sufficiently small, they will continue to do it. However, if this becomes difficult (eg. a slow, inaccessible, or unresponsive interface) or too frequent, the user may decide it is not worth the effort. In the case of investigative journalism where the user has other things to worry about and may not be able to suffer regular interruptions for complex passwords, they may choose to use a different product.

Systems that rely on user secrets also need to consider the resilience of the user themselves. A sufficiently robust secret-based authentication process may accidentally incentivise an attack on the user to learn their password and disarm the system. Figure 15, a comic strip reproduced from xkcd.com, illustrates this point neatly.

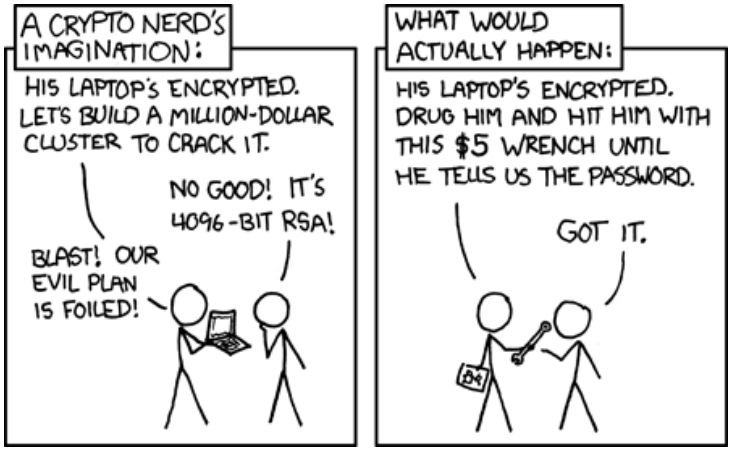


Figure 15: xkcd: security [36]

7.2.2 Biometrics

Biometric technologies are employed by a wide range of consumer and enterprise devices to authenticate the user. An arms-race is underway between crackers (wishing to create false positive inputs with increasingly sophisticated deceptions) and manufacturers (wishing to prevent this with increasingly sophisticated detections, able to observe identities and signs of life).

For the purposes of this thesis, it is important to note that:

- A biometric signal alone may not be reliable evidence of aliveness. Some sensors are more vulnerable than others[29].
- Implementations that use biometrics as a proxy for aliveness should understand how these signals could be falsified.
- A remote device that captures such a signal ought to have a means to attest that it has not been tampered.

Tamper resistance and attestation are features of trusted computing, discussed in section 7.3.1.

7.2.3 Human judgement

Humans are *thought* to be reasonably good at judging whether they have recently had an authentic interaction with someone who is alive and well. However, technology is making rapid progress with techniques that can successfully deceive them.

Originally called the Imitation Game, and proposed by Alan Turing in 1950, the Turing Test is a competition style event to determine whether conversational chatbots can convince a panel of judges that they are human. To pass, the software must deceive more than 30% of the judges.

A common critique of the Turing Test, that it "relies solely on the ability to fool people" and so is a "sorely inadequate test of intelligence" [37], increases its relevance to this thesis.

In *Hidden interlocutor misidentification in practical Turing tests* [38], 2010, Shah and Warwick note that, at the 18th Loebner prize event[30] in October 2008, a team of invited judges from the University of Oxford correctly distinguished humans from bots with a success rate of only 56%.

The quality of chatbots has improved since 2008 and is expected to continue to improve. Similarly, the ability for bots to simulate the appearance of human voice and video has been steadily improving.

In 2017, a startup called Lyrebird (now owned by Descript[31]) developed and demonstrated [39], a technology able to mimic human voices and give them new speech. Although the demo clips are limited, this technology is expected to have improved since first debut.

In 2018 Google Duplex [40], a feature designed for their assistant product, was demonstrated placing a call to a salon to book an appointment. The recipient of the call did not seem to detect that they were talking with software. Google Duplex, as described by the team, operates in a tightly defined domain - and cannot carry out general conversations, yet. It would not be expected pass the Turing Test without strong constraints on topic.

It is possible to deceive humans with falsified video, at least for a short period of time[32] as seen with projects such as DeepFake or Avatarify [42]. The quality of these deceptions is expected to improve over time.

Humans are also susceptible to a number of possible attacks (often referred to as social engineering):

- Deception
- Persuasion
- Intimidation
- Blackmail

When relying on people to decide if someone is alive and well, there are a variety of strategies available to help mitigate the risks that they may not behave as required. For discussion of this, see section 7.1.

7.2.4 Paralysis proofs

Paralysis proofs are designed to manage the risk of group collusion. In a scenario where a group of people share control of a cryptocurrency fund with a K of N threshold system, it is possible for members of the group to collude, declare that one of them is no longer available, and misappropriate the funds.

[29] For instance, *capacitive* fingerprint readers are better equipped to detect the capacitive properties of live human fingers, whereas *ultrasonic* or *optical* fingerprint readers may be vulnerable to prosthetics that capture and reproduce the ridges on a finger.

[30] The Loebner Prize for Articificial Intelligence is a Turing Test event where bots are submitted and tested in a competitive environment.

[31] Unfortunately, Lyrebird's original demonstration is not available through the Descript website.

[32] Short video *Fake Elon Musk joined the Zoom call* [41], from 2020, is a perfect example of this.

Paralysis proofs rely on properties of cryptocurrency blockchain, and smart contracts, to offer a solution that allows N parties to govern the spend of an amount of cryptocurrency.

In *Paralysis Proofs: Secure Dynamic Access Structures for Cryptocurrency Custody and More* [3], Zhang, Daian and Bentov describe handing a fund to an account controlled by a smart contract or SGX enclave.

If one party, the *missing party*, is believed dead, the *remaining parties* issue a challenge. The *missing party* may then provide evidence that they are still alive, by spending a small amount (known as the *life signal*), which in turn prevents action being taken despite a consensus from the *remaining parties*. This is illustrated in figure 16.

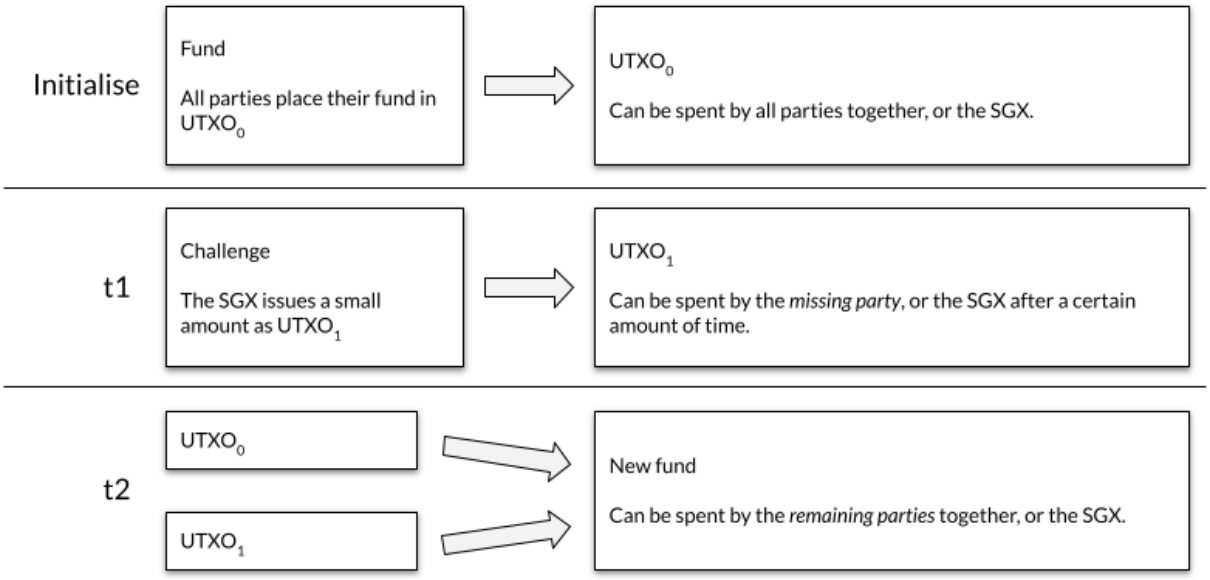


Figure 16: Paralysis proof: $t2$ relies on $UTXO_0$ & $UTXO_1$

For a detailed description of Paralysis Proofs, see Appendix E.

There are some risks associated with this:

Remaining parties could issue any number of challenges unless properly constrained by the SGX, and so could overwhelm the *missing party* (or issue a challenge at a time when they know the *missing party* will be unavailable). If they cannot respond to every challenge (assuming that the system does not have mitigations in place for such a high frequency of challenges).

Paralysis proofs deal with the management of funds, however Dead Person Switches deal with secrets. Cryptocurrency blockchains operate in full public view, and are not suitable for storing secrets.

7.2.5 Inactive account managers

Inactivity of a digital account, or a set of accounts, might also qualify as evidence of a person's death.

An example system that uses this is Google's Inactive Account Manager (IAM) [12]. It allows users to choose who is granted access to their data after a number of months inactivity (or if the account and its data should be deleted).

It's intended to help a person plan to bequeath control of their digital assets after their death (or loss of capacity). It has the advantage of being able to rely on Google's authentication facilities - expected to steadily improve over time, as Google are highly incentivised to remain at the forefront of identity assurance, in order to properly serve their ecosystem of tools.

The IAM is an example where the motivation of the operator is reasonably clear. Google rely on their reputation for business. They have many products that rely on secure storage, and an authentication process that's resistant to attack. It is reasonable to suppose that the IAM also benefits from the maintenance of these shared systems. Incentives pose an important aspect of systems that rely on trusted third parties.

Another example of a succession plan is GitHub's Deceased User Policy [18], and documentation describing how a user may nominate a successor [13].

> An appointed successor can manage your public repositories after presenting a death certificate then waiting for 7 days or presenting an obituary then waiting for 21 days.

Under this scheme, GitHub require that the successor present documentation to the effect that a person has passed away and then wait for a predefined period. During this time, the account owner themselves have an opportunity to present evidence that they are still alive - but the core of this policy relies on a user's trust in their chosen successor.

7.2.6 Evidence

As illustrated by the GitHub Deceased User Policy above, documentation (such as a death certificate, or an obituary) may provide some level of assurance that a person has passed away.

Assurance of the legitimacy of documents is a difficult proposition, particularly where documents are physical and cannot be digitally signed.

Some documents have security features designed into their physical attributes. Good examples of this are the security features built into UK passports - used to ensure that the document is genuine, and has not been tampered. Passports also have the advantage of carrying an NFC chip able to present a digitally signed document that contains the same attributes as the printed passport. Validation of these security features is described in the UK Government's Good Practice Guide 45, *How to prove and verify someone's identity* [19].

In the UK, physical death certificates do not have strong security features. However, they are (increasingly) supported by a programme of digitisation underway at the General Registry Office, providing an authoritative source of live event records. The *Life Event Verification Service* [43] provides data about recorded births, marriages and deaths.

The ability to validate a document could help where an attacker is attempting to cause an *early release* by presenting a false death certificate.

NB. While the General Registry Office provide information about registered deaths [44], and a number of databases are made available to commercial companies, the LEV service currently appears to be only available to government departments.

If GitHub have a facility to validate death certificates, or seek assurance from the user themselves that they are still alive, this fail state can be mitigated.

7.2.7 Data protection regulations

Under European and UK law, *General Data Protection Regulations* [20] (GDPR) dictate a number of requirements to protect a user's right to control their own data.

GDPR defines person data in relation to a "natural person" [45] - indicating that its scope does not cover data relating to "legal persons" such as companies. An important caveat of this is that the person must be alive. Protections for personal data do not extend to the deceased.

GDPR does not specify how a service should assure themselves that a person is alive, or define levels of assurance for this.

GDPR's disinterest in data after death presents a decision for the subject. They must select organisations they trust to control and process their data. Clearly defined policies go a long way to providing assurance of the intent of these organisations, even if they may be difficult to enforce after death.

7.3 Confidential computation

This thesis defines a requirement for confidentiality:

> A DPS must keep a user's secret confidential until it determines it is appropriate to release the secret (confidentiality).

There are a number of means by which software can keep secrets - through encryption or the application of security controls to prevent intruders, eavesdroppers, and leaks.

7.3.1 Trusted Computing

In *The Ten Page Introduction to Trusted Computing* [4], 2008, Andrew Martin describes the concept of Trusted Computing, and some of the reasoning behind architectural decisions it incorporates.

Technologies that implement trusted computing offer a way to detect alterations to the operating system, or the presence of malicious code that could produce unwanted behaviour. In turn this makes it possible to mitigate the risk of attacks that can modify the code of the DPS - causing it to alter (or destroy) the content of the secret, leak the secret, or fail to release the secret when required to do so.

7.3.2 Trusted Platform Module (TPM)

Computing systems are built in layers, from the hardware, CPU, through BIOS, operating system, and applications. This is illustrated in figure 17, reproduced from Trusted Computing Infrastructure [46], delivered by David Grawrock in 2019.

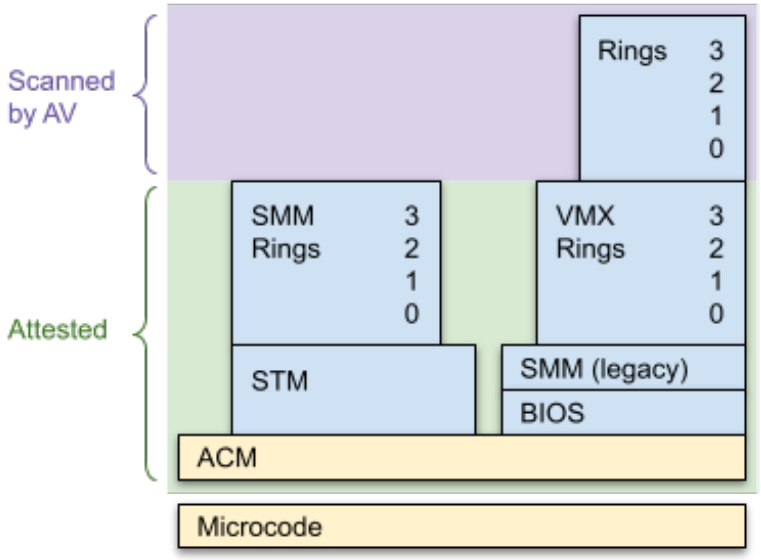


Figure 17: Hardware and operating system layers in an Intel CPU, reproduced from Grawrock [46]

Underpinning much of Intel's trusted computing architecture, and now a required component for Windows 11, is a product called the Trusted Platform Module (TPM). The TPM is responsible for measuring and attesting to each step of a system's boot sequence - so providing guarantees of the integrity of the hardware's microcode, BIOS, and operating system.

For more detail on the TPM, see Appendix F.

7.3.3 Trusted Execution Environment (TEE)

Trusted Execution Environments offer opportunities to execute code in an environment that is protected from other software on the system. Code running in a TEE is expected to behave correctly and to retain its secrets, even if the operating system is compromised. This immediately lends itself to the requirements of a DPS, as it offers guarantees around the integrity of the code, and the confidentiality of the secret.

7.3.3.1 Intel SGX The design principles behind SGX are intended to provide trusted enclaves, managed by Intel CPUs, that are reliably isolated from all other software running on a system. This is illustrated in figure 18, also reproduced from Grawrock [46].

An SGX-enabled application is divided into an untrusted portion, and a trusted portion comprising 1 or more trusted enclaves, managed by SGX. An SGX enclave contains private data and code - inaccessible, even to the rest of the application, and a well-defined interface for calling the code.

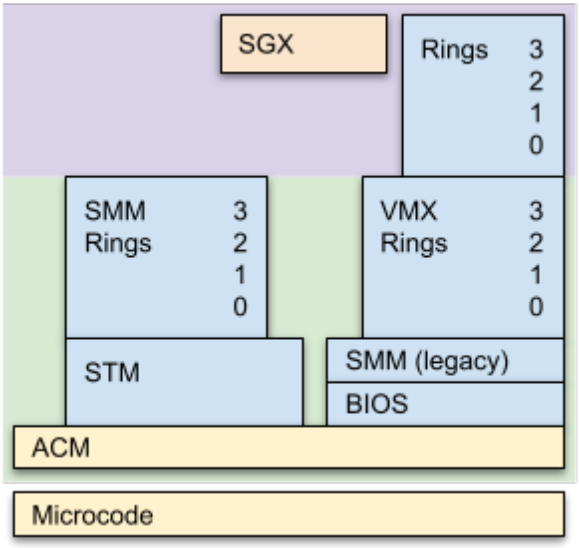


Figure 18: Hardware and operating system layers including SGX, reproduced from Grawrock [46]

For more detail on Intel SGX, see Appendix F.

7.3.4 Homomorphic encryption

Under normal circumstances, data must be decrypted to operate on it, leaving it vulnerable to exploits that could compromise its confidentiality. Fully Homomorphic Encryption (FHE) is a form of encryption that permits operation on encrypted data without first decrypting it. Subsequent decryption of the data then results in the same output as if the operations had been applied to the decrypted information.

This does not immediately lend itself to the use case of a DPS - as the secret is not manipulated. However, it may also be possible to apply FHE to the storage and execution of *code* without decrypting or exposing it. In *How to Run Turing Machines on Encrypted Data* [21], 2013, Goldwasser et al. describe what is required to run a turing machine[33] over encrypted data. (This replaces the need to represent algorithms as circuits for homomorphic encryption techniques.)

FHE is not yet considered ready for general use cases, as it is still extremely inefficient. It is an active area of research. Should efficiencies be found, this technique may provide a provably safe alternative to the use of TEEs (or it may find its way into future implementations of TEE technologies).

7.3.5 Source code protection

There are compelling arguments to suggest that the code of a DPS itself should *not* be confidential (although its integrity remains essential).

There are a number of recognised benefits to open source software including automated vulnerability detection[34], community and researcher scrutiny, and lower cost of testing and improvement.

7.4 Guaranteed execution

Availability is an essential property of a DPS solution. There are a variety of ways to ensure that code will be executed - uninterrupted, and unaltered.

7.4.1 Hardened devices

Hardening devices is a complex topic, and a full exploration of this is outside the scope of this thesis. However, there are a number of existing techniques and technologies which can help mitigate the risk of a

[33] ie. general purpose computation.

[34] These tools are particularly helpful when analysing software libraries imported into projects - and help to mitigate supply-chain attacks found in known versions of libraries with vulnerabilities.

physical attack. If a single, hardened device were considered the most appropriate option to operate a DPS, a mobile phone might be a good candidate:

- They are almost never out of the company of a person who is motivated to keep them safe[35].
- They are in almost constant use for a wide number of purposes.
- They have a lengthy battery life when disconnected.
- They connect to a highly available mobile network.
- Many contain sophisticated security features, such as HSM-like capabilities[36] for storing fingerprint data and private keys, and verified boot processes[37].

However, if the subject of a DPS is attacked, and their phone is operating as their DPS, it is reasonable to assume that the attacker may investigate their phone, and very likely disable it - even if they are not expecting it to contain a DPS. Many mobile phones are quickly disabled after theft to prevent recovery through tracking tools.

It is also reasonable to assume that even phones designed for security are not protected against nation state attack. A large amount of organised crime was interrupted in 2020 when it was revealed that police across Europe had successfully infiltrated EncroChat [48] - an encrypted communications system built with custom devices.

In the *Journalist's Resource for Safe and Ethical Reporting* [23], Reporters San Frontières advise:

> "Rule 2: Be wary of smart phones."

Noting that:

> "A smartphone can be treacherous. It constantly emits large amounts of data to enable it to connect to mobile networks and the Internet, which can easily be used to locate you. If it falls into unknown hands, even for just a few minutes at a checkpoint or customs post, malicious software can be installed which can transform it into a bugging device. This can make it your worst enemy."

7.4.2 Distributed computing

Distributed computing networks perform a variety of problem solving tasks - there are a number of prolific networks, including those for distributed science (SETI@home, folding@home), and vast networks concurrently mining for cryptocurrency, maintaining blockchain ledgers, and executing smart contracts.

There are a number of advantages to adopting a distributed computing mechanism that already exists:

- Networks with an established community are difficult to attack. In most cases, a majority of nodes will need to be compromised before the network becomes unreliable or unavailable. This is considered out of reach of all but the most resourced attackers - and likely to draw a lot of attention to the attack.
- Participants are already motivated to continue operating the distributed network.
- If participants are unaware of the task they're executing, an opportunity arises to conceal the activity of the DPS.

7.4.2.1 Distributed science The distributed computing effort SETI@home was designed to process signal data from radio telescopes in the search for evidence of alien life. It was initiated in 1999, and concluded in 2020. At its peak, it registered as one of the most powerful supercomputers on the planet.

The folding@home project is currently running and claims to have "created the most powerful supercomputer on the planet" [49]. It is primarily used for simulating potential folding configurations for proteins.

Although highly available, some properties of these networks rule them out as an appropriate distributed computing network for a DPS:

[35] Per ONS Property Crime data, [22, year ending March 2020 dataset, tables 15 and 17], in the year ending March 2020, of an estimate 48.6 million mobile phone owners, 325,000 experienced mobile phone theft.

[36] A hardware security module (HSM) is a physical tamper-resistant device that provides confidentiality for sensitive data. For instance, an HSM is able to store private keys and use them to decrypt ciphertexts, or sign messages, without releasing those private keys to the operating system.

[37] eg. Titan-M chips [47], built into Pixel phones (2018); or the Secure Enclave in iOS devices.

- They are designed to operate on open scientific data, and so do not need their operating data to remain confidential. (Integrity is important, though.)
- They are single purpose networks, and cannot be repurposed without altering the agreement with the users who donate computing resources.

7.4.2.2 Smart contracts Cryptocurrencies use blockchains as their ledger of transactions, which provide guarantees about the atomic nature[38] of operations (such as financial transactions), and a permanent, non-repudiable record agreed by consensus. Some cryptocurrencies, such as Ethereum, also allow instances of code, called smart contracts, to be stored on their blockchains and executed by their network. There are a number of properties of smart contracts that Ethereum enforces:

- Like transactions, smart contracts are universally agreed by consensus, and atomic in nature.
- Smart contracts run on Ethereum nodes in an isolated virtual machine called the EVM.
- The cost ('gas') to run a smart contract is related to the resources it consumes when executing.

Smart contracts have a number of properties which are valuable to the implementation of a DPS:

- They are guaranteed to run. (Availability)
- They cannot be altered after being committed to the blockchain. (Integrity)
- Outputs, and state are affirmed by consensus. (Integrity)

However, although the EVM is specified as an isolated virtual machine, this cannot be completely guaranteed:

- Each node may have its own EVM implementation.
- Each EVM has access to the code and state of the smart contract, in order to execute it.

This leads to an undesirable property:

- Smart contracts cannot protect the confidentiality of secrets[39]. (Confidentiality)

Further, it is not impossible to attack Ethereum [24] - and if a network were successfully compromised, this would disrupt the guarantees made above. In a *51% attack*, an attacker attempts to take control of the majority of the computing power of the network. This gives them control over transactions - allowing them to alter the order of, or cancel transactions (and re-spend the currency).

In "A survey of attacks on Ethereum smart contracts" [25], Atzei, Bartoletti and Cimoli describe a number of vulnerabilities in smart contracts that could also be exploited.

As with all technologies, there are risks - and selecting a network that is resilient[40] to host a DPS solution, that relies on smart contracts, will offer the strongest chance of recovery and continued operation in the face of an attack.

7.4.3 Durability

When selecting a platform and technologies for a DPS, continuity is an important consideration. A DPS might be expected to endure the natural lifetime of a person.

The longest running distributed network described above is SETI@home, which ran from 1999 to 2020: 21 years. Others may last as long, or longer. In each case this is difficult to predict.

Many technologies have flourished and then disappeared since the dawn of computing. The question of how to ensure that software will continue to execute in the future is a difficult one.

7.4.3.1 Blockchain smart contracts Cryptocurrencies are seen by some as a fad, others as a new, decentralised way of conducting transactions which will last because of the unregulated benefits they offer.

[38] Atomic transactions are considered to succeed or fail completely, no matter how many steps are involved.

[39] Even secrets encrypted for specific recipients are at risk of early release - as the intended recipients could obtain them at any time by exploiting the EVM.

[40] Some schools of thought might suggest that a network such as Ethereum, which has experienced and recovered from several large scale attacks like this, has developed resilience in response to threats like these.

As mentioned in section 6.2.4, cryptocurrencies that rely on proof-of-work mining activity consume vast amounts of energy [50], posing an existential risk to the habitability of the world [34] over time. This is now tracked at the Cambridge Bitcoin Electricity Consumption Index [51].

In a recent development, Ethereum have announced their intention to switch from mining to a new technique called proof-of-stake [52]. They expect the change to be complete in 2022. This is estimated to reduce power consumption from participants in the network by 99.9%.

This increases the likelihood that smart contracts across the Ethereum network will endure, as it becomes more affordable, more practical, and less environmentally damaging to participate in the network's distributed computing efforts.

7.4.3.2 The Domesday Project In 1986, the BBC published the Domesday Project - a large data collection project containing articles about local geography, social issues, daily life across the UK, maps, photographs, statistical data, virtual walks, OS maps, colour photographs and the 1981 census data.

By 2002, it was reported that the hardware required to view the Domesday project had become obsolete [53] - rendering the information unreadable. This serves to illustrate how difficult it is to build a piece of software that successive generations of new hardware can execute.

A DPS system must attempt to do this in the face of numerous unanticipated changes to networking, human interfaces, software and hardware. It must be patched and maintained to ensure that 0-day vulnerabilities are not exploited as they are discovered.

7.4.3.3 Legacy systems Some extremely old systems, such as banking mainframes, the Police National Computer (introduced 1974), or software written in COBOL (developed 1959), have proved very long lived. However, maintenance or replacement of these systems is an expensive endeavour[41]. Today there are a diminishing number of COBOL developers, and banks are (finally) realising the benefits of migrating away from (and releasing themselves from the risks of) their failing mainframe hardware and software.

7.4.3.4 Emulation Emulation of older hardware offers a possible solution, although in security sensitive applications such as DPS it comes with risk. Unpatched vulnerabilities from the original system will still be present and this must be reflected in risk analyses and mitigations.

7.4.3.5 Institutions A long running institution dedicated to maintaining the software and hardware for a DPS may prove a reliable option. Wikipedia provides a list of oldest institutions in continuous operation [55]. Of course, many of these have survived the test of time without building dependencies on, or developing expertise in, modern technologies. It may not be practical to persuade such an organisation to host a DPS system (many are religious institutions, or specialised companies). However, options exist such as academic institutions, or modern technology companies that are now considered "too big to fail."

7.5 Publishing mediums

It is important that a secret revealed by a DPS is distributed in a reliable way, preserving integrity and ensuring availability - either as a public resource, or as a message sent to specified individuals.

Thought should also be given to resilience of a chosen medium:

- Is it susceptible to a denial of service attack?
- Can information be altered?
- Is it managed by a single operator?
- Is that operator resilient to attack?
- Which legislation governs that operator?

[41] Attempts to replace the Police National Computer were reported in 2020 as £45m over budget, years late, and not expected to show results until 2025 [54].

7.5.1 Public forums

Public forums are a widely available option for publication, and worth considering.

The motives of the operator are important in deciding whether to trust a forum with the publication of a valuable secret:

Some are operated by organisations that prioritise free speech, and may resist attempts to censor the secret. A good example of this is 4chan - a site designed to facilitate anonymous comment.

There are a number of advantages to sharing the secret with a public forum:

- High volume of traffic means the secret will be widely distributed.
- Some may be too poorly curated to detect or remove sensitive content quickly.
- Some sites have developed resistance to censorship - locating their hosting in countries with poorly enforced copyright and libel laws.

For sites such as these, even if the original post is eventually censored, many people will have seen the information, and some of those may copy and re-post it - making it very difficult to censor. This is known as the Streisand Effect [56], and is illustrated by the 2007 AACS Encryption Key controversy - a failed attempt to censor a particular number used in the copyright mechanism for DVD players [57].

There are also risks: Anonymous forums often have a poor signal to noise ratio for truthful information, and some sites have a high tolerance for discriminatory content, libel or abuse. The published secret may not be taken seriously.

7.5.2 Direct communication

The simplest form of direct communication is email. A DPS solution may publish the subject's secret by sending it to pre-determined people.

There are advantages to this:

- The secret can be encrypted for the chosen recipients (eg. with PGP[42]), making it harder to compromise a stolen secret.
- Chosen people can be tasked with acting on the information if they should receive it (perhaps by bringing it to law enforcement to protect the subject, or initiating legal proceedings against the threat).

There are also disadvantages:

- Risk is passed to the chosen recipients, who may also be at risk of attack.
  - A switch will need to store metadata such as the email addresses of recipients. If successfully attacked, this information can be used to identify them.
  - Emails can be intercepted, and this can be used to determine the identities of these people.
- Email addresses change, and people may not be checking old addresses. Regular contact to ensure they are still using the account may also draw the attention of threats.

7.5.3 Trusted channels

Some organisations, such as the Good Law Project[43] or Wikileaks[44] offer methods for the anonymous submission of whistleblowing information.

Choice of organisation is important, as their moves must be understood. Some are operated by journalists and organisations with the capability to investigate, present, and publish the information safely.

Wikileaks has drawn a lot of attention in recent years for its activity publishing information about national intelligence services [58].

[42] OpenPGP is an open standard for identity and encryption: https://www.openpgp.org/, GnuPG (GPG) is one of those implementations: https://gnupg.org/

[43] The Good Law Project tip-offs: http://goodlawproject.org/got-a-tip-off/

[44] Wikileaks submissions: https://wikileaks.org/#submit

7.5.4 Distributed File Systems

Another way to achieve highly available distribution is through a distributed file system (DFS).

The Interplanetary File System[45] (IPFS) is a DFS, and the most commonly cited example amongst the existing systems reviewed for this thesis. Content is distributed amongst participating nodes on the network, so making it difficult to lose or censor.

IPFS is built as a peer-to-peer network with a common protocol. Files are stored as chunks with a cryptographic hash and content identifier (CID), based on the hash of the file. Nodes on IPFS cache content as they retrieve it, and serve it on - so increasing the capacity to deliver more popular files (similar to the way BitTorrent clients operate).

IPFS maintains the integrity of files available on it through cryptographic hashes - and when a file changes, the CID changes too - creating a new version.

A DPS could store its encrypted secret on IPFS, so ensuring that it is accessible and available to any user who wishes to cache and inspect it. Then, should the DPS need to release the key, it can also push this to IPFS (and optionally issue a decrypted version of the payload). All of these files will then be publicly available, and their availability is difficult to disrupt.

[45] Interplanetary File System: https://ipfs.io/

8 Implementation - Stage 3: Propose and evaluate designs

Three new designs are proposed in subsequent sections, each illustrating a different approach to the build and operation of a DPS.

Table 43: Proposed design summaries

#	Section	Summary
1	8.1	A simple design built with micro-services for a cloud hosting provider.
2	8.2	A dApp built with an Ethereum smart contract that relies on the SECRET network to manage the subject's secret.
3	8.3	A passive encryption system that allows a blockchain proxy for proof-of-life to serve as the decryption witness.

These designs are chosen to represent distinct strategies for the implementation of a DPS. They are intended to meet as many of the requirements for a DPS laid out in section 5.2 as possible.

For each design:

- The premise is explained.
- Relevant information security research and components are presented.
- The design is described.
- Diagrams are presented to aide visualisation.

Design 1 serves to illustrate a micro-services architecture using classic system design components. For that design a risk analysis is conducted, highlighting weaknesses in the design. This serves as a point of reference.

As described in the methodology, section 4.5, designs 2 and 3 rely on one or more blockchains, and so the systems they represent have boundaries that encompass large distributed computing networks with complex properties. Rather than developing a risk analysis for them, each design is evaluated against the requirements for a DPS:

- Findings for each requirement are presented.
- Potential mitigations for found risks are offered.
- Further considerations are discussed.

In section **??**, these designs are compared to discuss strengths and weaknesses in the context of all proposed systems.

8.1 Design 1: Hosted micro-service DPS

8.1.1 Premise

This design represents a DPS constructed as a series of micro-services, for a cloud based hosting environment.

The service is designed to be operated by a single operator, on a cloud platform such as Amazon Web Services, Microsoft Azure, or Google Cloud.

It is intended to support multiple subjects, each of whom may construct one or more switches with a secret, and then submit their own aliveness signals to the system.

Careful application of design principles, and division of the services into a number of security zones, help to ensure that the service is resilient against a variety of different attacks.

The National Cyber Security Centre (NCSC) provide a number of secure design principles [59]. Some key principles lifted from the document guide this design:

- **Design for easy maintenance.** The system is designed to be clear to understand, with a number of simple micro-services that each performs a simple task.
- **Use a zoned or segmented network approach.** The system deliberately zones the user-write operations away from the system-read operations - so that a compromised process is not in the same security zone as, or able to directly reach, its opposite. The databases themselves are protected

by processes as small as lambdas, responsible for a single read or write operation. Access to each lambda is also controlled.

- **Reduce attack surface.** The system features a limited number of external inputs, and these are indicated clearly in the design. Users may only perform a small number of operations. Each endpoint in a zone that accepts inputs from a less-trusted zone is protected by a firewall to ensure that inputs come only from expected services, and only to expected endpoints. Additionally, read-only endpoints and write-only endpoints limit the possibility for unexpected operations on data.
- **Transformation, validation, and safe rendering.** The system transforms all inputs from the user before sending them to endpoints in more trusted zones. Inputs are validated before use.

8.1.2 Design

Figure 19 illustrates the architecture of this system.

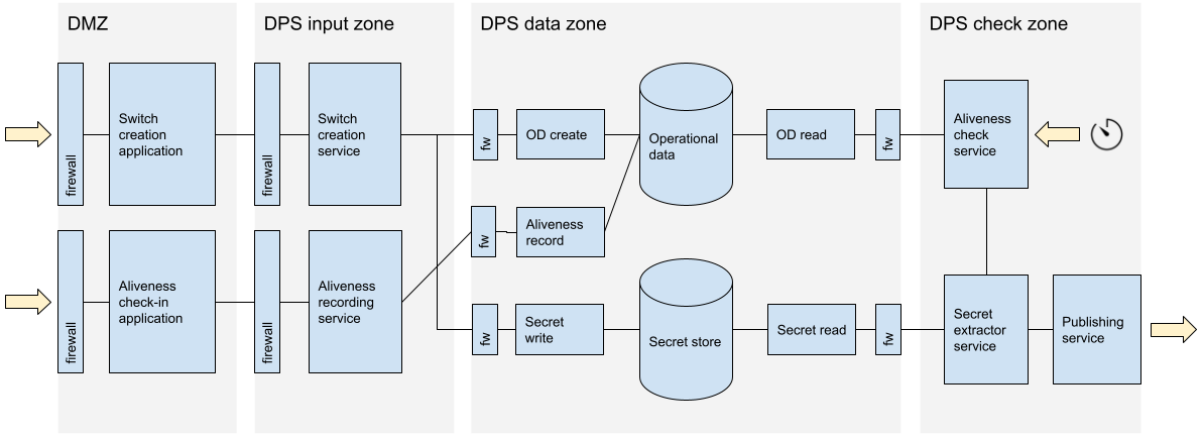


Figure 19: Design proposal: micro-services architecture

The design features a number of micro-services divided into several trust zones:

- **DMZ.** The demilitarised zone. Public-facing web applications reside here, protected by a firewall, and accept user input. They validate and transform this input before passing it into the **DPS input zone**.
  - **Switch creation application.** Accepts information required to construct a new switch, including the secret a user will attach to their aliveness evidence.
  - **Aliveness check-in application.** Accepts an assertion from a user, accompanied by their secret, to indicate they are currently alive.
- **DPS input zone.** Services that represent the logic of user actions (create a switch, or provide aliveness evidence) operate here. They accept and validate input from the DMZ, and make calls to individual lambdas in the **DPS data zone**.
- **DPS data zone.** The data zone contains two data stores, an HSM, and several lambdas control operations on those stores.
  - **Operational data store.** Contains switch definitions (including the public key required to validate a subject's aliveness evidence, and the identity of the subject's secret).
  - **Secret store.** Contains user secrets, encrypted at rest by the HSM.
  - **HSM.** The HSM (not visible, but accessible to each lambda in its zone with its own access rules) contains the keys and the capability to encrypt and decrypt the subject's secret.
  - **Lambdas.** The following operations are defined as lambdas, which can operate on the data store:
    - **OD Create.** Operational data creation for the initialisation of a DPS.
    - **Secret write.** Encryption and storage of the subject's secret during initialisation of a DPS.
    - **Aliveness record.** Storage of a signed piece of evidence indicating that the subject is alive at a given time.
  - **OD read.** The read and verification of operational data when checking for a subject's aliveness.
  - **Secret read.** The read and decryption of the subject's secret for publication.

- **DPS check zone.** Services here are periodically activated by a recurring trigger to check for subject aliveness and if necessary extract and publish a secret.
  - **Aliveness check service.** Periodically checks the operational data to determine if a subject has missed a check-in, and trigger the **secret extractor service** if required.
  - **Secret extractor service.** Retrieves the subject's secret, and passes it to the publishing service.
  - **Publishing service.** Transmits the secret provided to a chosen publishing medium. This could be an email address, or a public and highly available service, such as twitter.

8.1.3 Risk analysis

Threats to a DPS are listed in section 5.4.2.

General informational assets stored by a DPS are listed in section 5.4.1, followed by determination of the impact of loss of security properties for those assets in section 5.4.3.

Additional assets related to this design are described below.

Tables for this risk assessment exercise are available in Appendix G.

8.1.3.1 Additional assets and impact Table 44 lists assets specific to this design, and evaluates the impact of loss of their CIA[46] properties.

Table 44: Additional assets and impact for a micro-services design

Asset	Name	Description	CIA	Impact	Level
A005	Architecture	Data controlling the design of the system, creation of micro-services, databases, trust zones, rules for permitted connections between micro-services.	I	If altered, the system's behaviour could be adjusted, resulting in the loss of confidentiality of subject secrets, or loss of availability of the system.	High
A006	Administration account	Account and credentials required to construct or modify system architecture.	C	If confidentiality of the credentials is compromised, any attacker can alter the system's architecture and change any part of its behaviour.	High
A006	Administration account	Account and credentials required to construct or modify system architecture.	IA	If access to the account is lost (ie. if the bill cannot be paid, the credentials are modified without knowledge of the operator, or the cloud hosting provider prevents access for some other reason), either: 1. the service will be shut down (in which case, switches become unavailable), or 2. it becomes difficult to patch or update the service. Eventually exploits will be found that can be used to attack it.	High
A007	DMZ services, input zone services, data zone services	Web services in the DMZ responsible for creating new switches, and for accepting evidence of subject aliveness; services in the input zone that coordinate creation of switches and recording of subject aliveness evidence; services in the data zone that store operational and secret data, and share it with the check zone.	IA	If services in any of these zones are compromised, it may be possible to alter the behaviour of the service - either to make it unavailable, or to reject user input. It may also be possible for a compromised service to record the user's credentials and use them to play back false aliveness evidence at a later date.	High
A008	Subject aliveness evidence secret	A secret that the subject shares with their aliveness assertion.	C	If this becomes known, an attacker may use it to submit false aliveness evidence.	High
A009	Data zone services	Services in the data zone that store operational and secret data, and share it with the check zone.	C	If the data in the secret store is released, it will be encrypted with a key retained in the HSM, and so of little use to an attacker. However, operational information (pseudonyms and data used to verify aliveness information) may be uncovered and this could be used to manipulate the switch into accepting false aliveness information.	Medium

[46] CIA: Confidentiality, Integrity, Availability

Asset	Name	Description	CIA	Impact	Level
A010	HSM	The HSM in the data zone contains keys required to decrypt subject secrets.	C=A	If the HSM is compromised it is expected to render itself inoperable. This results in user secrets becoming unavailable. If this can be detected, users will be forced to resubmit their secrets in order to restore operation.	Medium
A011	Check zone services	Services in the check zone responsible for checking aliveness, extracting secrets, and publishing if necessary.	IA	If the secret extractor is compromised, it could lead to loss of confidentiality of the user's secret. If either service is altered or disabled, it could lead to the service becoming unable to carry out secret extraction and publishing if the subject becomes unavailable.	High
A012	Operator	The person or persons that operate the system	IA	If the operator is attacked or compromised in some way, the could be persuaded to release secrets permitting an attacker to take control of the system, learn subject secrets, or shut down the service.	High

As the purpose of the system is to store subject secrets and to release them when required, and because the impact of failing to do so could put a person at risk of attack, almost every asset here results in a high or medium risk should it lose CIA properties.

8.1.3.2 Vulnerabilities A number of basic vulnerability types are apparent across the system, and presented in Table 45.

Table 45: Vulnerability types for a micro-services design

Vuln ID	Vulnerability	Assets affected	Description	Vuln level
V001	Human weakness	A012 (operator)	Staff at the cloud platform provider, or the operator, may be vulnerable to bribery, threat, or disgruntled staff may act on malice. Humans are vulnerable to duress, and also make mistakes like having guessable passwords.	M
V002	Configuration errors	A005 (architecture), A007 (services), A009 (services), A010 (HSM), A011 (services)	Mistakes or deliberate errors in the configuration for security appliances (such as the firewall), any backups, micro-services or the HSM.	M
V003	Coding flaws	A007 (services), A009 (services), A011 (services)	Weaknesses in the micro-services themselves.	M
V004	Unpatched software vulnerabilities	A007 (services), A009 (services), A010 (HSM), A011 (services)	Managed micro-services are considered to be at low risk from OS patching. However, the micro-services themselves will need regular patching as new vulnerabilities are discovered.	M
V005	Default passwords	A006 (admin acct), A010 (HSM)	Accounts used to manage infrastructure with default passwords.	H
V006	Physical vulnerability of data centre	A005 (architecture), A007 (services), A009 (services), A010 (HSM), A011 (services)	The data centres administered by large cloud providers are considered to have adequate security in place to ensure only authorised staff may enter for good reasons.	L
V007	Capacity limits of architecture	A007 (services)	It may be possible to overwhelm the service's public facing endpoints.	M

Vuln ID	Vulnerability	Assets affected	Description	Vuln level
V008	Legal compulsion	A012 (operator)	Staff at the cloud platform provider, or the operator, are more likely to choose cooperation with a legal request over a hefty fine or a custodial sentence.	H

Each vulnerability represents a way that one or more asset could be compromised in the system.

8.1.3.3 Threats Threats are estimated by combining information about the vulnerability level, and the capability of threat actors. These are evaluated in Table 46.

Table 46: Threats for a micro-services design

Threat ID	Threat	Actor	Motive	Cap.	Target Assets	Method	Freq.	Notes
T001	Denial of service	Nation state (extra-legal means)	Prevent the service from operating, to protect their own secrets	H	A007 (services)	Overwhelm the front-end services with requests.	L	Considered Low frequency in some democracies, but far higher (or even automatic) in some countries.
T002	Denial of service	Organised crime	Prevent the service from operating, to protect their own secrets	H	A007 (services)	Overwhelm the front-end services with requests.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.
T003	Denial of service	Activists	Prevent the service from operating because it is perceived as unethical to withhold evidence of wrongdoing	M	A007 (services)	Overwhelm the front-end services with requests.	M	This could form a part of an ongoing campaign if activists are sufficiently motivated.
T004	Exfiltrate secrets	Nation state (extra-legal means)	Learn secrets stored by dissidents or investigative journalists investigating government wrongdoing	H	A001-A004 (information), A009 (data zone), A011 (check zone), A012 (operator), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	M	This is an attractive target for an intelligence agency that wishes to control dissidents and investigative journalists.
T005	Exfiltrate secrets	Organised crime	Learn secrets stored by enemies or people they wish to blackmail / Sell the information or accept payment not to release it	H	A001-A004 (information), A009 (data zone), A011 (check zone), A012 (operator), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.

Threat ID	Threat	Actor	Motive	Cap.	Target Assets	Method	Freq.	Notes
T006	Exfiltrate secrets	Activists	Embarrass the service, and leak secrets because it is perceived as unethical to withhold evidence of wrongdoing	M	A001-A004 (information), A009 (data zone), A011 (check zone), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, or by exploiting vulnerabilities in the micro-services.	L	Activists are considered to have limited resources to do this repeatedly, and may not need to - as the goal of reputational damage to the service would be achieved if sufficiently many subject's secrets were compromised.
T007	Destroy secrets	Nation state (extra-legal means)	Prevent evidence of government wrongdoing from being released	H	A001-A004 (information), A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	Considered Low frequency in some democracies, but far higher (or even automatic) in some countries with less regard for rule of law.
T008	Destroy secrets	Organised crime	Prevent evidence of their own wrongdoing from being release, or make it possible to attack an enemy	H	A001-A004 (information), A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.
T009	Destroy secrets	Activists	Prevent the service from operating because it is perceived as unethical to withhold evidence of wrongdoing	M	A001-A004 (information), A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, or by exploiting vulnerabilities in the micro-services.	L	Activists are considered to have limited resources to do this repeatedly, and may not need to - as the goal of reputational damage to the service would be achieved if sufficiently many subject's secrets were deleted.
T010	Install malware	Nation state (extra-legal means)	Make service inoperable, with plausible deniability	H	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	L	This is difficult to do on a managed micro-services infrastructure - and once done, there would be little benefit to doing it again, as the service would be disabled.
T011	Install malware	Organised crime	Exploit computation resource to make some money, or ransom back the capability to operate	H	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	M	This is difficult to do on a managed micro-services infrastructure, but organised crime are incentivised to do it repeatedly if they can.
T012	Install malware	Script kiddies	Excitement, reputation	L	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	M	This is difficult to do on a managed micro-services infrastructure, but persistent *attempts* are to be expected by any service.

Threat ID	Threat	Actor	Motive	Cap.	Target Assets	Method	Freq.	Notes
T013	Weaken secret protections by altering architecture	Nation state (extra-legal means)	Learn about future secrets and users	H	A005 (architecture), A006 (admin acct)	Adjust the architecture to remove protections for secrets, or install additional accounts that can be used to obtain secrets in future, by learning or guessing the administrator password (or obtaining it through duress); or by making imperceptible changes to the architecture designs and waiting for the operator to refresh the system from this data.	L	This activity can be detected, and is not considered the easiest way to achieve the same weakening of the service's secret protections. However, it may serve a purpose and could be quicker to implement than legal compulsions.
T014	Weaken secret protections by mandating a backdoor	Nation state (legal means)	Learn about future secrets and users	H	A005 (architecture), A006 (admin acct)	Take the operator or platform service provider to court, or threaten to, to incentivise them to install a backdoor (ie. an additional account), or alter the micro-services, to make it possible for a government body to obtain the secrets stored in the DPS.	M	Wide-reaching mass surveillance programmes (such as Prism) have been revealed in recent years. Data collection was achieved by legal (and other) means. New programmes would be equally hard to discover. It is likely that major providers are collaborating with law enforcement, or intelligence agencies, to some degree.
T015	Weaken secret protections by altering architecture	Organised crime	Learn secrets stored by enemies or people they wish to blackmail / Sell the information or accept payment not to release it	H	A005 (architecture), A006 (admin acct)	Adjust the architecture to remove protections for secrets, or install additional accounts that can be used to obtain secrets in future, by learning or guessing the administrator password (or obtaining it through duress); or by making imperceptible changes to the architecture designs and waiting for the operator to refresh the system from this data.	L	Organised crime may not care if their changes are discovered provided they can exploit them reasonably swiftly.
T016	Shut down service with court order	Nation state (legal means)	Prevent use of the switch for a variety of reasons	H	A001-A004 (information), A006 (admin acct), A012 (operator)	Take the operator, or the cloud platform provider, to court (or threaten to) to incentivise them to halt the account hosting the DPS.	L	Shutting down the entire service seems less useful to a nation state than exploiting it to gather information.

Threat ID	Threat	Actor	Motive	Cap.	Target Assets	Method	Freq.	Notes
T017	Fines for switches used for blackmail	Law enforcement	Enforce blackmail legislation	H	A012 (operator)	Take the operator, or the cloud platform provider, to court for alleged use of the DPS for blackmail.	M	This could happen. It is likely that individual cases would be dealt with, ie. an individual DPS might attract attention of law enforcement, and the court case may require only the removal of an individual switch.
T018	Fines or jail for refusing to release secrets	Law enforcement	Support criminal investigations, incentivise release of secrets stored by people engaged in illegal activity	H	A012 (operator)	Threaten the operator with fines, or a custodial sentence, to incentivise them to release secrets belonging to a person of interest.	L	This is a more extreme case, where law enforcement wish to learn the information kept in the switch.

The threats illustrate a number of different ways that threat actors may attempt to attack the system - estimating their capability and the expected frequency of such attacks.

8.1.3.4 Risks Table 47 presents evaluated risks, showing calculated estimates for likelihood of success, threat level, and risk level. These calculations are based on combinations tables described in section 4.5.1.

Risk prioritisations are derived from the (slightly biased) combination table shown in Figure 20.

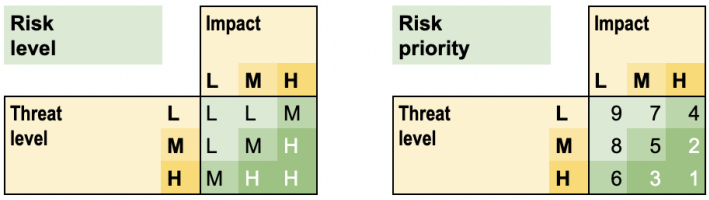


Figure 20: Risk level and priority combination tables

Table 47: Risks for a micro-services design

Risk ID	Threat IDs	Vuln IDs	Asset IDs	Risk	Capability	Vuln level	Likelihood of success	Freq. of attempt	Threat level	Impact	Risk level	Risk priority
R001	T001, T002, T003	V007	A007	DoS attack by overwhelming the public facing endpoints.	H	M	H	L	M	M	M	5
R002	T004, T005, T007, T008	V001, V002, V005	A001-A004, A006	Exfiltrate or destroy secrets by guessing or learning the administrator credentials.	H	H	H	M	H	H	H	1
R003	T004, T005, T007, T008	V002, V003, V004	A001-A004, A009, A011	Exfiltrate or destroy secrets by exploiting vulnerabilities in the micro-services.	H	M	H	M	H	H	H	1
R004	T004, T005, T006, T007, T008, T009	V001	A001-A004, A012	Exfiltrate or destroy secrets by learning the administrator credentials through duress.	H	M	H	M	H	H	H	1

Risk ID	Threat IDs	Vuln IDs	Asset IDs	Risk	Capability	Vuln level	Likelihood of success	Freq. of attempt	Threat level	Impact	Risk level	Risk priority
R005	T010, T011, T012	V002, V003, V004	A007	Install malware by exploiting vulnerabilities in the micro-services.	H	M	H	M	H	H	H	1
R006	T013, T015	V002, V003	A005	Adjust the architecture to remove protections for secrets by making imperceptible changes to the architecture files and waiting for the operator to refresh the infrastructure.	H	M	H	L	M	M	M	5
R007	T013, T015	V001, V005	A005, A006	Adjust the architecture to remove protections for secrets by learning or guessing the administrator credentials.	H	H	H	L	M	M	M	5
R008	T013, T015	V001, V005	A005, A006, A012	Adjust the architecture to remove protections for secrets by learning the administrator credentials through duress.	H	H	H	L	M	M	M	5
R009	T014	V008	A005, A006	Weaken secret protections by mandating a back door through legal means.	H	H	H	M	H	M	H	2
R010	T016	V008	A001-A004, A006, A012	Shut down the service by threatening the operator with court (or taking them to court and threatening them with a file or custodial sentence).	H	H	H	L	M	H	H	1
R011	T017, T018	V008	A001-A004, A006, A012	Shut down an individual subject's switch by taking the operator or cloud platform provider to court for aiding blackmail, or other legislation in criminal investigations.	H	H	H	M	H	L	M	7

This table is more clearly presented in Appendix G.

8.1.3.5 Responses and controls There are a number of possible techniques/controls that can help to mitigate the risks. These are classified by a common taxonomy describing their strategic function. Controls may:

- avoid/terminate a risk - ie. by not using the functionality that leads to the risk.
- transfer/share a risk - ie. by contracting a third party, or by taking out insurance.
- reduce/modify a risk- ie. a control that reduces the frequency or impact of a risk.
- accept/tolerate a risk - considered distinct from ignoring a risk (not an option).

Within a strategic function, controls represent tactical options. Controls may:

- detect a risk - ie. advise or warn when an incident is occurring
- correct a risk - ie. reduce the impact or the likelihood of success
- prevent a risk - ie. reduce the likelihood an attempt
- represent a directive - ie. policies and procedures for people

At the operational level, controls are classified as:

- physical controls - eg. a lock on a door
- procedural controls - eg. a clear desk policy
- technical controls - eg. the use of antivirus software

For each risk, responses are chosen and a number of controls are proposed to help mitigate the risks presented, and these are presented in Appendix G.

A lot of the controls proposed align with common application security advice - such as that found in NCSC guidance on password strength [60]. Where the administration credentials pose a high risk, for instance, requiring strong passwords or 2FA[47] as policy can help to mitigate the risk that a password can be easily guessed.

Much of the advice is also deemed affordable, and includes common mitigations for risks such as: backup regimes, monitoring, code review processes, code dependency analysis. Some may cost a little more - such as relocating to a cloud service provider in a country with a regime unlikely to interfere with the operation of the service.

8.1.3.6 Further considerations When reflecting on the risk model presented, a number of additional considerations present themselves:

- The risk model does not incorporate risk of insider attack.
  - A common mistake, repeated across organisations is to assume that staff are not motivated to abuse their levels of access to data.
  - In this case, the staff operating the switch (likely a small team), have been assumed to be very motivated and unlikely to attack the system (unless attacked themselves, and compromised in some way).
  - Implementation of additional monitoring and careful management of staff accounts (ie. requiring elevation of privileges to access sensitive parts of the system for specific reasons) can help to reduce the likelihood of insider attacks.
  - Similarly, an inclusive and non-toxic work environment can also reduce the motivations that drive some disgruntled insiders to steal data.
- The system does not incorporate a backup/restore capability.
  - This may be provided as a feature by the cloud service. However any offering should be fully understood, as data could now be stored in multiple locations, and may be encrypted with keys that must be protected.
  - It is especially important not to leave unencrypted data at rest, as this data will be stored in backups with different security properties.
- The system does not incorporate monitoring capabilities.
  - Logs should be collected and periodically reviewed to record and assess the behaviour of the system.
  - Alerts should be generated for unexpected behaviour, or availability issues.
  - Periodic review of logs and alerts will help to decrease the likelihood that a breach (or attempted breach) will go undetected.
- The system is not resilient to denial of service attacks that attempt to overwhelm it.
  - High demand: Load balancing and autoscaling micro-services.
  - Denial of service attacks: Frontend protection, eg. Cloudflare
- The system endpoints are identifiable as residing on a given service.
  - Adding a reverse-proxy makes it much harder to learn where the service actually resides.
  - Frontend protection such as Cloudflare may also offer a reverse-proxy functionality.
- Although the system does not feature 'admin bypasses,' its infrastructure is administered by a cloud services operator.
  - Create separate accounts with limited access for everyday use.

[47] Two-factor authentication (2FA) - a second form of identification.

– Elevate privileges to read logs, modify the architecture, or upload changes to the micro-services.
– Monitor and alert for activities like these.

8.1.4 Evaluation against requirements

8.1.4.1 Confidentiality The system, as designed, works hard to preserve the confidentiality of subject secrets. The secret store is separated from other processes and the slightly less sensitive operational data store, and its contents are encrypted with keys managed by an HSM. This means that even if the code and data are stolen, they cannot be decrypted except in the managed, hosted environment.

Only administrators may alter the operation of the system, and to do so would require the alteration of, or introduction of new, micro-services. Controls mentioned in the risk review, and below when discussing resilience, help to mitigate this risk.

8.1.4.2 Awareness The switch is designed to only release the subject's secret should they stop providing evidence of aliveness. That evidence is signed by the HSM on storage, and verified by the aliveness checking service on retrieval. This provides a strong guarantee that the evidence was recorded at a given time, and believed by the system to be coming from the subject.

The aliveness signal chosen is a secret (eg. a code phrase) shared by the system and the subject. If the secret can be guessed the system's ability to detect when the subject has been attacked is compromised. Mitigations proposed for credentials in the risk review can also be applied. Strong code phrases, and use of 2FA, increase the effort required to guess the secret.

8.1.4.3 Timing The aliveness checks are initiated by a regular timed trigger in a trust zone far removed from public access. It is considered very difficult for an attacker to compromise it, or cause it to fail.

8.1.4.4 Resilience A risk review such as the one shown, if carefully applied can help to mitigate the risks to a system from a number of known attackers.

It does, however, indicate a number of weaknesses of the system, too:

- The risk review recommends compliance in the face of legal requirements, and law enforcement requests (suggesting that they each be contended in court to ensure that the reason behind them can be evaluated). This is noted to be a compromise required to keep the system running for other users.
- Some attacks are very difficult to protect against, and the review recommends providing a backup regime to ensure that if the service is attacked and disabled, it can be restored.

It is important to understand that adding backups to a service increases the attack surface. The backups themselves must be incorporated into a new design and risk model to ensure that they do not increase risk.

This design has reasonable protections against remote attack built in to it, including patching and code reviews to help reduce the number of vulnerabilities available to exploit.

Clean separation of the write and read capabilities into trust zones helps to ensure that even if an individual's account were attacked, their secret remains safe (and so can be released when the system realises that they are unavailable).

The model does not incorporate the risk from internal attacks, which are often overlooked by small and medium-sized enterprises. A number of motivations exist for internal threats. *The CERT Guide to Insider Threats* [61] contains several models describing insider threat motivations and activity, supported by case files. It presents 3 models of *malicious* insider threat:

- Insider IT sabotage
- Insider theft of intellectual property
- Insider fraud

A number of controls exist, to help detect and mitigate insider threats before they are realised. Exploring IT sabotage, the CERT Guide indicates:

"Most insiders who committed IT sabotage were disgruntled due to unmet expectations"

It also offers a number of behavioural precursors:

- Conflicts with co-workers or supervisors
- A sudden pattern of missing work, arriving late, or leaving early
- A sudden decline in job performance
- Aggressive or violent behaviour
- Sexual harassment
- Poor hygiene

In a small organisation, it may be enough to work closely together, observe each other, be aware of the precursors, and have some good controls in place - such as code review, pairing to complete sensitive tasks, fair workplace rules[48], and regular evaluations to ensure that employee expectations are being met.

8.1.4.5 Affordability No billing model is attached to the design, and the risks here remain unassessed. In particular, adding billing to a DPS has a number of consequences to the resilience of the system:

- If the billing system cannot be made anonymous, it presents a risk to the subject and the system by linking them.
- If traditional banking is used, a user's bank account might become another attack surface.
- Billing may need to store information which, if compromised, could lead to attacks such as the freezing of an individual's assets (which would result in their account with the DPS closing and their switch being disabled). In many countries, it is a legal requirement to keep records of all transactions for tax and fraud purposes.

This system is designed to support multiple users, and so if a billing system were put in place, it is likely that operating costs will not be excessive. This, in turn, means that a regular subscription model is likely to be affordable to users such as journalists or the organisations that support them.

8.1.4.6 Durability Managed micro-services on a cloud platform will receive regular upgrades and patches. The software will still need to be maintained, but this provides a degree of support for the operating systems that the services rely on.

Billing is not considered in this design. Without it, the service will not be able to make payments to the cloud platform provider - and risks shutting down as and when the budget runs low. Implementing a billing system that can protect users' privacy, but secure the future of the service, is essential.

8.1.4.7 Explainability A classic micro-services design benefits from the simplicity of its components. Each is a small process, dedicated to one task. When arranged in a diagram, as in figure 19, it becomes easy to explain how the system works.

However, as with all other solutions, the threats to the system, and its mitigations require some awareness of security practices and will need some work to make them clear for lay-persons. Some good first steps towards explainability:

- Publish the code as an open source solution, and encourage security researchers to explore it.
- Demonstrate responsiveness to responsible disclosure reports, and explain the patches made in a blog.
- Publish the architecture diagram, with an explainer.
- Publish the risk assessment, with an explainer.

In addition, endorsements or explainers from trusted figures in academia and software development may go some way towards helping potential users to develop trust in the system.

8.1.4.8 Visibility In its current state, the design does not report on which switches are active. It would not be difficult to develop some additional functionality to publish information about the switches, possibly including "sample" information to show that real incriminating information is held.

[48] (including immediate repercussions for infractions such as aggressive, violent behaviour or sexual harassment)

Of course doing so introduces an additional endpoint, and that increases the attack surface of the solution - so it must be done with caution. Instead of hosting an additional endpoint, perhaps the system can push this sample information to the publishing medium where it can be seen without exposing the location of the DPS itself.

8.2 Design 2: dApp with managed secrets

8.2.1 Premise

This alternate design builds on the premise that smart contracts can provide a number of key requirements for a DPS. Smart contracts have high availability and integrity properties - as information held on blockchains is very difficult to censor or repudiate.

However, as smart contracts are executed across a distributed network, and because the isolation requirements of the EVM[49] cannot be guaranteed, they are unable to protect the confidentiality of any data they hold. Another component is required to meet this requirement and protect the subject's secret.

Existing dApp solutions, described in section 6.2, manage this in a couple of ways:

- **KillCord** relies on a hidden, trusted publisher application to store and release the decryption key for the subject's secret.
- **Kimono** distributes SSSS fragments of the decryption key amongst a trusted group of participants, who are incentivised to release at the right time.

This design relies on a third party network, which can offer similar availability and integrity properties as a blockchain smart contract, but can also offer confidentiality properties. SECRET (described in section 8.2.2.1) is one such network - and can bridge to other, more public networks, such as Ethereum.

Finally, the design uses the InterPlanetary Filing System (IPFS, described in section 7.5.4) to distribute the encrypted subject secret - so assuring its wide availability and resistance to censorship.

8.2.2 Research and components

The properties of smart contracts on cryptocurrency networks are discussed in section 7.4.2.2.

8.2.2.1 SECRET network Formerly known as *ENIGMA* [62], the *SECRET Network* [63] addresses the issue of confidentiality amongst smart contracts. As described on their site, scrt.network:

> "SECRET Network is built as a decentralized network of computers (secret nodes) utilizing trusted execution environments (TEEs) to enable secure, private computation over encrypted data."

Trusted Execution Environments (TEEs, described in section 7.3.3), such as SGX, offer private computation facilities - protected from the host operating system and any software running on the device. As such, they make a good candidate for the execution of *secret contracts* (smart contracts equipped to run in TEE enclaves, and to operate on data that remains secret), whilst allowing a network of nodes to agree on their outcomes, and the state of the network, by consensus.

Secret contracts have all the benefits of smart contracts, but add the confidentiality property noted missing from other dApp networks.

The SECRET network ensures that nodes on its network execute transactions and smart contracts in TEEs to ensure that the confidentiality of the executing code and data is preserved.

The Secret Network operates as a "Layer One" solution, and so it has its own consensus and can operate without any reliance on other blockchains.

At "Layer Two" it has a number of bridges built out to other networks (eg. an Ethereum Bridge), allowing activity initiated in another blockchain (say, by a non-secret smart contract) to trigger a secret contract in the SECRET Network - and so perform private activity there before returning a result to the originating smart contract. This allows for some dApp designs spanning more than one network.

[49] The Ethereum Virtual Machine (EVM) is a virtual machine specified to execute smart contracts for the Ethereum network.

8.2.3 Design

This design proposes a combination of:

- An Ethereum smart contract, responsible for:
  - managing evidence of aliveness,
  - triggering the secret contract.
- A secret contract on the SECRET network, responsible for:
  - holding and releasing the subject's secret decryption key when appropriate.
- IPFS, responsible for:
  - distributing an encrypted copy of the subject's secret.

It is illustrated by a number of processes.

8.2.3.1 Process: Creation

This process is illustrated in Figure 21.

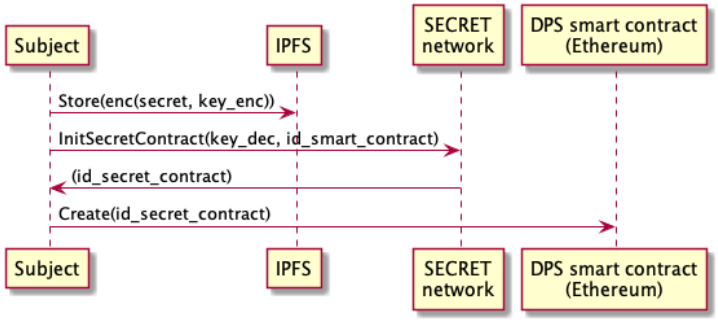


Figure 21: Design proposal 2: dApp creation flow

1. The subject first pushes the encrypted version of their secret to IPFS.
2. The subject creates a secret contract on the SECRET network, giving it the identity of the DPS smart contract on Ethereum that manages aliveness evidence, the subject's Ethereum identity, and the decryption key for their secret.
3. The subject initiates a new DPS by calling the Create method on the DPS smart contract on the Ethereum network.
   - This smart contract will store details of their identity, and the identity of their secret contract.

8.2.3.2 Process: Check-in

This process is illustrated in Figure 22.

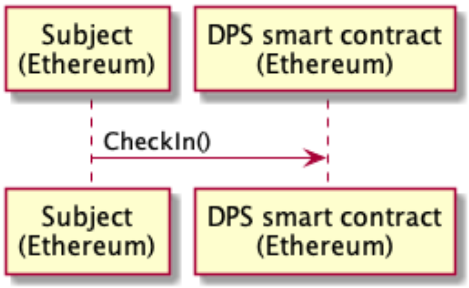


Figure 22: Design proposal 2: dApp check-in flow

1. The Ethereum smart contract will store the subject's last known check-in time, trusting only the subject's Ethereum identity to update it.

8.2.3.3 Process: Activation

This process is illustrated in Figure 23.

1. A reliable Timer within the Ethereum network causes the DPS smart contract on the Ethereum network to call `Trigger` function on the subject's secret contract on the SECRET network.

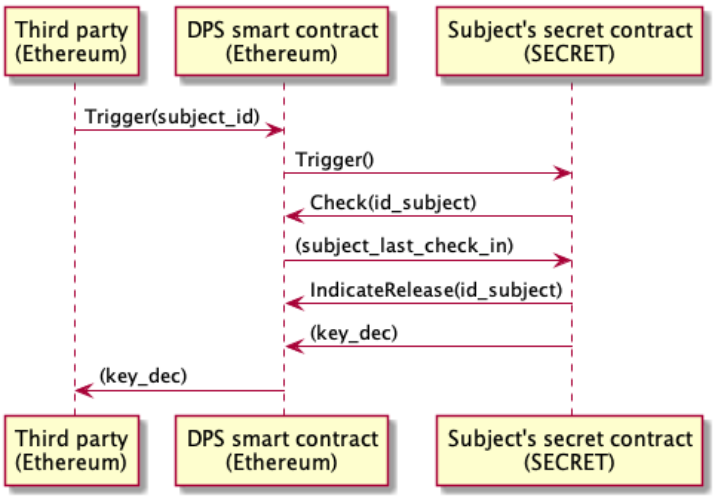


Figure 23: Design proposal 2: dApp activation flow

2. The subject's secret contract queries the Ethereum smart contract by calling the `Check` method with the subject's identity. The smart contract returns the time of the user's last check-in.
3. If the required time has elapsed without a check-in from the subject, the secret contract will then:
   - Indicate to the DPS smart contract that it has released the subject's secret by calling the `IndicateRelease` function with the identity of the subject.
   - Return the decryption key to the DPS smart contract.

8.2.3.4 DPS smart contract The design of the Ethereum DPS smart contract is represented below with simplified pseudo-code.

A simplified C-type language syntax is used for clarity, and the contract represents contracts.

Assumptions:

- The current time is available to the contract as `time.now`.
- The identity of the account that called the smart contract is available as `current_user.id`.

```
Map<id,bool> subjects_switch_created;
Map<id,time> subjects_last_check_in;
Map<id,id>   subjects_secret_contract_id;
Map<id,bool> subjects_secret_released;

// create a DPS for the subject, allowing the secret contract to communicate with it
Create(id_secret_contract)
{
    subjects_switch_created[current_user.id] = true;
    subjects_last_check_in[current_user.id] = time.now;
    subjects_secret_contract_id[current_user.id] = id_secret_contract;
    subjects_secret_released[current_user.id] = false;
}

// allow the subject to check-in
CheckIn()
{
    if (subjects_switch_created[current_user.id])
    {
        subjects_last_check_in[current_user.id] = time.now;
    }
}
```

```
// allow any third party to learn the subject's secret id if appropriate
Trigger(id_subject)
{
    if (subjects_switch_created[id_subject])
    {
        return Call(subjects_secret_contract_id[id_subject], "Trigger");
    }
}

// allow the secret contract to learn the subject's last check-in
Check(id_subject)
{
    if (subjects_switch_created[id_subject] &&
        current_user.id == subjects_secret_contract_id[id_subject])
    {
        return subjects_last_check_in[id_subject];
    }
}

// allow the secret contract to indicate that it has released the subject's secret
IndicateRelease(id_subject)
{
    if (subjects_switch_created[id_subject] &&
        current_user.id == subjects_secret_contact_id[id_subject])
    {
        subjects_secret_released[id_subject] = true;
    }
}

// allow a third party to check if a subject's secret was released
IsReleased(id_subject)
{
    return subjects_switch_created[id_subject]
        && subjects_secret_released[id_subject];
}
```

This is a simple contract, but it illustrates the key functions that are required - allowing it to serve as a highly available, highly integral, and non-repudiable source of evidence for subject aliveness evidence.

8.2.3.5 Subject's secret contract By comparison the secret contract has less to do - once initialised, it is responsible for running a check against the DPS smart contract to determine if too much time has passed since the subject's last check-in. If so, it responds with the subject's secret decryption key.

This can also be represented as pseudo-code:

```
id dps_smart_contract;
id subject_id;
byte[] key_dec;

// expect to be called by the DPS smart contract,
// allow any party to initiate a check on a subject's well-being
Trigger()
{
    last_check_in = Call(dps_smart_contract, 'Check', subject_id)
    if (time.now - last_check_in > Time.Hours(48))
```

```
    {
        Call(dps_smart_contract, 'IndicateRelease', subject_id);
        return key_dec;
    }
}
```

8.2.4 Evaluation against requirements

8.2.4.1 Confidentiality An encrypted copy of the subject's secret is distributed to IPFS when the DPS is initialised. The confidentiality of the decryption key represents the confidentiality of the secret.

This design stores the subject's secret decryption key in a secret contract on the SECRET network. This secret contract benefits from the security properties of the SECRET network, which in turn relies on TEEs to preserve confidentiality of the secret contract's state.

Although TEEs, like every technology, have undiscovered vulnerabilities - dedicated hardware for confidential computation is considered a strong protection.

The nature of this design, and the guarantees made by smart contracts, mean that once it has been set up there are no users (with administration privileges or otherwise) that can retrieve the encryption key until the appropriate time.

8.2.4.2 Awareness This design uses a call to a `CheckIn()` method on a public Ethereum smart contract from the subject's Ethereum identity as a proxy for their aliveness.

This is considered a good proxy. Ethereum identities are well guarded, and access to them relies on secrets (and sometimes hardware, such as a 2FA or YubiKey device) that only the user will know (or have).

8.2.4.3 Timing This DPS only activates when triggered by a curious party - a call to the `Trigger()` function on the DPS smart contract is enough to set it in motion, after which it will return the subject's decryption key if they are considered no longer available. This design does not poll, and this may be problematic for some subjects - who may wish the system to announce their disappearance, rather than rely on the curiosity of others.

8.2.4.4 Resilience The qualities of smart contracts, secret contracts, and IPFS make this design very resilient to attack. If the guarantees made by these networks are upheld, then this switch is resilient against attacks on its CIA properties:

- Confidentiality: The SECRET network is designed to protect the confidentiality of the state of its secret contracts, using TEE hardware to support that.
- Integrity: Both SECRET and Ethereum operate by consensus - so making it very difficult to attack and alter the outcome of the contracts' behaviours. IPFS, too, does not allow its data to be altered after distribution - and data is addressed by hash, so making it impossible to retrospectively change 'in place.'
- Availability: SECRET, Ethereum, and IPFS are all distributed networks. It is very difficult to censor them - as their operations are played out across many nodes.

Possible attacks might include:

- 51% attacks - where a system attempts to gain control of more than half the compute power on the Ethereum network.
- Transferring the attack to the user, and coercing them to surrender access to their Ethereum account.

8.2.4.5 Affordability

- The secret contract must be called to trigger the DPS smart contract on Ethereum. This costs money.
  - In fact, every action costs money.
  - It may be affordably small, but it is constant.

  - Unlikely to prove a significant disincentive to people searching for the subject's secret, but it is not ideal - the distribution of the secret should be frictionless.

8.2.4.6 Durability The Ethereum network is well established, and from 2022 will be distancing itself from wasteful proof-of-work mining activity - saving its participants 99.9% of their power consumption. This is likely to make Ethereum popular cryptocurrency network over the next few years. A willingness to adapt and upgrade technologies is an important aspect for a currency that wishes to endure.

The future of other technologies involved is not known.

The SECRET network solves the problem of confidentiality by building a network that relies on TEEs. It is still in development, and has many other features not provided by mainstream networks. It may find itself superseded in future, though, if other cryptocurrency networks adapt and develop these capabilities for themselves. At current time, Windows 11 relies on Trusted Computing features such as TPMs [64], and could well mandate additional features such as TEEs in future. In turn, this will raise the likelihood that more consumer devices will have this capability.

IPFS is currently in use by a large number of projects [65]. It is developed and maintained by Protocol Labs in Palo Alto, California. Wikipedia notes that "as of 2021, it has 130 members, $55.7M in funding" [66].

8.2.4.7 Explainability Distributed apps are more difficult to explain, but do benefit from a number of properties that a lay person can understand:

- They run on many computers, and so cannot be stopped.
- They are checked by many computers, and so once created cannot be changed.

These properties, combined with a good explainer, may go some way towards helping users develop trust in the system.

As with all systems, providing the code as an open source solution and subjecting it to review from security researchers goes a long way towards helping establish that it is a trustable system. In addition, endorsements or explainers from trusted figures in academia and software development may also help potential users to develop trust in the system.

8.2.4.8 Visibility The aliveness checks for this system operate on a public blockchain, and so are visible to a party that is looking for it. With a little additional work, the secret contract could be adapted to store 2 keys: One that it released on request, which can decrypt a small portion of the subject's secret, and a second which decrypts the remainder. This would demonstrate clearly to a potential attacker that the secret is real, contains valuable information, and can be decrypted.

8.2.5 Further considerations

A number of outstanding issues arise from this design:

- The user is still susceptible to physical coercion by an attacker to grant access to their Ethereum identity.
  - A subject who anticipates this might choose a particularly difficult secret deliberately in order to be able to air the opinion that they might forget it under duress.
  - This could further disincentivise an attack if the attacker knew it wouldn't help them gain control of the DPS.
- The design of the smart contracts makes an assumption that a value representing the current time is available and correct.
  - In particular, the design should not trust its host machine to provide the correct time - and this could lead to .
  - One solutions is described in *Decentralizing a Dead Man's Switch* [26]. A cluster of smart contracts are tasked with time keeping, and communicate the length of the current blockchain as a proxy for the current time.
  - Use of a blockchain as proxy for the passage of time is described in section 8.3.2.2.
  - A full implementation would be written in Solidity for the EVM, and should incorporate an understanding of all the pitfalls involved in writing smart contracts [25].

- The subject must either be knowledgeable enough to communicate with smart contracts, and to create their own secret contract; or they must trust a third party to do it for them.
  - Open source code can help here - as it gives opportunities for people and organisations the subject trusts to create tutorials, or services to help them.
- The design is pretty simple, and limited to 1 secret per Ethereum identity.
  - As a user can create a separate wallet to operate the DPS, this may aide user privacy by serving as a pseudonymous identity.

8.3 Design 3: Witness encryption

8.3.1 Premise

Unlike classic cryptography, where a user is present with a secret key that can be used to decrypt a ciphertext, a DPS must decrypt and share a secret when that user is not present - and so reliance on the provision of a key is not sufficient.

This design draws on the concept of *witness encryption* - a generalisation of encryption that permits a witness (a mapping in an NP language) to serve as the decryption key for a ciphertext.

It relies on the notion that a subject's activity on a cryptocurrency blockchain constitute a good proxy for evidence of life - and proposed that a witness encryption scheme could be derived that relies on this activity to provide a valid decryption witness.

The implication of this is that an entirely passive scheme can be constructed that allows a person's encrypted secret to be published, which becomes easily decryptable by any party should their proxy for aliveness not appear on the blockchain for a given amount of time[50].

8.3.2 Research and components

8.3.2.1 Witness encryption Witness encryption, described by Garg et al. in 2013 [67], provides a means to generalise encryption and decryption operations beyond the use of keys.

Witness encryption offers a way to predicate the successful decryption of a ciphertext on the provision of a solution to a 'hard' puzzle, described as an NP language, L. It is hard to determine which values should be in L, but easy to verify a given *witness* (solution).

Cryptography of this sort is described with *circuits* - a series of logic gates which can be used to represent a particular function, and in this case the scheme relies on a circuit that is able to determine if a particular witness belongs to the language L - this is known as the prover.

$WE.Enc(1^\lambda, x, m)$ is the encryption algorithm that outputs a ciphertext, c, with inputs:

- 1^λ - a security parameter
- x - an string of any length
- $m \in M$ - a message

$WE.Dec(c, w)$ is the decryption algorithm, which either outputs m or $\perp$, with inputs:

- c - the ciphertext
- w - the witness, a bit-vector

In a more concrete example, perhaps a person wishes to prepare a prize (the key to a cryptocurrency wallet, for instance) for the solution to a puzzle that they do not know the answer for. (They may even prepare such a prize without knowing that a solution exists.) Examples could range from "a valid crossword puzzle" through to "the proof for an as-yet unsolved problem in mathematics." Provided a circuit can be constructed that verifies the answer, this can then be used as a component of witness encryption to encrypt the prize in such a way that only a verifiable solution will decrypt the prize.

The extremely general nature of witness encryption promises some exciting advances, however it also requires that the puzzle at the centre of such a scheme share the properties of an NP language.

A DPS must answer the question "is a particular person unavailable," or "has a particular person been attacked." In order to serve as the witness in an encryption scheme, validation of the proof of the subject's

[50] As cryptocurrency blockchains grow at an approximately steady rate, they can be used as a proxy for the passage of time.

well-being would need to be represented by a cryptographic circuit, and in order to have similar properties to an NP language, valid proofs should be infeasible to discover (ie. it shouldn't be practical to forge such a proof), but easy to validate.

Validating evidence that represents a person's state in the physical world to a sufficient level of confidence is a very complex problem, and beyond the scope of cryptographic circuits as understood today. It would rely on a number of corroborating sources, and verifications that bring with them a wealth of complexity and vulnerabilities.

This does not, at first glance, seem to lend itself to an NP problem. However, as discussed in section 7.2 various measures may be substituted as a proxy for aliveness.

8.3.2.2 Blockchain length as a proxy for passage of time To understand how a proxy for aliveness may serve as a witness, first it is helpful to examine how witness encryption can fashion a proxy for the passage of time in a time-locked encryption scheme.

This can be achieved by defining a witness as a valid blockchain (from a given starting block) of a given length.

The bitcoin blockchain can be represented as a sequence of hashes, conforming to some conditions that control how difficult it is to determine the next hash. It grows at an approximate steady rate (considered to be approximately 1 new block every 10 mins) - and this is managed by increasing the difficulty of creating new blocks depending on the resources currently contributing to the chain. As a result, the blockchain is a reasonable (if imperfect) proxy for the passage of time.

It is not feasible to create a valid blockchain significantly faster than cryptocurrency miners, as they are highly motivated, well resourced, and competing to uncover bitcoins and add the next block to the chain the quickest.

Liu, Jager, Kakvi and Warinschi describe the term *computational reference clock* and illustrate how the Bitcoin[51] blockchain can be used as such in *How to build time-lock encryption* [68], 2018.

To combine this with witness encryption, Liu et al. define the NP relation R as:

1. For $x \in N$, each statement has the form 1^x (the unary representation of x).
2. Any *valid* Bitcoin blockchain hash sequence[52] $w = (B_1, ..., B_x)$ of length at least x is a witness for statement 1^x.

ie. $(1^x, w) \in R$

For pure witness encryption, the size of the ciphertext has $O(N)$ complexity in relation to the size of the witness. Similarly, the encryption and decryption operations also have $O(N)$ complexity in time. This would be manageable complexity, but all forms of witness encryption available in 2018 are based on multilinear maps - which leads to a polynomial multilinear level, problematic for the size of the witness and complexity of the scheme.

To contain the level of multilinearity for witness encryption, Liu et al. apply SNARKs (Succinct, Non-Interactive Argument of Knowledge). This ensures that the multilinearity level of witness encryption remains constant.

The result is a potential application of witness encryption to time-lock encryption - designed to permit a user to encrypt a message which can only be decrypted with a valid blockchain of a predetermined length. This serves as a proxy for the passage of time - and so the scheme allows for the preparation and later decryption of a secret, only after a given amount of time has passed.

8.3.3 Design

8.3.3.1 Blockchain activity as a proxy for aliveness Having established that it is possible to predicate the decryption of a message on the provision of a valid chain of blocks, this design proposes an extension of the constraints of the witness encryption scheme to require additional properties from the chain:

[51] This generalises to blockchains beyond Bitcoin, of course.

[52] The hash sequence itself is most important here - the remaining content of the blocks (ie. other transaction data) can be ignored. Further, B_1 does not need to be the genesis block. It is possible to initialise the clock from the most recent block, and ignore older parts of the blockchain.

- A valid chain of given length without evidence of aliveness (here, activity from the user's cryptocurrency wallets).

This can be represented as an extension of the application described by Liu et al. redefining the NP relation R as:

1. For $x \in N$, each statement has the form 1^x (the unary representation of x).
2. Any *valid* Bitcoin blockchain hash sequence $w = (B_1, ..., B_x)$ of length at least x, where a specific Bitcoin wallet (or wallets) do not initiate a transaction for at least x blocks, is a witness for statement 1^x.

Although this significantly complicates the model (Liu et al. simply use each block's hash, but this requires transactional information from blocks, too) it leads to a witness encryption scheme where only a blockchain that does not contain a proof of life for the subject may be used to decrypt the secret.

8.3.3.2 Process The process for this proposed solution is very simple:

1. A subject creates two wallets for themselves in a cryptocurrency of their choice.
2. They construct a witness encryption scheme that requires a witness as described above, where the length of the required blockchain is set to the equivalent of approximately a week.
3. They encrypt their secret with the scheme and publish somewhere public, eg. IPFS.
4. They proceed to transfer a small amount of cryptocurrency between the wallets each week.

Should they ever stop, a valid blockchain that goes a full week without their transactions will serve as a valid witness to decrypt their secret. Provided it is easily discoverable, and instructions for applying the scheme to decrypt the secret are shared, this will result in the secret being decryptable by any party.

8.3.4 Evaluation against requirements

8.3.4.1 Confidentiality This scheme encrypts the subject's secret, and provided they delete their original, the secret is protected against any attempts to decrypt it without a valid witness.

8.3.4.2 Awareness Activity on the blockchain may be considered a good proxy for aliveness - although it isn't perfect.

Integrity of a user's evidence of aliveness is now as strong as the protection on the cryptocurrency wallets they use. A wallet that uses a biometric signal in addition to secrets/credentials could improve the quality of its proxy for aliveness and well-being. A custom wallet could be constructed to serve this purpose.

8.3.4.3 Timing Provided the blockchain continues to grow at a reliable rate, this scheme's ciphertext will not be decryptable until a given amount of time has passed. However, over the course of a person's lifetime blockchains may intentionally change their rates (or even the mechanism by which they operate).

8.3.4.4 Resilience If the assumption that such a scheme can be constructed and represents an NP language holds, this solutions is extremely resilient. The only way to defeat it would be to develop an equally valid alternative chain. This may be where the scheme is most at risk:

An attacker may seek to construct their own, valid, alternative chain that does not contain evidence of the user's transactions. It may take time, but if they succeed, then an attacker could construct a witness that can decrypt the secret.

Of course, the main threat would wish to keep the secret protected - but this does alter the ability of the switch to control the release time, and early release also has negative connotations for the subject.

A potential mitigation for this risk:

> Further constrain the witness such that it must also include positive evidence of activity from any other chosen identity, predicted to reliably transact over the given period. The signature of this (potentially even unwitting) identity will be impossible for an attacker to fake.

8.3.4.5 Affordability This solution is extremely affordable, as it is largely passive. A small piece of software could be constructed to encrypt the secret. The subject must then perform some activity on the blockchain to show that they are alive. As this can be a small transaction from one wallet they own to another they own, it is likely to be very inexpensive.

8.3.4.6 Durability This solution depends on continued activity on the blockchain of the cryptocurrency of the subject's choice. If this activity changes, or the cryptocurrency loses popularity, it may be necessary to rebuild the ciphertext for another. As the subject will be making regular transactions as a proof of life, they are likely to be aware if this is the case.

8.3.4.7 Explainability This solution relies on some complex mathematics. It may be possible to explain some of the assurances that it provides in simpler terms, but it is unlikely that many of the target subject will want to verify it for themselves. They may rely on trusted academics and peer review to determine that the scheme is resilient.

8.3.4.8 Visibility Under this scheme, the ciphertext can be safely published - and a subject is free to publish information about where it can be found, and how it can be decrypted. It may be prudent to encrypt a second sample with a scheme that allows it to decrypt sooner (more like a time-lock encryption) to demonstrate that the secret is real.

8.3.5 Further considerations

This design is hypothetical, and relies on the assumption that a witness encryption scheme can be constructed that is capable of evaluating activity on a blockchain. It has yet to be shown that this constitutes a valid mapping in an NP language. Although it is not yet developed, it may form a fruitful avenue for further research.

8.4 Scoring

Each solution's evaluation against the requirements for a DPS[53] is scored with the same criteria used to evaluate existing systems, reproduced here:

Table 48: Scoring criteria for evaluated DPS solutions, reproduced

Score	Description
0	Does not meet requirement, or no information available.
1	Shows awareness of the requirement, minimal steps towards a solution, with significant room for improvement.
2	Partially meets the requirement, with some room for improvement.
3	Meets the requirement comprehensively, with little or no room for improvement.

This scoring matrix is reproduced as figure 24.

The proposed designs each offer a different approach to the problem.

The hosted micro-services architecture relies on the security properties of its hosting provider. Whilst the major providers are considered to have strong protections including excellent staff policies, and physical protections for their data centres, they may be vulnerable to legislation and law enforcement in the countries that the data is hosted in. This limits its *confidentiality* and *resilience* properties.

It also needs additional work to ensure that it is highly available. Additional protections could help to prevent denial of service attacks:

- DoS protections from a service such as Cloud Flare.
- Load balancing to help with periods of high load or smaller attacks.
- Proxies and a reverse-proxy to help conceal its location.

[53] Requirements are defined in section 5.2.

Type	Solution	Confidentiality	Awareness	Timing	Resilience	Affordability	Durability	Explainability	Visibility
Hosted	DeadMansSwitch	1	2	2	1	3	1	1	0
Hosted	DeadMan	1	2	3	1	3	0	1	0
Hosted	DeadManTracker	2	2	3	2	3	3	2	0
Hosted	Letters Cloud	2	2	2	2	3	0	1	0
dApp	KillCord	2	3	3	2	3	3	2	2
dApp	Kimono	3	0	3	2	3	3	2	2
dApp	SilentDelivery	3	0	3	3	3	3	2	2
OSS	skickar/DeadManSwitch	1	1	2	1	3	1	2	0
OSS	h313/dead-mans-switch	0	1	2	1	3	1	2	0
dApp	deadmenswitch/dms	1	3	2	1	3	3	2	1
OSS	EsmailELBoBDev2/Dead-man-s-switch	0	1	2	0	3	1	2	0
OSS	dmp1ce/DMSS	2	0	2	2	3	2	2	0
Proposed	**#1 Microservices architecture design**	2	2	3	2	3	3	3	0
Proposed	**#2 Distributed application**	3	3	3	3	3	3	2	3
Proposed	**#3 Application of witness encryption**	3	3	3	3	3	3	1	3

Figure 24: Scoring matrix for proposed and existing DPS solutions

The micro-services design is weakest where trust is concerned. Users must decide to trust the operator, and so such a service will need to go to lengths to show that it will not leak secrets, and has a good grip on internal threats as well as external. A privacy policy is a good first step. It does, however, have an advantage over the others for *explainability* - it is simple to describe, and may follow a less-technical user's assumptions about how such a system should work.

The dApp design benefits from the strong availability properties of smart contracts, and the confidentiality provided by secret contracts. As it spans 2 networks, however, and relies on complex concepts to assure its properties, it may be difficult to explain to target subjects, suffering in *explainability*.

Although considered affordable, the cost of using such a dApp may change with time, as cryptocurrencies fluctuate in value.

The witness encryption scheme benefits from the appeal of being an almost entirely passive solution. Once the secret is encrypted, it will not decrypt until the subject's proof of life is absent from the blockchain. It almost seems magical!

As this scheme depends on complex (and currently undeveloped) mathematics, it may also prove unsuitable for the target subjects unless they can accept advice about its safety from trusted academics. For that reason it has been awarded a low score for *explainability*.

As discussed, the proposed witness encryption scheme may be susceptible to a false chain that does not contain the subject's proof of life. Careful thought must be given to the definition of a valid witness to mitigate this risk.

8.4.1 Conclusion

The strongest scoring solution presented is the distributed application, using an Ethereum smart contract and a SECRET secret contract. It meets most of the requirements comprehensively.

A lot can be learned from building, iterating on designs, and user acceptance testing. These designs make good candidates for future work.

9 Evaluation

This thesis set out to contribute one or more designs to tackle the requirements of a Dead Person Switch. In early research it became apparent that very little formal work exists in this field.

In order to deliver the designs, the work was divided into several stages, each of which is discussed in this section.

Table 49: Implementation stages for evaluation

Stage	Task
1	Development of requirements
2	Assessment of existing solutions and related components
3	Proposal and evaluation of designs

The limitations found in this project lead to a number of recommendations for future work, discussed in section 10.1.

9.1 Development of requirements

Research to support the development of requirements proposed in section 5.2 was conducted through several means:

- Desk research
- Study
- Survey

Desk research was limited by available material.

Very little formal work on Dead Person Switches is available. In the absence of this, other sources (blog posts, news articles) provide a view of high profile cases where a DPS or similar tool is used. A number of information-rich articles mentioned in section 3 also contribute to the picture, and broadened research into time-lapse cryptography techniques, too.

The researcher is aware of ongoing work to develop a taxonomy of threats to journalists. Unfortunately, this was not published (or available for review) at time of writing.

Although not directly related, time-lapse cryptography provides a number of more formally explored techniques, some of which could then be adapted. This line of research led to the inclusion of witness encryption as a proposed solution.

Learning about user needs is key to gathering requirements. The survey conducted was designed to learn about journalists, their experience of risks to personal safety, how they would reason about the uses of a DPS, the impact of such a system failing, and how they would decide to trust such a service.

With limited resources and reach, the survey was not targeted at whistleblowers, but rather the journalists who work with them. This addresses some, but not all, of the possible aspects of an investigative journalism / whistleblower use case.

Despite efforts to connect with multiple journalism support communities[54], the survey responses were limited. As shown in figure 4, only 5 of 73 participants completed it - providing some indirect evidence of the difficulty developing trust with investigative journalists.

With only 5 respondents, the survey results are not a comprehensive picture of the experience of investigative journalism - and this limits the gravitas and rigour that can be applied to any inferences drawn from the exercise.

Participants that did respond indicated that they did not believe they would use a DPS (one stating that it would be too expensive to operate), couldn't trust the operator of such a service, and didn't

[54]Particular thanks to those in the Bureau Local community who helped broaden the reach of the survey, and colleagues who vouched for the intentions of the research.

believe that a secret could have a strong deterrent effect. Initially disheartening, these outcomes led to the incorporation of additional requirements relating to cost, trust and deterrence[55].

A further requirement, perhaps missing, is the ability to retire a secret if the threat to a subject is no longer considered dangerous[56]. This could impact several of the proposed and evaluated solutions as, for instance, smart contracts cannot be altered once committed to a blockchain. This means that if they do not include a cancellation mechanism, the subject risks committing to suppressing release of their secret indefinitely.

9.2 Assessment of existing solutions and related components

The derived explainability requirement highlights a strong weakness found in all existing solutions: Users must find a reason to trust a service, and none so far have made comprehensive efforts to describe their processes. In fact, only one provides a privacy policy.

The absence of comprehensive documentation for closed source solutions makes it much harder to review and comment on the qualities of these systems. Research was limited to an understanding of existing systems based on the material they do provide. Attempts to enquire about design (queries sent to all 4 hosted solutions) provided a little more information about Dead Man Tracker, and Letters Cloud - but in each case, the architecture and design was not available. This paints an incomplete picture of the landscape, although in several cases there was enough information to give the researcher confidence in their commentary.

With unlimited resources, it might have been possible to further incentivise information sharing for this research:

- Remuneration could be offered to hosted solutions willing to share their architecture and threat models.
- Reluctance to communicate could be reduced with the offer of an NDA, and assurances from the University of Oxford. (If necessary, ethical approval could be sought for this part of the research, too, and the protections put in place shared with these services to help them decide about sharing their information.)

In order to assess the capabilities of the hosted services, in addition to reading material published on their websites, the researcher created accounts and tested them. In some cases, this was possible - although in others, the services were no longer working. This generated insights into durability. Where possible, services were assessed against the requirements defined in section 5.2, which led to the creation of comparable reports.

Further colour and insight could be obtained by inviting subjects (professional investigative journalists) to try these services and share their opinions, too.

Each existing solution was assessed against the *affordability* requirement defined in section 5.2. However, it has not been rigorously defined. Further information about expected salaries, and expectations of the cost of essential tooling learned from further survey of journalists could help to clarify this measure.

9.3 Proposal and evaluation of designs

As discussed above, research in the space of Dead Person Switches isn't far enough developed to have given clear direction for a single implementation. Instead, this thesis delivers 3 designs, each tackling a different approach.

Limited resources and a word-limit constrained this thesis, and led to a careful prioritisation of work. Only the first (micro-service architecture) proposed solution is complemented by a risk analysis, as this is a time consuming, and high volume, output. By providing a risk analysis for the "classic" solution, this thesis provides a point of reference when reasoning about the others.

Further, the other solutions proposed do not lend themselves easily to traditional risk analysis: Both have a strong reliance on cryptocurrency networks, which are decentralised, complex systems[57] that quickly

[55] Requirements: Affordability, Explainability, Visibility

[56] Perhaps an investigation has concluded, or perhaps they have turned over a new leaf.

[57] The researcher does not shy away from complex work, but modelling something this vast is beyond the scope of an MSc thesis, and a distraction from the techniques being explored. It would significantly curtail the other work in order to meet

complicate risk analysis. (This is discussed in section 4.5). Instead, a pragmatic approach was adopted: Analyse by evaluation against the requirements, and compare and contrast the 3 proposed solutions.

The designs themselves were selected to represent different, and promising, approaches to the solution. These 3 designs are not believed to represent a comprehensive collection of all *possible* solutions. Further resources, time, and research could lead to the proposal and assessment of additional designs exploring the breadth of related components.

9.4 Breadth vs. depth

At several points in this thesis, choices were made regarding the scope - and that has resulted in a broad study of solutions, components and related research. This, combined with the constraints of an MSc thesis, limited the depth to which aspects could be explored.

Some supporting material is available in the appendices, but ultimately some research that did not lead to design proposals had to be shortened and summarised, or removed, to meet the constraints.

This is particularly true of desk research into trust networks and subjective logic, which begins to address relevant questions around trust and reliability. Ultimately it was determined to be impractical in its current state, and inapplicable to the proposed solutions. It has been summarised in section 7.1.

9.5 Ethical risk

This thesis chooses to focus on a use case that is considered ethical. However, even the threat model presented in section 5.4.2 acknowledges that there are some activists who may disagree.

Of course any system may be used for any purpose, but the act of building a system that could be used for malicious activity is fraught with risk. It should be explored further before production systems are built.

---

the constraints of the format.

10 Conclusions

10.1 Recommendations for future work

10.1.1 Collaborative research

The survey was limited in scope. Subsequent work to learn from the investigative journalism community could be broadened by conducting research in partnership with journalism outlets, unions and communities[58].

Doing so is an opportunity to learn more from potential subjects, including questions that might reveal insights regarding the potential additional requirement mentioned in section 9: the ability to cancel a DPS.

10.1.2 Alignment with research

The risk model developed for this thesis should be reviewed in the light of ongoing research into journalism and safety. The taxonomy of threats to journalists, as yet unpublished, will positively impact the quality of research into this field - providing a common language for conversation about these risks.

10.1.3 Alternative design proposals

As discussed in section 9, the designs proposed are not considered to be comprehensive, and a number of possible alternative designs could also contribute to the understanding of this field. Proposed further research might include (but is not limited to) some of the following:

- A system operated entirely by people (building on work exploring trust networks, game theory, and remuneration effects to manage the honesty of large groups of strangers).
- A system with physical components (assessing physical protections for data, device hardening measures, and the utility of obfuscation of physical location, information, etc).
- A system that relies on strong biometric signals to measure aliveness and well-being (developing ideas about embedded bionic hardware, or relying on other systems that do this - such as a mobile phone device unlock).
- A system that builds on proof of life forensics to determine the aliveness of a person, using a wide variety of signals generated from the footprint of daily activity.

10.1.3.1 Trust networks The human element in this field is a particularly wide area for further research.

Research into trust networks is still in its infancy. It was dismissed in this thesis as impractical for target subjects to apply, but it is only through steady research that working solutions can be developed. Methods for measuring and estimating the behaviour of participants in a trust network could bring this field closer to practical applications. A cross-disciplinary approach to understanding human behaviour in these scenarios (perhaps through an exploration of game theory, psychology, and subjective logic) might help.

10.1.3.2 Human judgement Developing a method for measuring humans' ability to determine the aliveness of others (and just how much effort is required to deceive them) will help to determine just how well trust networks are suited to inclusion in a DPS.

Calculating the $F1$ scores for human judges could provide a scale that steadily changes over time - as the quality of deceptions improves.

With sufficient data, Turing Tests could form the basis of an $F1$ calculation for the accuracy of humans as judges.

10.1.3.3 Proof of life forensics The application of techniques in proof of life forensics (used in investigation contexts to determine if and when a person has died) could also prove valuable to this field.

[58] A researcher *without* a public profile indicating their background in voluntary policing may also gain more traction with investigative journalists.

10.1.4 Visibility

The visibility requirement links evidence suggesting that a DPS is working (and contains a valuable secret) to a deterrent effect.

Proof of knowledge is seen in other areas of research, too - including zero knowledge proofs, and other forms of verification.

In secretive services, such as dark markets, vendors selling stolen user accounts will often provide a small sample to show that they "have the goods." Research into how effective this is as a sales tactic could help to establish which proofs can convince a threat that a DPS holds valuable information.

10.1.5 Ethics and abuse

An exploration of *abuse cases* may help to develop a stronger understanding of which properties make a service attractive to users who wish to conduct blackmail, as compared to users with more wholesome pursuits such as whistleblowing and journalism. This could lead to design recommendations that nudge users towards certain desired behaviours, or discourage others.

10.1.6 Extended scope

Complex scenarios were scoped out of this thesis - including some possibilities that have the potential to quickly multiply the complexity of any solution.

- How do the incentives change when there is more than 1 threat, or more than 1 secret to store in a DPS?
- If a subject owns two switches, one with a secret about each potential threat, and is attacked - how should each switch determine whether it should activate?
- If a user has a single switch containing multiple secrets pertaining to multiple threats, how might they be expected to behave if they do or do not know about each other?
  - Does the risk to the subject automatically increase with the number of threats?
  - Can threats be persuaded to change their behaviour (to protect the subject, even) if they perceive that another threat may succeed in killing the subject?
  - What behaviours can be ascribed to threats who also have reason to sabotage each other?
- What happens when one participant in a trust network is trying to undermine the whole group?

These questions could form the foundation for a number of cross-disciplinary research projects.

10.1.7 Build and test

A lot can be learned from building and testing systems, and work to develop one or more of the proposed designs could be extremely valuable. User acceptance testing, in this case, could inform work to discover how best to achieve the explainability requirement - and how users might develop trust in a system.

The development of a witness encryption scheme is likely to require some intensive research, which might conclude that it is impossible. This is worth establishing. It has the potential to guide the creation of a passive system with strong properties, meeting the requirements of a DPS.

Developing these solutions in the public domain as open source projects will allow security researchers to evaluate and attack them - providing valuable insights into their strengths and weaknesses.

10.2 Final summary

This thesis has presented research into the background and uses for a Dead Person Switch, developed a set of requirements, and used those to evaluate existing systems.

It has shown that although some existing systems come close, none are really capable of meeting the needs expressed in the investigative journalism use case.

It has also contributed three evaluated designs, each based on a different approach to the problem: a micro-services architecture, a dApp, and an application of witness encryption. These designs are shown to meet many more of the established requirements for a DPS than the existing available systems.

The research conducted, designs proposed, and their evaluation, leads to a number of recommendations for future work, outlined above.

Bibliography

1. Josang A, Hayward R, Pope S. Trust network analysis with subjective logic. *Conference Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*. Australian Computer Society, 2006, 85–94.

2. Jøsang A. *Subjective Logic*. Springer, 2016.

3. Zhang F, Daian P, Bentov I *et al.* Paralysis proofs: Secure dynamic access structures for cryptocurrency custody and more. *AFT 2019 - Proceedings of the 1st ACM Conference on Advances in Financial Technologies* 2019:1–5.

4. Martin A. The ten-page introduction to trusted computing. 2008.

5. Abelson H, Anderson R, Bellovin SM *et al.* Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity* 2015;**1**:69–79.

6. Nemec M, Sys M, Svenda P *et al.* The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. *24th ACM Conference on Computer and Communications Security (CCS'2017)*. ACM, 2017, 1631–48.

7. ROCA: Infineon TPM and secure element RSA vulnerability guidance. 2017. https://www.ncsc.gov.uk/guidance/roca-infineon-tpm-and-secure-element-rsa-vulnerability-guidance

8. Costan V, Devadas S. Intel sgx explained. *IACR Cryptol ePrint Arch* 2016;**2016**:1–18.

9. Popescu A *et al.* A critical analysis of whistleblower protection in the european union. *Journal of Public Administration, Finance and Law* 2015:135–40.

10. The WikiLeaks insurance files tweet. 2016. https://twitter.com/wikileaks/status/743824112376766465

11. Snowden's contingency: 'Dead man's switch' borrows from cold war, WikiLeaks. 2013. https://www.wired.com/2013/07/snowden-dead-mans-switch/

12. Tuerk A. Plan your digital afterlife with inactive account manager. 2013. https://publicpolicy.googleblog.com/2013/04/plan-your-digital-afterlife-with.html

13. Maintaining ownership continuity of your user account's repositories. https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/maintaining-ownership-continuity-of-your-user-accounts-repositories

14. More than half of parents don't have a will. 2018. https://www.which.co.uk/news/2018/12/half-of-adults-dont-have-wills-but-what-happens-to-your-children-when-you-die/

15. Fifteen men on a dead man's switch. 2018. https://blog.lopp.net/fifteen-men-on-a-dead-man-s-switch/

16. Hackers solve password mystery. 2002. https://chnm.gmu.edu/digitalhistory/links/cached/preserving/8_2_password.htm

17. Shamir A. How to share a secret. *Communications of the ACM* 1979;**22**:612–3.

18. GitHub deceased user policy. https://docs.github.com/en/github/site-policy/github-deceased-user-policy

19. How to prove and verify someone's identity. 2021. https://www.gov.uk/government/publications/identity-proofing-and-verification-of-an-individual/how-to-prove-and-verify-someones-identity#validity

20. Complete guide to GDPR compliance. 2018. https://gdpr.eu/

21. Goldwasser S, Kalai YT, Popa RA *et al.* How to run turing machines on encrypted data. *Annual Cryptology Conference*. Springer, 2013, 536–53.

22. National Statistics O for. Property crime tables. 2020. https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/datasets/focusonpropertycrimeappendixtables

23. A journalist's resource for safe and ethical reporting, chapter 4: Digital safety. 2021. https://training.rsf.org/chapter-4-digital-safety/

24. Orcutt M. Once hailed as unhackable, blockchains are now getting hacked. 2019. https://www.technologyreview.com/2019/02/19/239592/once-hailed-as-unhackable-blockchains-are-now-getting-hacked/

25. Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (sok). *International Conference on Principles of Security and Trust*. Springer, 2017, 164–86.

26. Sutherland A. Tell no tales? Decentralizing a dead man's switch. 2019. https://blog.enigma.co/tell-no-tales-decentralizing-a-dead-mans-switch-6217e2f4361b

27. Branwen G. Time-lock encryption. 2019. https://www.gwern.net/Self-decrypting-files

28. Shostack A. Experiences threat modeling at microsoft. *MODSEC@ MoDELS* 2008;**2008**.

29. Shostack A. STRIDE chart. 2007. https://www.microsoft.com/security/blog/2007/09/11/stride-chart/

30. Goodwin M. Threat dragon lightning demo. 2020. https://www.youtube.com/watch?v=n6JGcZGFq5o

31. Jamal khashoggi: All you need to know about saudi journalist's death. 2021. https://www.bbc.co.uk/news/world-europe-45812399

32. Anderson T. Researchers spot thousands of android apps leaking user data through misconfigured firebase databases. 2020. https://www.theregister.com/2020/05/12/report_thousands_of_android_apps/

33. Celebi GFM, Fletcher-Hill P, Que D. Kimono: Trustless secret sharing using time-locks on ethereum. 2018. https://medium.com/@pfh/kimono-trustless-secret-sharing-using-time-locks-on-ethereum-8e7e696494d

34. AR6 climate change 2021. 2021. https://www.ipcc.ch/report/ar6/wg1/

35. Beautement A, Sasse MA, Wonham M. The compliance budget: Managing security behaviour in organisations. *Proceedings of the 2008 New Security Paradigms Workshop*. 2008, 47–58.

36. Munroe R. Security. https://xkcd.com/538/

37. Shieber SM. Lessons from a restricted turing test. *arXiv preprint cmp-lg/9404002* 1994.

38. Shah H, Warwick K. Hidden interlocutor misidentification in practical turing tests. *Minds and machines* 2010;**20**:441–54.

39. Greene D. New software can mimic anyone's voice. 2017. https://www.npr.org/2017/05/05/527013820/new-software-can-mimic-anyones-voice

40. Leviathan Y. Google duplex: An AI system for accomplishing real-world tasks over the phone. 2018. https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html

41. Aliev A. Fake elon musk joined the zoom call. 2020. https://www.youtube.com/watch?v=lONuXGNqLO0&ab_channel=AliAliev

42. Aliev A. Avatarify. 2020. https://github.com/alievk/avatarify-python

43. Life event verification (LEV) API reference. https://docs.api.lev.homeoffice.gov.uk/life-event-verification-lev-api/reference

44. Research your family history using the general register office. https://www.gov.uk/research-family-history

45. Art. 4 GDPR, definitions. 2018. https://gdpr.eu/article-4-definitions/

46. Grawrock D. Trusted computing, lecture. 2019.

47. Xin X. Titan m makes pixel 3 our most secure phone yet. 2018. https://blog.google/products/pixel/titan-m-makes-pixel-3-our-most-secure-phone-yet/

48. Dismantling of an encrypted network sends shockwaves through organised crime groups across europe. 2020. https://www.europol.europa.eu/newsroom/news/dismantling-of-encrypted-network-sends-shockwaves-through-organised-crime-groups-across-europe

49. Together we are powerful - folding@home. https://foldingathome.org/

50. Bitcoin consumes 'more electricity than argentina'. 2021. https://www.bbc.co.uk/news/technology-56012952

51. Criddle C. Cambridge bitcoin electricity consumption index. https://cbeci.org/

52. Leising M. Bye-bye, miners! How ethereum's big change will work. 2021. https://www.bloombergquint.com/quicktakes/bye-bye-miners-how-ethereum-s-big-change-will-work-quicktake

53. McKie R, Thorpe V. Digital domesday book lasts 15 years not 1000. 2002. https://www.theguardian.com/uk/2002/mar/03/research.elearning

54. Merger of national policing systems over budget and behind schedule. 2020. https://www.computerweekly.com/news/252492695/Merger-of-national-policing-systems-over-budget-and-behind-schedule

55. List of oldest institutions in continuous operation. https://en.wikipedia.org/wiki/List_of_oldest_institutions_in_continuous_operation

56. The streisand effect. https://en.wikipedia.org/wiki/Streisand_effect

57. The AACS encryption key controversy. https://en.wikipedia.org/wiki/AACS_encryption_key_controversy

58. List of material published by WikiLeaks. https://en.wikipedia.org/wiki/List_of_material_published_by_WikiLeaks

59. Secure design principles. https://www.ncsc.gov.uk/collection/cyber-security-design-principles

60. Password policy: Updating your approach. https://www.ncsc.gov.uk/collection/passwords/updating-your-approach

61. Cappelli DM, Moore AP, Trzeciak RF. *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.

62. Zyskind G, Nathan O, Pentland A. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:150603471* 2015.

63. Woetzel C. Secret network: A privacy-preserving secret contract & decentralized application platform. https://scrt.network/graypaper

64. Update on windows 11 minimum system requirements, windows insider blog. 2021. https://blogs.windows.com/windows-insider/2021/06/28/update-on-windows-11-minimum-system-requirements/

65. IPFS ecosystem directory. https://ecosystem.ipfs.io/

66. InterPlanetary file system. https://en.wikipedia.org/wiki/InterPlanetary_File_System

67. Garg S, Gentry C, Sahai A *et al.* Witness encryption and its applications. *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. 2013, 467–76.

68. Liu J, Jager T, Kakvi SA *et al.* How to build time-lock encryption. *Designs, Codes and Cryptography* 2018;**86**:2549–86.

Appendix A: Survey participant documents

The following documents are included in this appendix on the following pages:

- Participant information sheet
- Survey schedule of questions

Participant Information Sheet

Dead Man's Switches - with and without trust

Version	1.0
Date	2021-04-08

Who is conducting this research?

This research is being conducted as part of an MSc project in Software & Systems Security by Lewis Westbury, a graduate student at Kellogg College, Oxford; under the supervision of Dr Christopher Hargreaves.

- Lewis.Westbury@kellogg.ox.ac.uk
- Christopher.Hargreaves@cs.ox.ac.uk

Why is this research being conducted?

The specific definition of Dead Man's Switch (DMS) in scope for this project describes a system for managing the release of a piece of secret information under some very specific circumstances. A whistle-blower or investigative journalist (leaker), may grant control of the release of a specific secret to a DMS. The DMS will then attempt to monitor the wellbeing of the leaker - and if certain conditions are not met (ie. if they are not alive and well) it will then release the secret. Provided the leaker chooses a secret sufficiently damaging to their expected attacker, this can be used to disincentivise a physical attack on the leaker.

The goal of the interview is to learn about use-cases where a DMS might have a positive impact on the safety of an investigative journalist or whistle-blower.

Why have I been invited to take part?

As an investigative journalist with knowledge/experience of the needs of whistle-blowers, you are able to provide key insights into the use cases for a DMS.

Do I have to take part?

No. Taking part is entirely voluntary. You can ask questions about the research before deciding whether or not to take part. If you do agree to take part, you may withdraw yourself from the study at any time, without giving a reason by advising us of this decision.

You may ask to skip any questions you would rather not answer.

If you decide to withdraw from the research, any information you have provided will be safely deleted.

What will happen to me if I take part in the research?

You will first be offered an online consent questionnaire, to ensure that you understand what this study is about, and that you wish to take part.

After that is completed, if you have consented to take part in the research, you will then be interviewed about your experience as an investigative journalist - this will take place by phone call or video call, or you may choose to fill out an online survey for yourself. If you choose to be interviewed by phone call or video call, the researcher will fill out the online survey on your behalf. The interview is expected to take approximately 30 minutes. (You can ask to stop or pause the interview at any time.)

Are there any benefits or risks to taking part?

There are no direct benefits to your taking part, but you will have access to a copy of the final dissertation if this would be of interest to you.

The main risks to your taking part would be that you or your sources might be identified in some way, or that we might lose the personal data that you gave us. We are using an online questionnaire service that is GDPR compliant, ISO 27001 certified, with strong protections for your data. We will seek to anonymise data wherever possible, and only to store it with the questionnaire service selected - which will also offer good protections for your data including access control, encrypted storage, and safe deletion. We will not record names, even if mentioned in the survey. You may ask us to redact any part of your answers, for any reason. Please ask if you would like to know more about how we will do this.

What happens to the data I provide?

The data you provide is divided into two parts:

- Your consent form - this contains contact information for you, and confirmation that you consent to take part in the research.
- Your interview form - this contains the answers that you gave in the interview.

Any data you provide which can be used to identify you is known as **personal data.**

Only your consent form is considered to contain personal data. However, both forms will be stored with the approved, secure survey service - JISC online surveys. They will not be downloaded to any other devices. Instead, they will be reviewed through that service only.

The information we are collecting is qualitative. Your interview will be summarised alongside others and used to determine a set of use-cases for a Dead Man's Switch. These insights will be included in the final dissertation, and used to inform the rest of the project.

The final dissertation will be submitted to the university for grading, and will also be made available online. We will take all possible care to ensure that no names or personal details will be included in the final report.

What information will you collect?

- Some contact information about you, and your consent to take part in the research will be stored in a JISC online survey - the consent questionnaire.
- Your interview answers will be stored in another JISC online survey.
- If interviewed in person, your answers will be transcribed into a JISC online survey by the researcher.
- You may read the interview questions ahead of the interview, or to help you make your decision to participate or not. They are to be distributed with this information sheet. If you have not received them, please let us know.

How will you maintain confidentiality?

- Your records will be stored in an online, GDPR-compliant facility (JISC online surveys).
- Access control will be restricted to:
    - The researcher who conducted your interview.
    - The research supervisor.

Responsible members of the University of Oxford may also be given access to the data for monitoring and/or audit of the study to ensure we are complying with guidelines, or as otherwise required by law.

Your record will be pseudo-anonymised:

- Your interview record will not contain your name. Instead it will be assigned a unique id.
- That unique id will also be linked to your consent form (which contains your name).
- No names (yours, or any sources you mention) will be recorded in your interview answers.
- If you accidentally include identifiable information in your answers, about yourself or any other person, we will retrospectively remove that information.
- You may also ask us to redact any part of your answers if you feel they could be used to identify you or another person, even without names.

How long will you hold my information for?

- Your interview answers will be stored until the project has been completed.
- Your consent information will be stored for 3 years from the point of publication of the dissertation.

Will the research be published?

The research may be published online - through social media or blog posts.

The University of Oxford is committed to the dissemination of its research for the benefit of society and the economy and, in support of this commitment, has established an online archive of research materials. This archive includes digital copies of student theses successfully submitted as part of a University of Oxford postgraduate degree programme. Holding the archive online gives easy access for researchers to the full text of freely available theses, thereby increasing the likely impact and use of that research.

The research will be written up as a student's dissertation.

On successful submission of the dissertation, it may be deposited both in print and online in the University archives to facilitate its use in future research. If so, the thesis will be openly accessible.

Who do I contact if I have a concern about the study or I wish to complain?

If you have a question or concern about any aspect of this project, please speak to Lewis Westbury, lewis.westbury@kellogg.ox.ac.uk (researcher), or Dr Christopher Hargreaves christopher.hargreaves@cs.ox.ac.uk (supervisor), who will do their best to answer your query.

The researcher should acknowledge your concern within 10 working days, and give you an indication of how they intend to deal with it.

If you remain unhappy or wish to make a formal complaint, please contact Professor Andrew Martin, Chair, Computer Science Departmental Research Ethics Committee, Wolfson Building, Parks Road, Oxford OX1 3QR or ethics@cs.ox.ac.uk

The chair will seek to resolve the matter in a reasonably expeditious manner.

Data protection

The University of Oxford is the data controller with respect to your personal data, and as such will determine how your personal data is used in the study. The University will process your personal data for the purpose of the research outlined above. Research is a task that we perform in the public interest. Further information about your rights with respect to your personal data is available from: https://compliance.admin.ox.ac.uk/individual-rights

Who has reviewed this study?

This study has been reviewed by, and received ethics clearance through the Computer Science Departmental Research Ethics Committee at the University of Oxford.

They can be contacted at: ethics@cs.ox.ac.uk
Please quote approval number: **CS_C1A_21_011**

Interview Schedule

Dead Man's Switches - with or without trust

Version	1.0
Date	2021-04-08

Introduction

This is a preview of the interview questions that will be asked as a part of the study described in the accompanying material. If you have any questions, please feel free to ask them before taking part.

This study has been reviewed by, and received ethics clearance through the Computer Science Departmental Research Ethics Committee at the University of Oxford.
Approval number: **CS_C1A_21_011**

The **Participant Information Sheet** that should accompany this interview schedule contains contact details for the researcher, supervisor, and ethics approval contact should you have any questions or complaints.

Interview questions

A reminder: You may ask to skip over any question you do not wish to answer.

Please take care not to share personal anecdotes or personal history that could be used to identify you. You may ask to have any information removed from your answers at any time.

What did the term Dead Man's Switch mean to you before you encountered the materials for this research?

For the purposes of this interview, a Dead Man's Switch can be considered to mean any system where a secret can be stored (not limited to digital systems, and including with colleagues, friends, or strangers) for release on the failure of an aliveness check.

We'll refer to Dead Man's Switches as DMS going forwards.

Have you ever been unable to complete a piece of work because of threat of reprisal, or substantiated reprisals?

Have you worked with any whistle-blowers or sources where there were reasons to fear reprisals should their identities be uncovered?

Have you ever had reason to suspect that you were personally at risk of reprisal from subjects of the investigative aspects of your work?

In your professional opinion, can secrets obtained through whistle-blowing or other investigative means serve as a deterrent against reprisals?

Are there any DMS systems that you consider sufficiently safe for use professionally?

Have you ever employed a DMS or similar system?

- **(If no.) Do you consider that you might if the need arose?**
- **(If yes.) Which system did you use?**
- **(If yes.) Was it required to serve as a deterrent against reprisals?**
  - **(If yes.) Do you consider that successfully prevented those reprisals?**
    - **(If no.) What would you have changed about it to meet your needs?**

What assurances would you accept that a DMS system is resistant to the following:

- **Physical or digital assaults to make it unavailable?**
- **Physical or digital assaults to retrieve its stored secrets or alter its behaviour?**
- **Bribery to retrieve its secrets or alter its behaviour?**

How would you decide whether to trust the operator of a DMS?

What are the consequences of a DMS releasing a secret before it was required to?

What are the consequences of a DMS releasing a secret after it was required to?

What are the consequences of losing confidentiality of a secret stored in a DMS?

What are the consequences of losing the ability to access or communicate with a DMS?

What are the consequences of the secret stored in a DMS being altered?

Do you have any other thoughts or comments that you would like to add to help clarify the required properties of a DMS?

Appendix B: Threat model report for a generic DPS

The threat model shown on the following pages was generated in Threat Dragon to represent a generic DPS.

Components included:

Table 1: Components in the generic DPS model

Component	Role
Initialiser	Component that handles the **secret** initially provided by the **subject**. Arranges safe communication and storage of the **secret**, and establishes a method for the DPS to check the well-being of the **subject**.
Secret store	Component that stores the **subject's secret** in such a way that it can be extracted only when the right conditions are met.
Aliveness checker	Component that can monitor **evidence** of the **subject's** well-being, and able to activate or suppress the **secret extractor** dependent on the **subject's** state.
Secret extractor	Component that can retrieve the **secret** from the **secret store**, and share it to the **publishing medium**.
Publishing medium	Either a public medium where the **secret** can be published, or direct communication of the **secret** to specific recipients.

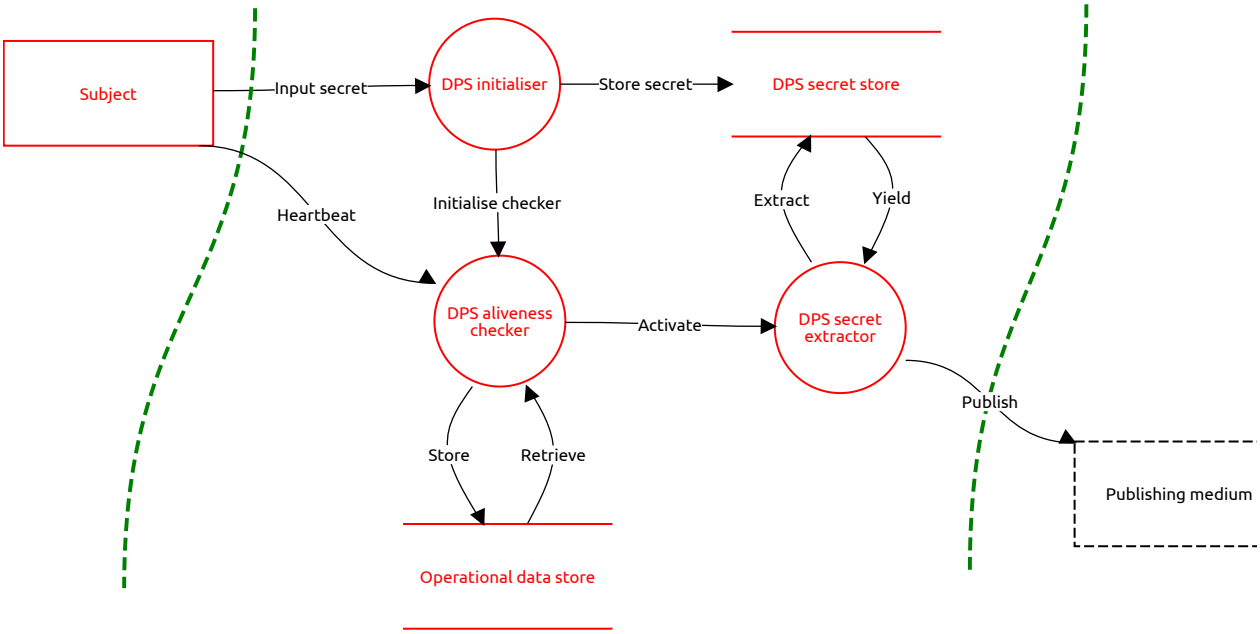
Threat model report for Generalised DPS model

Owner:
Lewis Westbury
Reviewer:
Contributors:

High level system description

Very generalised implementation of a Dead Person Switch, describing the main components and the data flows between them.

High level DPS components



Subject (External Actor)

Description:
Owner of the DPS, and owner of the Secret.

Spoofing identity of the subject

Spoofing, Open, High Severity

Description:
If the subject's identity is spoofed, it may be possible for an attacker to submit false well-being information about them - this could then be used to suppress the activation of the DPS secret extractor, and prevent the switch from releasing the secret in the absence of the real subject.

Mitigation:

DPS initialiser (Process)

Description:
Stores the user’s secret in the store, and establishes a regular aliveness check.

Spoofing the DPS initialiser

Spoofing, Open, High Severity

Description:
An attacker that spoofed the initialiser could prevent the subject from succesfully storing their secret.

Mitigation:

Tampering with DPS initialiser

Tampering, Open, High Severity

Description:
By succesfully tampering with the initialiser, an attacker could cause it to store a modified/empty secret, or fail in any number of ways during DPS creation.

Mitigation:

Information disclosure from the initialiser

Information disclosure, Open, High Severity

Description:
In this design, the initialiser briefly knows and stores the subject's secret. An attacker with access to the console may be able to extract operating information from the intialiser and learn the secret as it is delivered.

Mitigation:

DoS of the DPS initialiser

Denial of service, Open, High Severity

Description:
If the initialiser is made unavailable, the subject will not be able to create a DPS and store their secret. This could be achieved by overwhelming it with requests to initialise new DPS instances.

Mitigation:

DPS secret store (Data Store)

Description:
Protects the user’s secret until required.

Tampering of the secret store

Tampering, Open, High Severity

Description:
If the secret store is tampered, the subject's secret could be erased or (less likely) modified. This prevents the subject's secret from being released when required.

Mitigation:

Information disclosure from the secret store

Information disclosure, Open, High Severity

Description:
If the store leaks information, the subject's secret could be released early.

Mitigation:

DoS of the secret store

Denial of service, Open, High Severity

Description:
If the secret store is made unavailable, it may not be possible to store the subject's secret, or the secret may not be available for extraction at the appropriate time.

Mitigation:

DPS aliveness checker (Process)

Description:
Periodically confirms that the user is alive and well.

Spoofing the aliveness checker

Spoofing, Open, High Severity

Description:
If an attacker succesfully spoofs the aliveness checker service, they could cause the Subject to submit their well-being information to the spoofed service, rather than the real one. The real one will then cause the release of the subject's secret.

Similarly, a fully spoofed aliveness-checker could submit activation or suppression instructions to the DPS secret extractor, causing either early release of the secret or preventing its release.

Mitigation:

Tampering with the aliveness checker

Tampering, Open, High Severity

Description:
If tampered with, the aliveness checker could be used to cause early release or prevent the release of the subject's secret.

Mitigation:

DoS of the aliveness checker

Denial of service, Open, High Severity

Description:
A successful DoS attack against the aliveness checker prevents the subject's well-being information from being processed by the DPS. (This could be achieved by overhelming the checker with requests, either nonsense, or replays of previously accepted submissions.) Without suppresion information from the checker, the secret extractor could then publish the subject's secret early.

Mitigation:

DPS secret extractor (Process)

Description:
Extracts the user’s secret if the user is not determined to be alive and well.

Elevation in the secret extractor

Elevation of privilege, Open, High Severity

Description:
It ought to be very difficult to reach the secret extractor, but an attacker that succeeds should be able to cause the early release of the subject's secret, or prevent it from being release.

Mitigation:

Tampering with the DPS secret extractor

Tampering, Open, High Severity

Description:
If an attacker tampers with the DPS secret extractor they could cause early release of the subject's secret or prevent its release at the right time.

Mitigation:

Input secret (Data Flow)

Description:
The subject creates an instance of a DPS by providing a secret to store.

No threats listed.

Heartbeat (Data Flow)

Description:
The subject provides evidence of their well-beling to the DPS.

No threats listed.

Store secret (Data Flow)

Description:
The DPS initialiser stores the Subject's secret.

No threats listed.

Activate (Data Flow)

Description:
Suppresses the DPS secret extractor if the subject is alive and well. Otherwise, activates it.

No threats listed.

Extract (Data Flow)

Description:
The DPS secret extractor requests the Subject's secret from the DPS secret store.

No threats listed.

Initialise checker (Data Flow)

Description:
The DPS initialiser creates the DPS aliveness checker, with enough information to be able to recognise well-being signals from the Subject.

No threats listed.

Operational data store (Data Store)

Description:
Operational data store contains information required to accept/reject the Subject's well-being information.

Tampering with the operational store

Tampering, Open, High Severity

Description:
Tampering with the store could cause alterations to the subject's information - making it impossible for the DPS to recognise well-being from the real subject, or allowing it to accept false subject well-being information.

Mitigation:

Information disclosure from the operational store

Information disclosure, Open, High Severity

Description:
Information leaked from the operational store could be used to generate false well-being information about the subject. This, if then fed in to the DPS aliveness checker could prevent the release of the subject's secret even in their absence.

Mitigation:

DoS of the operational store

Denial of service, Open, High Severity

Description:
if made unavailable through denial of service, the aliveness checker will fail to function - as it cannot recognise the subject's well-being information. In turn this could cause the early release of the subject's information, as the DPS secret extractor will activate in the absence of a suppressing instruction from the DPS aliveness checker.

Mitigation:

Store (Data Flow)

Description:
The DPS aliveness checker stores initial operational data to help it accept well-being data from the subject.

No threats listed.

Retrieve (Data Flow)

Description:
Data is the operational data store is used to accept or reject well-being data from th S

No threats listed.

Yield (Data Flow)

Description:
The DPS secret store delivers the requested secret to the DPS secret extractor.

No threats listed.

Publish (Data Flow)

Description:
The DPS secret extractor publishes data to the Publishing medium.

No threats listed.

Publishing medium (out of scope External Actor)

Description:
A public place to share the secret, or sharing by communication to specific individuals.

Out of scope reason:
Security issues at the publishing medium, other than availability, are considered out of scope. The publishing medium is more often than not a public resource - and data pushed here should be visible to anyone.

Appendix C - evaluated solutions

This Appendix contains tables showing the DPS solutions evaluated by this thesis.

These solutions are divided into 3 categories:

1. hosted solutions,
2. distributed applications (dApps), and
3. open source solutions.

Each solution is summarised in a table per category (Tables 01 - 03), and all solutions are evaluated against the requirements for DPS proposed by this thesis in comparison and scoring tables (Tables 04 - 05).

List of tables

01	-	Hosted solutions	*Consumer hosted DPS solutions.*
02	-	dApp solutions	*Distributed application DPS solutions.*
03	-	Open source solutions	*Open source DPS projects.*
04	-	Evaluations by requirement	*Solutions evaluated by requirements for a DPS.*
05	-	Scored evaluations	*Solutions scored against requirements, based on the evaluations in Table 04.*
Notes		DPS requirements	*A reproduction of the requirements for a DPS, as defined in the thesis body, are provided at the end of this appendix.*

Appendix C **Table 01 - hosted solutions**

Solution	URL	Privacy policy	Twitter	Last tweet	Contact	Contacted	Status	Year	Summary	Technicals	Use cases	Disclaimers	Notes	More info
DeadMansSwitch	https://www.deadmansswitch.net	N/A	@stochastic	2020-06-13	hi@stochas info@stocha	2021-07-27 2021-08-10	Contacted	2017	A system that checks for signs of life, and if not met, sends a number of emails to chosen recipients.	Not known	* end of life needs (ie. messages for loved ones)	"Dead Man's Switch is provided without any guarantees of anything, not even that it will do its job properly" "this service is meant for casual use by the average person. Please don't use the service if you need strong guarantees of privacy, e.g. if you are a whistleblower or any similar life-and-death situation. It is NOT meant to safeguard against high-value messages."	Mediums: * web push notification, telegram, email, sign in * publish to: email	
DeadMan	http://www.deadman.io/	N/A	@m3ntat	2016-01-13	http://www.c	2021-07-27	Contacted	2012	A system that checks for a response, and distributes documents by email if it does not receive that response.	* Hosted on Google App Engine * Written in Python * Twilio & send grid integrations	* whistleblowers * wilderness excursions (missing persons) * shut-ins * the unknown * end of life needs	"Deadman doesn't provide heath, life, auto, etc. insurance. It provides the kind of insurance that keeping a fire-extinguisher in your house provides or having a spare tire in your car."	Mediums: * email, txt, phone * attach files * using: twilio, sendgrid	https://news.ycombinator.com/item?id=4381905
DeadManTracker	https://www.deadmantracker.com/	https://www.deadmantracker.com/privacy-policy	@DeadManTracker	2021-05-19	support@de	2021-07-27	Responded	2019	A service that automatically contacts your friends and/or family in the event that something happens to you.	* Hosted on AWS * Payment through Stripe / Google / Apple	* missing persons (inc. location tracking)		Mediums: * email, txt, phone * attach files Also a dedicated app, with a discreet mode.	
Letters Cloud	https://letters.cloud/	N/A (mentioned)	N/A	N/A	anomia@pr	2021-07-27	Responded	2020	A service that allows you to create messages to be sent, should you stop visiting your trigger link.	* Compartmentalised design	* any	"This is the place, this is the secret server."	At time of writing, neither komprom.at or letters.cloud was functional: "The system is being upgraded to increase capacity and resiliency, enhance your calm. We'll be back soon!" Mediums: * email or tweet * aliveness checks by URL	
~~**WeCroak**~~	~~https://www.wecroak.com/~~	~~https://www.wecroak.com/privacy-policy~~	~~@WeCroakApp~~	~~2021-07-18~~	~~N/A~~	~~N/A~~	N/A	~~2018~~	~~Advice service.~~	~~N/A~~	~~N/A~~		~~Not a DPS!~~	
~~**Afternote**~~	~~https://www.afternote.com/~~	~~https://www.afternote.com/privacy-and-disclaimer~~	~~@AfternoteTweets~~	~~2017-07-14~~	~~https://www.~~	~~2021-07-27~~	Contacted	~~2017~~	~~End of life planning, granting access to trustees.~~	~~N/A~~	~~* end of life needs (farewell msgs, funeral arr.)~~		~~Not a DPS!~~	
~~**BeRemembered**~~	~~https://beremembered.com/~~	~~https://beremembered.com/more/privacy-policy~~	~~@BeRemDotCom~~	~~2015-09-02~~	~~https://berer~~	~~2021-07-27~~	Contacted	~~2016~~	~~End of life planning, granting access to trustees.~~	~~N/A~~	~~* end of life needs (farewell msgs, funeral arr.)~~		~~Not a DPS!~~	
~~**MyWonderfulLife**~~	~~https://www.mywonderfullife.com/~~	~~https://www.mywonderfullife.com/customer-service-faq~~	~~N/A~~	~~N/A~~	~~N/A~~	~~N/A~~	N/A	~~2009~~	~~Planning service.~~	~~N/A~~	~~* end of life needs (farewell msgs, funeral arr.)~~		~~Not a DPS!~~	

Appendix C	Table 02 - dApp solutions								
Solution	Year	URL	Article	Source	Updated	Description	Technologies and components	Disclaimers	Notes
KillCord	2018	https://killcord.io/		https://github.com/nomasters/killcord	2021-07-16	Killcord is a tool used to build resilient deadman's switches for releasing encrypted payloads. In its default configuration, killcord leverages ethereum and ipfs for censorship resistance. The killcord project owner hides a secret key from the world by checking in to the killcord smart contract on ethereum. If the owner stops checking in after a period of time, the killcord is triggered and the secret key that decrypts an encrypted payload is published.	* **ipfs** for decentralized, immutable, peer-to-peer storage of the encrypted payload * **an ethereum smart contract** for trustless and censorship resistant application state. * **a hidden publisher** written in go that communicates with the ethereum smart contract and publishes the ipfs stored encrypted payload key in the event that the killcord owner stops checking in. * **a client** killcord cli written in go meant to run on a personally controlled system that bootstraps the entire killcord system and allows for checkins.	WARNING This software is in early alpha. Please do not rely on this with your life. Though great care has been taken to ensure that this code is as well structured and straightforward as possible, it has not undergone proper peer-review and could have both minor and major bugs that undermine the integrity of the system.	Use of ethereum and IPFS provides a lot of the integrity and availability required, but as we've seen before - ethereum can't keep secrets, it's a public blockchain and smart contract data and state are agreed by consensus. KillCord's solution is to operate a separate, hidden, publisher - which stores the decryption key, checks the state of the smart contract to determine aliveness, and does the work to decrypt and publish the secret if the user is no longer available.
Kimono	2018	https://kimono.network	https://medium.com/@pfh/kimono-trustless-secret-sharing-using-time-locks-on-ethereum-8e7e696494d	https://github.com/hillstreetlabs/kimono	2018-05-29	Kimono is a protocol that aims to accomplish time-locking without spending significant compute resources, without requiring multiple actions by the creator of the time-lock, and without trusting a single third party with the process. The platform uses a commit and reveal scheme but distributes the secret for each piece of data across a network of peers who share it with the public only once the time-lock is complete.	* relies on **ethereum** blockchain **smart contract** to manage * uses **SSSS** to split and distribute the decryption key amongst N revealers * uses **IPFS** to distribute the secret fragments to the revealers, encrypted for them * uses the **smart contract** to award prize money as incentive for behaving appropriately * uses a **combiner** to assemble the decryption key from K fragments		When the **revealBlock** is added to Ethereum (i.e. when the time-lock is complete), **revealers** race to publish the decrypted version of their secret fragment to the Kimono smart contract. **Trustless** is a strong word. The protocol **incentivizes** revealers to reveal their fragments after the time-lock, and as soon as possible. *kimono.network is apparently discontinued.*
SilentDelivery	2019		https://arxiv.org/pdf/1912.07824.pdf			The protocol maintains shares of the decryption key of the private information of an information sender using a group of mailmen recruited in a blockchain network before the specified future time-frame and restores the information to the information recipient at the required time-frame. SilentDelivery offers a protocol with 2 main advantages over Kimono: silent recruitment, and dual-mode execution.	* managed by an ethereum **smart contract C_agent** * still uses **SSSS** to split and distribute the decryption key amongst N **mailmen** * mailmen register with **C.newMailman()** with Whisper public key + deposit Strawman protocol: * **TIDS.send:** sender creates a new service with **C.newService()** * fragments are sent to mailmen off-chain (Whisper) * **TIDS.pend**: mailmen can **C.reportPremature()** and split a share with the sender * **TIDS.deliver:** all mailmen **C.revealShare()** * the recipient can **C.revealReceipt()** if satisfied		Seems to advance the field by creating a marketplace for participants (revealers, here called mailmen), and a system for 'silent recruitment' - without making clear on a public blockchain who the participants are. This also divides the protocol into lightweight and heavyweight modes. In lightweight mode, the gas consumed remains at O(1) (ie. does not scale with the number of mailmen). Heavyweight mode is only invoked if a participant issues a challenge. **TIDS** = timed information delivery service *Not included in main thesis - this solution improves on Kimono's approach, but isn't a significant enough departure.*

Appendix C **Table 03 - open source solutions**

Project	URL	Created	Last updated	Languages, tech, libs	Project description	Short summary	Initial notes	Confidentiality	Integrity	Availability	Suggested mitigations
skickar/DeadManSwitch	https://github.com/skickar/DeadManSwitch	2019-12-08	2019-12-08	* Python * AES (pyAesCrypt) * Twitter (twint)	A switch to encrypt a file if you don't tweet a keyphrase in a certain time period	A python script that can be used to encrypt a user secret (and delete the original) if a keyphrase is not tweeted from a given account in a certain amount of time. (Possible use could be to conceal damning evidence.) It could also be adapted to decrypt a user's file if the keyphrase is not tweeted - so releasing a secret.	Requires a little hands-on - some lines are commented so the script will initially encrypt the user's secret. On subsequent runs, the commenting needs to be altered to decrypt the secret (if conditions are not met). If the Twitter account has not tweeted at all, it will immediately exit, too (seems to support the initial encryption usage).	* Decryption key is held in RAM, could be leaked or lost. * Deleted file is not zeroed.	* Encrypted secret on filing system is dependent on OS protections, could be manipulated. * State is held in RAM for lengths of time. Could be manipulated.	* Vulnerable to power cut, disconnection. * Could potentially run on a remote server, which could make it more difficult to attack (DoS) until found. However, it would need a publishing capability. * Twitter API could be spoofed or the user's account compromised, to control the release.	* Implement publishing capability to allow the service to run on a remote server. * Check SSL certs to prevent MITM attack on Twitter API.
h313/dead-mans-switch	https://github.com/h313/dead-mans-switch	2017-04-26	2018-06-12	* Python * Web server (flask)	A dead man's switch which will send out a prerecorded message via email or SMS to predetermined people	A python web service that accepts a POST of a predefined password. If this has not been received after a day, it emails a predefined message to a given email address.	Makes no attempt to encrypt the message, password, or target email address.	* Secret and password are both stored in plaintext. Dependent on OS protections to prevent them being read. * Password does not mutate, and is POSTed to the endpoint - this coiuld be intercepted if SSL and cert checking not enabled.	* Secret on filing system is dependent on OS protections, could be manipulated.	* Vulnerable to power cut, disconnection.	* Require SSL and certificate checking. * As the service can run remotely, it might be possible to conceal the purpose of the server that runs it - making it more difficult to DoS.
deadmenswitch/dms NB. This solution is also a dApp.	https://github.com/deadmenswitch/dms	2017-10-22	2017-10-23	* Solidity (smart contract) * Web app (node) * Ethereum blockchain * Truffle (dev & testing environment)	This is a simple project implementing Dead Man's Swtich at Ethereum Blockchain	A web server wraps access to a smart contract (DMSContract). This stores a map of switch configurations.	Nice - uses a simple smart contract running on Ethereum - benefitting from guaranteed execution, and integrity protections from the blockchain.	* Smart contracts can't really hold secrets - their state is known to the EVMs on the nodes that execute them - and a tampered EVM could be persuaded to give up the data held by the contract.	* Good - the blockchain has our back, and won't allow the contract to be altered. The contract itself also checks its map when CreateDMSContract is called, and won't act (ie. won't overwrite) if the sender already has a switch.	* Good - this runs as a smart contract on a distributed network. It's very difficult to attack availability.	* Put visibility keywords against functions (default for functions in Solidity is public). * Function **getDataFromAddress** checks **msg.sender == beneficiary**, but not the expiry. Even if the web app takes care of this, it should be enforced at contract level. * Use **block.timestamp** (alias = **now**) to get the current timestamp. * Seek a better way to control the secret itself. One possibility is to encrypt the secret for a known recipient. This won't prevent early release, but it will mitigate the fact that the data held in the contract isn't secret. * Of course, this shifts responsibility for the decryption key to that person, so also consider resilience, and perhaps a key sharing scheme such as SSSS to distribute fragments that can be used to decrypt the secret amongst a larger group. * You'll still need to think about *their* resilience. * Alternatively, explore secret contracts, eg. using the SECRET network. *This all goes a bit beyond what you might fairly or reasonably expect to get done during a weekend hackathon.*
EsmailELBoBDev2/Dead-man-s-switch	https://github.com/EsmailELBoBDev2/Dead-man-s-switch	2020-04-27	2020-12-10	* Python * Keyring	It's an app that sends emails to all people you select after you die in case you wanted to say last words or there is info like bank accounts or your PC password or anything you wanted to say	A script that accepts a password. It's checked against the OS keyring service, and if correct, the date for release of the user's secret is pushed back by a day. If the user misses a day, the secret is sent by SMTP to a predetermined recipient.	Nice to see OS keyring facility used here. Even if imperfect, this makes it a lot harder to manipulate the release of the secret. However, data.txt contains the control dates for "today" and "tomorrow", so they could easily be manipulated to get a different behaviour at next run; and the secret itself is in plaintext in the code so easy to get hold of.	* The password is stored in the operating system's keyring service - which is a step up from keeping things in plaintext on disk. It's probably not foolproof, but it's better than not. * The secret itself is stored in plaintext in the python.	* The secret is stored in plaintext in the python, dependent on OS protections, could be manipulated.	* data.txt contains the dates which control the application. These could be manipulated to cause late/early release the next time the script is run. * The script needs to be run regularly to operate.	* Adapt the script for remote usage - ie. by implementing a web service. * Use OS keyring to manage encryption/decryption for the secret too.
dmp1ce/DMSS	https://github.com/dmp1ce/DMSS	2016-05-12	2018-03-30	* Haskell	An automated system for securely and reliably sharing secrets to trusted individuals as it becomes necessary. This library is not production-ready and should not be used for purposes where cryptographic failure may endanger anyone's life, liberty or pursuit of happiness.	An application to assist in the SSSS fragmentation of a symmetric key used for encryption / decryption of a secret, the distribution of those fragments to trustees (participants), and tracking the user's aliveness - notifying the trustees if the user is not responsive.	It's reasonably centralised and manages knowledge of a number of users as a part of the system. It seems to have some cryptographic functions. Implemention doesn't seem to be complete. The subject can split a secret using SSSS and distribute that to a number of recipients. Messages are sent as PGP emails. The subject provides an aliveness signal to the service, and if they miss that signal the service notifies the recipients - who can then recombine their SSSS fragments to reconstruct the secret.	N/A this project is incomplete	N/A this project is incomplete	N/A this project is incomplete	N/A this project is incomplete
petertodd/timelock	https://github.com/petertodd/timelock	2011-04-06	2021-03-07	* Python	Timelock encryption incentivised by Bitcoin. Create a secret key that can be decrypted in a known amount of time using parallel-serial hash chains. The chains are constructed such that Bitcoin addresses can be derived from them and bounties placed, incentivizing third-parties to crack the timelocks.	It looks like a series decryption, but the midpoints are all bitcoin addresses and keys - so incentivising the decryption.	Interesting approach to incentivisation - feels like pass the parcel with a bag of sweets at the centre. This isn't really a DPS implementation - but it is a relevant illustration of timelock encryption using serialised computation.				

Appendix C

Table 04 - evaluations by requirement

Type	Solution	Confidentiality [1]	Awareness [2]	Timing [3]	Resilience [4]	Affordability [5]	Durability [6]	Explainability [7]	Visibility [8]
Hosted	**DeadMansSwitch**	It's reasonable to assume the secret is treated confidentially. However even if it is encrypted, the service retains a key that can be used to decrypt it - (notwithstanding issues of configuration, about which information is not known) making it confidential to all except the service itself, and any system administrators with access to the service's keys. No privacy policy is available.	The system can check aliveness through a number of channels: web push notification, telegram, email, sign in.	A disclaimer states that the switch does not guarantee "anything, not even that it will do its job properly". Whilst likely written this way as a legal defense, this reduces confidence that the switch can reliably meet this requirement. No data on uptime, and no SLA are provided.	The disclaimer on the site clearly states that the service is not intended for high-value messages. The switch is unlikely to prove resilient to high or even medium capability threats to the confidentiality of the secret, integrity or availability of its service.	The switch offers a free tier, and a $50 single payment premium service. Both are considered affordable to a journalist or an organisation they work for.	The only paid tier is a lifetime membership. Unfortunately, this means that the service must steadily gain customers to maintain income. If it cannot, it becomes expensive to maintain.	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.
Hosted	**DeadMan**	The system is hosted on Google App Engine, and protections for that platform apply (dependent on good configuration). Although Dead Man does not have a privacy policy, Google's is substantial and GDPR compliant. It's reasonable to assume that, notwithstanding a misconfiguration, the system is designed to restrict user secret visibility to the service and its administrators.	The system can check aliveness through a number of channels, supported by Twilio and SendGrid: email, SMS, or phone.	The system is based on reliable technologies (Google App Engine, Twilio, SendGrid) and so is expected to be able to deliver messages reliably.	Although composed from several services considered to be resilient (Google App Engine, Twilio, SendGrid), the service itself was developed at a hackathon and is likely best considered a proof of concept. No information about a threat model is available, and so it's reasonable to assume the service is not resilient to well-resourced threats.	The site advertises a free service.	The service was created in 2012, and does not seem to have been updated for a while (as indicated by the fact that it is no longer possible to sign in or create accounts through the various mechanisms offered). With no apparent source of funding, it's likely that after initially creation, this service has not been maintained.	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.
Hosted	**DeadManTracker**	This is the only system surveyed that has a full privacy policy. The system is hosted on AWS - and protections for that platform apply (dependent on good configuration). Data is encrypted at rest, although the service must retain a key to decrypt it - making the user's secret confidential to all except the service itself, and any system administrators with access to the service's keys.	The system can check aliveness through email or push notification.	As an AWS hosted service, the system is based on reliable technologies, so is expected to be able to behave reliably.	The creator has shared priorities for a threat model, and has placed high value on system availability, and resilience to casual threats. Hosting on AWS provides that hhigh availability, and the design has separated the trigger mechanism from the rest of the service - to prevent a denial of service attack on the website from affecting delivery of secrets. The creator acknowledges some personal risk, noting that it's probably easier to attack the operator or individual users of the service, rather than the infrastructure.	The service has a free tier (offering a single switch), or a $10/year subscription service with unlimited switches, file storage, API access, and multiple publishing mediums. Both are considered affordable to a journalist or an organisation they work for.	The service is currently maintained, and the creator is responsive. The subscription model allows for time and costs to maintain the service.	The existence of a privacty policy goes some way towards documentation that a subject could use to determine whether they trust the system with their secrets. This system is *assumed* to follow a classic design. Communication with the operator revealed that it is hosted on AWS, and that subject data is encrypted at rest. However, as with other hosted solutions the architecture and code is not published - making it difficult to evaluate.	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.
Hosted	**Letters Cloud**	Not a lot of information is available. A short statement on the site's page indicates how little information is gathered by the system. Developers have taken pains to ensure their own identities are not listed. Whilst indicative of respect for privacy, this could make it more difficult to trust the system's operators.	The user is given a URL to visit regularly, called the trigger link.	Nothing is known about the service's reliability with regard to timing, but the description suggests that a missed check-in followed by a determined amount of time will result in messages being sent.	The trigger link given to the user has interesting security properties. Instead of regularly reaching out to the user with a reminder the service expects the user to visit the trigger link. The trigger link is considered a secret to be kept and used by the user. This alters the challenge for an attacker, making it more difficult to learn about the user's activity with Letters Cloud - as common communication methods (email, SMS, etc.) are not used. Other information about the service's security properties isn't available.	The service provides a free tier, and content suggests a freemium pricing model - however no information about the cost of the premium service is available, other than the intention to make it possible to pay with cryptocurrency.	The service seems to have fallen into disrepair. Difficulties experienced signing up to the service make it difficult to test.	As with other hosted solutions the architecture and code is not published - making it difficult to evaluate.	No information suggests that the service will indicate whether a switch has been created by a given subject, or about a given threat, and whether it is active.
dApp	**KillCord**	KillCord's smart contract stores the secret as an encrypted file on IPFS. As the smart contract cannot protect the decryption key, KillCord relies on a hidden publisher - which must be able to decrypt the subject's secret to release it. The confidentiality of the publisher is linked to the confidentiality of the user's secret, and relies on the security properties of the system it runs on.	The KillCord smart contract tracks the subject's aliveness: The user's use of Ethereum is considered a proxy for aliveness. and it will only accept evidence of wellbeing as a call to the smart contract made with the identity of the subject's cryptocurrency account.	The KillCord smart contract is guaranteed to run by the Ethereum network, however if the publisher can be located and attacked, it may be possible for an attacker to prevent release of the secret at the appropriate time.	The smart contract is resilient against attempts to compromise its integrity and availability through the properties of the cryptocurrency network. The user's cryptocurrency account is used as the identity to submit well-being check-ins to the smart contract. If located, the publisher (carrying the decryption key for the secret) could be attacked to deny service, alter the integrity of the secret during publishing, or obtain the decryption key and release the secret early. IPFS is used for the release of the encrypted secret, and its decryption key, offering high availability and integrity.	Use of KillCord requires a small amount of *gas* (Ethereum currency to be spent on distributed computing) for the smart contract, aliveness submissions, and final publishing. This has points of comparison to a subscription model.	Once a DPS is established, provided the small amounts of gas required are paid for, the network itself takes care of maintenance. Durability depends on continued support for EVM smart contracts, the Ethereum blockchain, and IPFS.	KillCord is fully described in articles and papers, and its source code repository. Although it relies on a number of complex components, including a smart contract, IPFS, and a hidden publisher - it should be possible to explain in simple terms. However, it is also anticipated that most investigative journalists have not developed sufficient expertise to create and configure a DPS using the command-line tools provided. This, combined with the requirement to learn about several complex aspects of the system before trusting it, may make it difficult to trust without advice from a trusted person.	The encrypted secret on IPFS may serve as a visible indicator that a KillCord switch is in use, provided it were advertised by the subject to indicate they had created it.

Appendix C **Table 04 - evaluations by requirement**

Type	Solution	Confidentiality [1]	Awareness [2]	Timing [3]	Resilience [4]	Affordability [5]	Durability [6]	Explainability [7]	Visibility [8]
dApp	**Kimono**	Kimono first breaks the user's secret into SSSS fragments, across a network of N peers to share with the public only once the time-lock is complete. Each fragment is encrypted for its recipient and distributed by IPFS. No individual can reveal the secret, and K or more of the participants would need to collude to release the secret early.	N/A Kimono is a time-lock solution. It would need to be adapted to serve as a DPS.	The smart contract behaves reliably with high availability - it's a good mechanism for determining when to release the secret. (As mentioned, Kimono is a time-lock solution, though.) The participants are incentivised to release their fragment on time, and will be rewarded by the smart contract for doing so.	At the heart of the solution is a K of N network of peers. Resilience of the system relies on the appropriate behaviour of at least K participants. This protects the confidentiality, integrity and availability of the system. Secret fragments are distributed by IPFS - considered highly available and with controls for integrity.	There's a balance between affordability and timing/resilience of the system: Assuming the only incentive for participants is financial reward, the subject must select a high enough reward to make their network resilient against bribes.	Durability depends on continued support for EVM smart contracts, the Ethereum blockchain, and IPFS.	Kimono is described in a formal paper, but as its site is no longer available simple explainers are not available. It ought to be possible to describe but as with KillCord, it relies on a number of complex components and this, further complicated by the need to trust a group of participants, may require a subject to spend time and effort understanding it before they could trust it.	The encrypted secret on IPFS may serve as a visible indicator that a Kimono time-lock is in use, provided it were advertised by the subject to indicate they had created it. The distribution of SSSS fragments to a number of participants may also serve to indicate that Kimono is in use, provided a party were sufficiently interested to discover this on IPFS.
dApp	**SilentDelivery**	Extends the model of Kimono, providing protections for the identities of recruited participants (mailmen).	N/A As with Kimono, SilentDelivery is a time-lock solution.	(As mentioned, SilentDelivery is a time-lock solution, though.)	Additional resilience obtained through silent recruitment of mailmen, from a large network of registered *possible* participants - making it significantly harder to identify and compromise participants.	Offers lightweight and heavyweight mode (only used when a particular mailmain is challeneged - ie. if they appear to have released their fragment early) - significantly reducing the cost of operation.	Durability depends on continued support for EVM smart contracts, the Ethereum blockchain, and IPFS.	Whilst designed to be an improvement to Kimono, SilentDelivery represents an increase in complexity. It is described in its paper as an open protocol, and that protocol has several features that make it complex to understand. For instance, it can operate in multiple modes in order to reduce the cost of the smart contract activity that drives it. Similarly, the steps that it takes to protect the anonymity of participants in the network (mailmen) push it further from being an intuitive solution. A subject would have to spend time and effort to understand it before they could trust it.	SilentDelivery goes to lengths to protect the identity of participants (mailmen). However, as with Kimono, it distributes the encrypted secret by IPFS - and this may serve as a visibile indicator that it is in use, provided the subject advertised that they had created it.
OSS	**skickar/ DeadManSwitch**	Holds the decryption key in RAM for the duration of execution. The original secret file is deleted (but not blitted). If used as a DPS, this means interrupting power (eg. by stealing the device) will destroy the key. However, the deleted file represents a risk as it may still be recoverable.	Monitors Twitter API for a given passphrase. Twitter activity is used as a proxy for well-being.	If used as a DPS, Able to take action (eg. to decrypt the secret) if the monitored twitter account does not tweet the required passphrase in a given time period.	Interrupted power to the device (or halting the process) can result in loss of availability. Resilience against confidentiality and integrity attacks are dependent on the security properties of the device that the application runs on.	Extremely affordable, requiring only python installed on a very low power device, and an internet connection.	This is a simple script, and not designed for much longevity. It was last maintained in 2019, and may well be abandoned now. Assuming the device is patched, the application risks becoming out of date as available python versions will eventually deprecated and then become unavailable. The device and OS will need to be patched and restarted regularly to ensure that the runtime environment has up to date security patches.	This is a simple script, and as the subject must run it on their own device, they are able to understand the extent to which they can trust the system. It performs a simple task that is easy to explain. However, it requires that the subject have some familiarity with a command shell to use, putting it into the realm of those with at least some confidence using linux-based systems. Explaining its weaknesses and limitations may also require some effort.	The script isn't designed to indicate to an outsider that it is running (and doing so might put it at risk).
OSS	**h313/ dead-mans-switch**	Makes no attempt to encrypt the message, password, or target email address. Reliant on the security properties of the server it's hosted on.	Operates as a web service, and receives a POST of a password as a proxy for the well-being of the user. If this is interecepted or learned, the switch can be suppressed.	Will take action if the password has not been provided for a day.	Serves a POST endpoint that accepts a password. No apparent rate limiting or other protections on the endpoint mean it makes itself a target for attack. (NB. rate limiting could be implemented by a firewall or other protection.) If succesfully exploited, all CIA properties are at risk. Runs on a single device, so not resilient to interrupted power, which may result in loss of availability.	Extremely affordable, requiring only python installed on a very low power device, and an internet connection.	As with other OSS projects, patching is required for security patches to the OS and software. If not maintained, this puts application at risk of code rot from breaking changes in python and libraries.	This is a simple service, and does what it needs to. However, it requires that the user stand up an instance of it, and manage that service. This puts it beyond the reach of many journalists, who shouldn't have to develop these skills. Explaining its weaknesses and limitations may also require some effort.	This service isn't designed to indicate to an outsider that it is running (and doing so might put it at risk). However, it ought to be possible to determine that the service is running with a port scan or crafted query - so it may be possible to detect.
dApp	**deadmenswitch/ dms**	Smart contracts can't really hold secrets - their state is known to the EVMs on the nodes that execute them - and a tampered EVM could be persuaded to give up the data held by the contract. This means that the subject's secret, and the identities of the subject and beneficiary (recipient) aren't well protected.	A web server facilitates subject's interaction with the smart contract. A call to the smart contract's **kick** function from the subject's Ethereum identity is the proxy for well-being, and serves to increase the user's switch timeout.	This is driven by recipients of the message calling **getDataFromAddress** on the smart contract to see if the switch they are beneficiary of has a message for them. NB. It seems as if a small bug in **getDataFromAddress** will grant beneficiaries immediate access to the message without checking the expiry.	Benefits from distributed computing across Ethereum blockchain - giving it high availability, and integrity properties.	Thought to be reasonably affordable, this smart contract isn't doing a lot of work. However, it does have to store all secrets and the identities for each switch - storage costs gas, and this could increase if the service were popular.	Longevity depends on the state of the Ethereum blockchain. Provided it continues to be mined, and proves a valuable computing resource for many other purposes, the switch will endure.	This is a more complex dApp, based on the premise that a smart contract will do some things reliably and with high availability. Explaining this to a user may require a complex understanding of distributed apps and cryptocurrency. Explaining its weaknesses, even more so.	The existence of the smart contract should serve to show that dms is in use. However, this also has to disadvantage of advertising the content of the secret, too, as the EVM cannot protect its confidentiality. If used, users are forced to balance visibility of their switch against confidentiality of its secret.
OSS	**EsmailELBoBDev2/ Dead-man-s-switch**	It uses the OS keyring service, which makes it a little harder to learn the password required to suppress the secret. However, the secret itself is not encrypted, and resides in python code. Confidentiality of the secret relies on the security properties of the OS.	The user enters the password in a terminal on the machine the script is running on. This is a proxy for the subject's well-being.	data.txt contains the dates which control the application. Manipulating it could cause late/early release the next time the script is run. The script must be run regularly to ensure a reliable service.	The data.txt file isn't encrypted or protected, and contains the control dates for "today" and "tomorrow", so they could easily be manipulated to get a different behaviour at next run; and the secret itself is in plaintext in the code so easy to manipulate. Similarly, any mechanism to run the script regularly (for availability) is also dependent on the OS for its integrity.	Extremely affordable, requiring only python installed on a very low power device, and an internet connection.	As with other OSS projects, patching is required for security patches to the OS and software. If not maintained, this puts application at risk of code rot from breaking changes in python and libraries.	This is a simple script that takes advantage of the operating systems keyring facility. This is a smart use of a strong tool, but may also require some explaining to a user to ensure they understand what has been made safe (especially in the light of the other weaknesses of this design). As with the other scripts, it requires the subject to have some familiarity or confidence with command-line based applications.	This script has no way to indicate that it is running (and as the secret is stored in the code itself, and its control dates are stored in plaintext, doing so would put it at risk of attack).

Appendix C **Table 04 - evaluations by requirement**

Type	Solution	Confidentiality [1]	Awareness [2]	Timing [3]	Resilience [4]	Affordability [5]	Durability [6]	Explainability [7]	Visibility [8]
OSS	**dmp1ce/ DMSS**	It runs on a single server, so relies on the security properties of the host OS. The secret is initially split using SSSS, and after that deleted from the subject's computer. SSSS fragments are delivered to recipients using PGP encrypted email. Provided recipients behave appropriately, fragments will not be recombined until the right time. Protection of the fragments relies on the properties of the recipients devices and operating systems.	The user will indicate their aliveness to the system with an (as yet undefined) signal. That hasn't been implemented yet.	Provided the service is running, the system will note a missed signal and notify the recipients.	The design makes efforts to ensure that the secret does not remain whole and unencrypted in any one place. Instead, it resides on recipient devices. Resilience of the secret's confidentiality ultimately relies on the participants themselves. Integrity of the secret is assured, as altered fragments will not recombine correctly. SSSS is a K of N threshold scheme - so provided only a few fragments are altered or missing, the secret can be reconstructed. If the switch itself is attacked, it could be made unavailable. In that instance, participants might need to discover for themselves that it had been attacked in order to them collaborate and reconstruct the secret.	Extremely affordable, requiring only haskell installed on a very low power device, and an internet connection. PGP is a freely available technology.	Unlike other solutions, this design also indicates that the system should make regular checks to ensure that the recipients still have their SSSS fragments. As with other OSS projects, patching is required for security patches to the OS and software. If not maintained, this puts application at risk of code rot from breaking changes in haskell and libraries.	This seems to be an early phase of a design for a system that will be hosted. It relies on two complex concept: SSSS and PGP - both of which can be explained to users, but may require a little effort to understand.	It is not known if this design will eventually incorporate a signal to indicate that it is in use.

Definitions of each requirement, as defined in the thesis body, are reproduced at the end of this appendix.

Appendix C	Table 05 - scored evaluations							
Type	Solution	Confidentiality	Awareness	Timing	Resilience	Affordability	Durability	Explainability	Visibility
Hosted	DeadMansSwitch	1	2	2	1	3	1	1	0
Hosted	DeadMan	1	2	3	1	3	0	1	0
Hosted	DeadManTracker	2	2	3	2	3	3	2	0
Hosted	Letters Cloud	2	2	2	2	3	0	1	0
dApp	KillCord	2	3	3	2	3	3	2	2
dApp	Kimono	3	0	3	2	3	3	2	2
dApp	SilentDelivery	3	0	3	3	3	3	2	2
OSS	skickar/DeadManSwitch	1	1	2	1	3	1	2	0
OSS	h313/dead-mans-switch	0	1	2	1	3	1	2	0
dApp	deadmenswitch/dms	1	3	2	1	3	3	2	1
OSS	EsmailELBoBDev2/Dead-man-s-switch	0	1	2	0	3	1	2	0
OSS	dmp1ce/DMSS	2	0	2	2	3	2	2	0

Scoring		
	0	Does not meet requirement, or no information available.
	1	Shows awareness of the requirement, with significant room for improvement.
	2	Partially meets the requirement, with some room for improvement.
	3	Meets the requirement comprehensively, with little or no room for improvement.

[1] Confidentiality: A DPS must keep a user's secret confidential until it determines it is appropriate to release the secret.

[2] Awareness: A DPS must have a mechanism to check the subject's well-being, and a method to release the secret if the subject fails this test.

[3] Timing: A DPS should not release the subject's secret early (ie. whilst they are still alive and well), late (ie. too long after the subject has become unresponsive), or never.

[4] Resilience: A DPS must assume the existence of, and protect against, attempts by hostile threats to compromise the secret's confidentiality and integrity, and be resilient against attacks intended to compromise its availability (CIA properties).

[5] Affordability: A DPS should use affordable technologies to provide its functionality, not relying on resources beyond the means of the target subject group.

[6] Durability: A DPS should be expected to last a significant amount of time (eg. the lifetime of a subject), with continued maintenance (including security patches), support, or upgrade pathways if the technologies in use become obsolete.

[7] Explainability: A DPS must be presentable as a model that the target subject group can understand and trust.

[8] Visibility: A DPS must be able to present evidence that it is operational, to contribute to its deterrent effect.

Appendix D: Subjective logic detail

This appendix provides a brief overview of subjective logic, a technique for reasoning about trust relationships between entities (or people) in networked groups.

Trust relationships are categorised as:

- **Direct functional trust** in a proposition (ie. A's belief that participant B will behave appropriately)
- **Indirect functional trust** (ie. B informs A how likely they believe it is that C will behave appropriately)
- **Direct referral trust** (ie. A's trust in B's advice about C)

Detail

In *Trust Network Analysis with Subjective Logic*, 2006, [1] Jøsang, Hayward, and Pope describe trust opinions in terms of subjective logic. Jøsang's 2016 book, Subjective Logic [2], is a valuable resource that informs this in-depth exploration.

Each trust opinion is composed of a trust component (effectively a probability assigned to the proposition), and a certainty component (a second-order probability, expressing confidence in the trust component). Together, these form a *trust opinion.*

In an example where the subject has formed direct functional trust of N participants (ie. an opinion of their behaviour), and wishes to combine that trust for the proposition "all participants will behave appropriately for the DPS today," this can be represented as:

$O_1 \wedge O_2 \wedge ... \wedge O_N$

Here:

- O_i represents the subject's opinion of person i.

NB. In terms from Subject Logic, this is an *aleatory* opinion, based on a *frequentist situation* (ie. each day is a new opportunity to measure the behaviour of a participant in the DPS). This could be a little misleading though, as it's likely easier to behave correctly on a daily basis while the DPS is not 'activated.' Should the DPS be needed, the likelihood of events that might change the behaviour of participants (blackmail, conflict of interest...) are increased - and this raises a possible second, *epistemic* opinion, based on a *non-frequentist situation* - "On the day it's needed, all participants in the DPS will behave appropriately."

When considering an unknown person, a subject may decide to seek the advice of an agency (ie. to perform some background checks and determine how trustworthy they consider that person to be, or how vulnerable to compromise they are). That trust can then be chained together:

$O_3 = O_1 \cdot O_2$

Here:

- O_1 represents the subject's direct referral trust (ie. of agency B's opinion) - this could be based on documentation about their background check process, or direct personal experience of their previous recommendations.
- O_2 represents agency B's direction functional trust of C, a person the subject wishes to learn about (eg. the agency's opinion of C, based on a background check).
- O_3 represents the subject's new, indirect functional trust opinion of C.

NB. In theory, O_1 could, in turn, be a chain composed of the subject's trust in a certification authority that has assured the agency's processes, and the rating of a certificate that they have awarded to the agency (which then leads to the question of how to develop some functional trust (direct, or indirect) in the certification authority). The chain could be extended again to incorporate an understanding of the reputation of the authority and knowledge of any registers of complaints, evidence of any lawsuits suggesting poor practices, etc. See: Figure 1

Deciding when to stop adding layers of indirection is a difficult problem to solve.

When considering a network of participants, assuming that the subject works with an agency about which they have a referral trust opinion. It's possible to combine the agency's opinions of each person in the

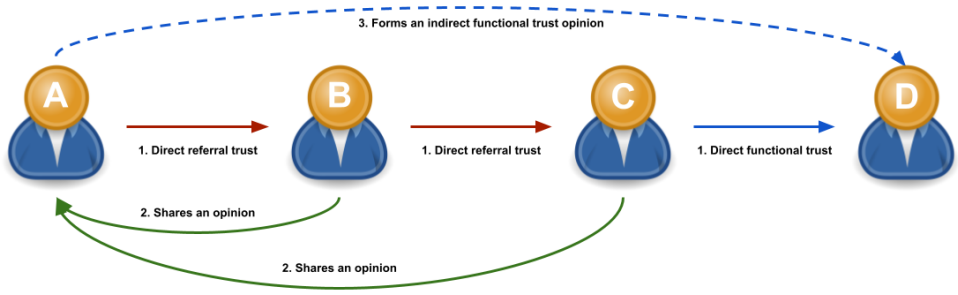


Figure 1: A longer chain of referrals

network without referral trust of the agency to determine the subject's functional trust of the original proposition:

$(O_1 \cdot O_2) \wedge (O_1 \cdot O_3) \wedge ... \wedge (O_1 \cdot O_{N+1})$

Here:

- O_N represents the subject's direct referral trust of the agency's opinion.
- $O_{X>1}$ represents the agency's direct functional trust of person $X - 1$.

The predicate for these individual opinions is binomial (the opinion of it is that it is true or false, with a given degree of uncertainty). The opinions are broken into components:

- b_X = Belief mass
- d_X = Disbelief mass
- u_X = Uncertainty
- a_X = The base rate (probability distribution in the absence of any beliefs)

These individual components can be visualised[1] as a barycentric triangle, as illustrated in Figure 2.

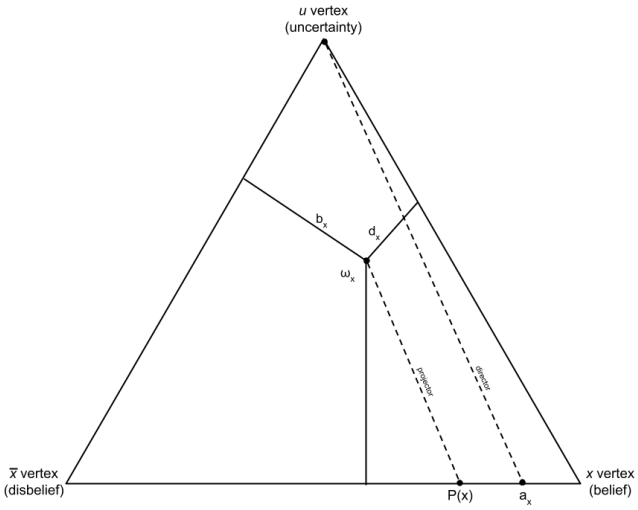


Figure 2: Barycentric triangle visualisation of binomial opinion[2]

Consider this example:

- Domain X contains outcome x (Xavier, from the participant network, is compliant), and outcome $\neg x$ (Xavier is not compliant).
- Domain Y contains outcome y (Yan, from the participant network, is compliant), and outcome $\neg y$ (Yan is not compliant).

[1] See also, the Opinion Visualiser, University of Oslo: https://folk.universitetetioslo.no/josang/sl/BV.html
[2] Reproduced from Subjective Logic [2], Jøsang, p25, Fig 3.1

The subject has formed an opinion of each of these positive outcomes (ω_x, and ω_y), and now would like to form an opinion of their conjunction:

$\omega_{x \wedge y} = (xy)$ (both Xavier and Yan are compliant in the domain $X \times Y$)

This is the dotted, highlighted area inside the cartesian product of the domains, as illustrated in Figure 3.

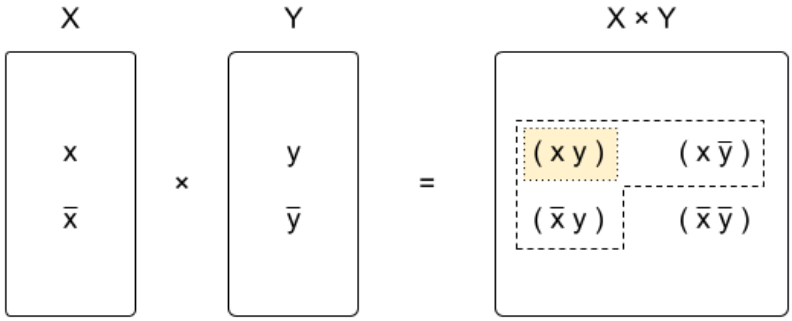


Figure 3: Cartesian product of two domains[3]

As the opinions are independent, this is calculated by applying binomial multiplication, as described in Subjective Logic [2], pp.101-102.

- Let: $\omega_x = (b_x, d_x, u_x, a_x)$ (an independent opinion on x)
- Let: $\omega_y = (b_y, d_y, u_y, a_y)$ (an independent opinion on y)[4]

$$\omega_{x \wedge y} : \begin{cases} b_{x \wedge y} = b_x b_y + \frac{(1-a_x)a_y b_x u_y + (1-a_y)u_x b_y}{1-a_x a_y} \\ d_{x \wedge y} = d_x + d_y - d_x d_y \\ u_{x \wedge y} = u_x u_y + \frac{(1-a_y)b_x u_y + (1-a_x)u_x b_y}{1-a_x a_y} \\ a_{x \wedge y} = a_x a_y \end{cases}$$

This can also be represented as: $\omega_{x \wedge y} = \omega_x \cdot \omega_y$

This can be visualised[5], and Jøsang provides an example in Subjective Logic [2] showing the multiplication of two opinions. See: Figure 4

An interesting observation, which may be helpful in reasoning about the opinions a subject would have to make when formally assessing a group of N participants in a DPS, emerges: The uncertainty of the product falls between the uncertainties of the two opinions, and so expect to see an 'averaging' effect where the outcome uncertainty is less extreme than any outliers.

System reliability

In section 7.2 of Subjective Logic [2], Jøsang describes application of subjective logic to system reliability.

A system, S, can contain a number of components, with dependencies on each other. Serial dependencies in a reliability graph show components that must both function in order for the system to function, whereas parallel dependencies represent components of a system where one or the other are required.

K of N systems are simple cases of a system that can be modelled with a reliability diagram.

For a K of N system, knowing K and N it's possible to compose a rough and ready reliability network using only these primitives.

eg. Figure 5 illustrates a participant network for $K = 2$, and $N = 3$, where x, y, and z represent the compliance of the 3 participants in the network. This network can be represented as: $(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$

The application of subjective logic multiplication (for $\wedge$) and co-multiplication (for $\vee$) would allow the representation an opinion of the K of N network.

[3] Reproduced from Subjective Logic [2], Jøsang, pp.101-102, Fig 7.1
[4] This represents a small typo correction from the description in Subjective Logic, which has used ω_x for both opinions.
[5] See also, the Operator Visualisation Tool, University of Oslo: https://folk.universitetetioslo.no/josang/sl/Op.html
[6] Reproduced from Subjective Logic [2], Jøsang, pp.103, Fig 7.2

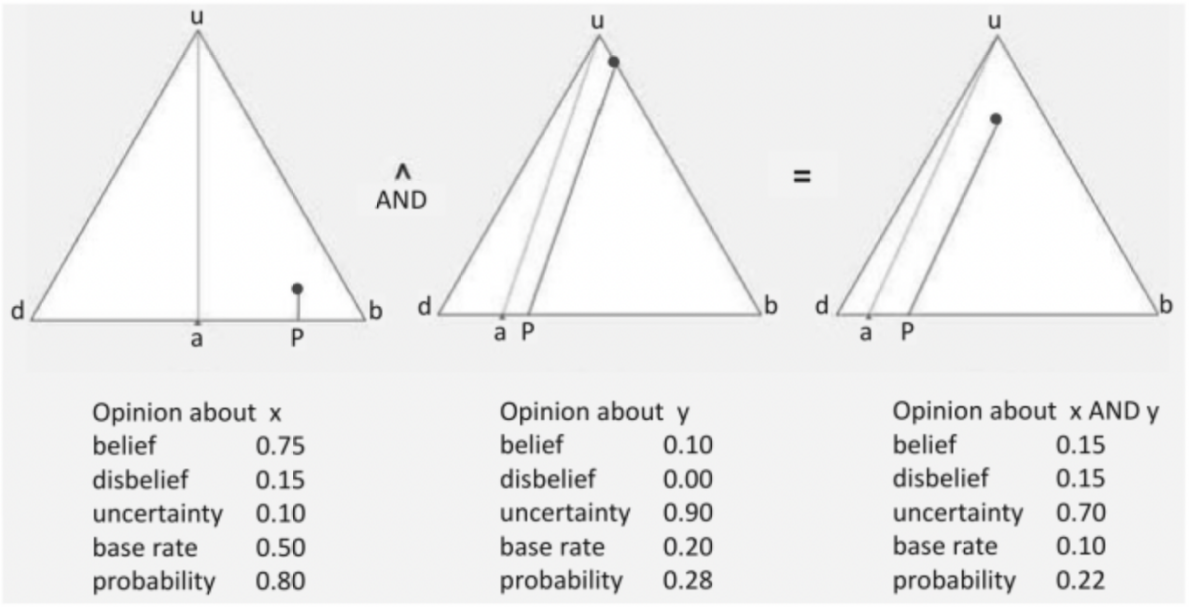


Figure 4: Binomial opinion multiplication[6]: $P(x \wedge y) = P(x) \cdot P(y) = 0.8 \cdot 0.28 = 0.22$

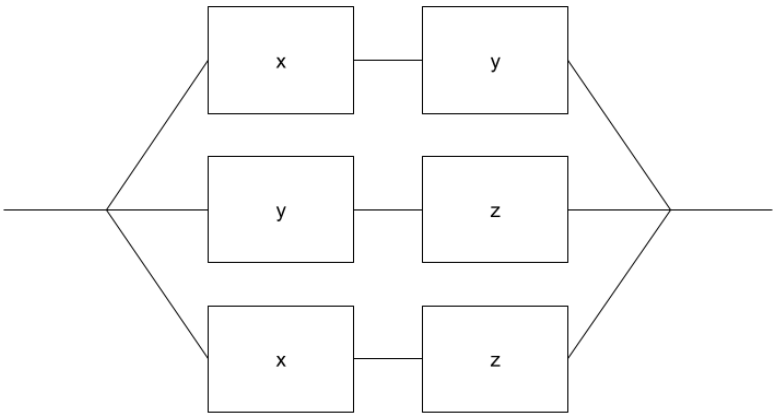


Figure 5: $K = 2$ of $N = 3$ reliability network: $(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$

This particular technique does not scale particularly well, as large networks will expand terms quickly. Each combination must be represented, creating $\binom{K}{N}$ parallel paths. Automated tools would be better suited to this task than manual expansions.

1. Josang A, Hayward R, Pope S. Trust network analysis with subjective logic. *Conference Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*. Australian Computer Society, 2006, 85–94.

2. Jøsang A. *Subjective Logic*. Springer, 2016.

Appendix E: Paralysis proofs detail

This appendix provides a brief overview of paralysis proofs, a technique for managing group control of an asset in the absence of one member:

- In the absence of a member, the group may initiate transfer of control to the remaining subset of members by issuing a challenge.
- The absent member may respond to the challenge, preventing further changes to the control of the asset.
- If they do not respond within a specified period, the asset is transferred to an account controlled by the remaining subset of the group.

In *Paralysis Proofs: Secure Dynamic Access Structures for Cryptocurrency Custody and More* [1], Zhang, Daian and Bentov describe paralysis proofs as a process by which a group may hand a fund to an account controlled by a smart contract or SGX enclave.

In doing so, this entity is treated as a trusted but *constrained* 3rd party. Smart contracts and SGX enclaves will execute the code they were set up with (as agreed by all parties) reliably, and this property distinguishes such a solution from simply holding the funds in escrow with a regular 3rd party. *This explanation refers to the constrained 3rd party as SGX.*

Paralysis Proofs exploit properties of blockchain based cryptocurrencies to ensure that the funds can only be passed to the *remaining parties* in the complete absence of the *missing party.*

In the following example, the group comprises 3 members:

- First the full fund is placed in $UTXO_0$, only able to be spent with the signatures of all 3 members together or by the SGX.
- One member of the group is determined to be missing by the remaining parties.
- The remaining parties issue a challenge, and the SGX prepares 2 transactions:
  - $t1$: a trivial amount to be transferred to $UTXO_1$, which can be spent either by the missing party, or by the SGX provided X blocks have passed since its release to the blockchain. This is referred to as a life signal.
  - $t2$: the transfer of $UTXO_0$ and $UTXO_1$ to a new location controlled by the remaining parties.

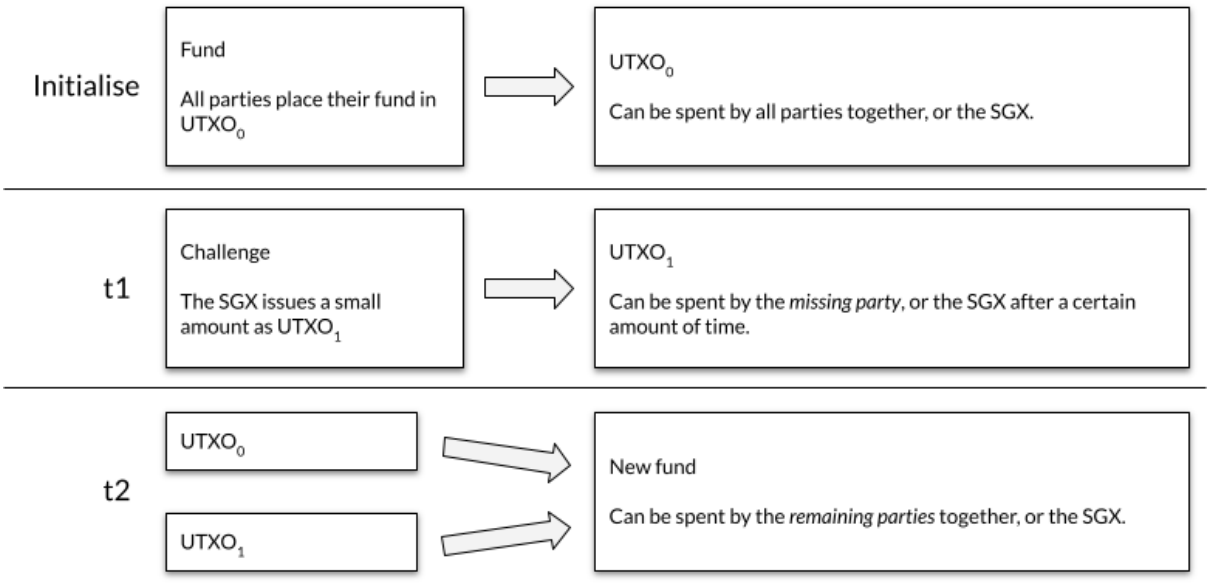


Figure 1: Paralysis proof: $t2$ relies on $UTXO_0$ and $UTXO_1$

Principle: $t2$ relies on $UTXO_0$ *and* $UTXO_1$, so cannot resolve until $t1$ has been resolved.

In order to gain control of the funds, the remaining parties must release $t1$ to the cryptocurrency network, wait until it has been added to the blockchain, and then wait until an additional X blocks have been

added[1]. After that, they may release $t2$, which will allow them to transfer the funds into a location they control.

However, during that time, $t1$ also created $UTXO_1$ (the life signal). If it is spent by the missing party, then it can no longer serve as input to $t2$ - and $t2$ will fail, so preventing the transfer of funds away from $UTXO_0$.

This mechanism allows the individual to participate in a scheme that is attempting to determine whether they are able to respond.

Some benefits:

- Cryptocurrencies are popular and rely on distributed networks.
- Smart contracts run on distributed systems (usually cryptocurrency blockchain verification networks) and offer guarantees of execution. It becomes impossible to execute a denial of service attack on a distributed system without attacking the majority of nodes (or exploiting another weakness of the system that relies on it).
- SGX and smart contracts offer guarantees that the right code will be executed, and that only that code may sign its outputs with its given identity (so allowing third parties to verify that it ran correctly).

1. Zhang F, Daian P, Bentov I *et al.* Paralysis proofs: Secure dynamic access structures for cryptocurrency custody and more. *AFT 2019 - Proceedings of the 1st ACM Conference on Advances in Financial Technologies* 2019:1–5.

[1]This is considered a proxy for the passage of time, as cryptocurrency network blockchains grow at a steady rate.

Appendix F: Trusted computing detail

This appendix provides some additional detail for trusted computing.

Computing systems are built in layers, from the hardware, CPU, through BIOS, operating system, and applications - as shown in figure 1.

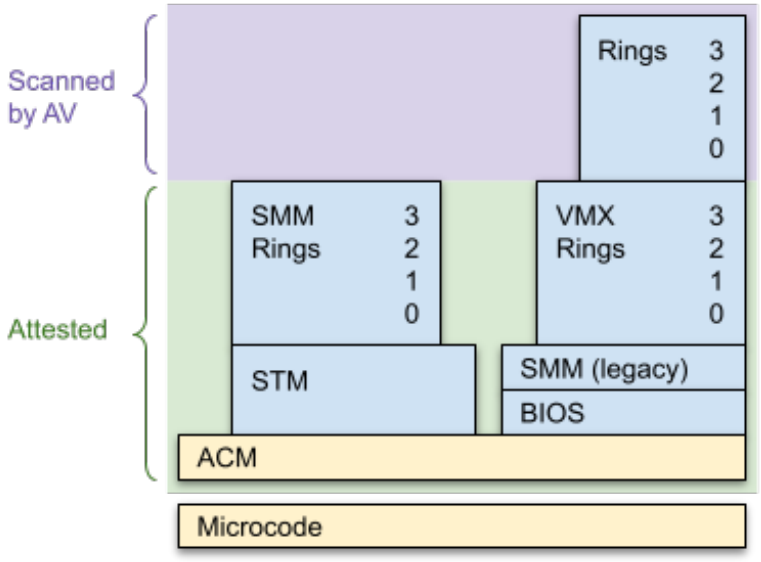


Figure 1: Hardware and operating system layers in an Intel CPU

Efforts to protect software from exploitation, by closing vulnerabilities, have incentivised attacks on "layers below" - as software must trust that the lower layers are behaving correctly in order to know anything. Trusted Computing considers all layers from the bottom up - including system hardware.

Trusted Platform Module (TPM)

Attestation is a process by which layers are measured and validated - determining if the system is in a known state - essential for a secure boot process. The Trusted Platform Module (TPM) initiates measurement of the first layer in the boot process, and provides the capability to reliably measure subsequent layers.

If all is working as it should, a Trusted Computing system offers the guarantee that a user can know with certainty that:

- The software running is the intended software.
- The layers below it - the Operating System, BIOS, and underlying CPU are also running the intended code (or microcode).

Described by Andrew Martin [1], the TPM provides a number of components to support a secure (measured, or verified) boot process - including the Root of Trust for Measurement (RTM).

- A "measured boot process" builds a chain of trust from the RTM, by measuring each layer in the sequence before executing it.
- An "authenticated boot process" also verifies each measurement before proceeding - so allowing the system to abort if it has been altered.

1. At power on ("platform reset") the RTM records its own identity.
2. The RTM takes a hash of the next component to execute, stores that value in a safe place[1], and transfers control to it.
3. This is repeated until the operating system has been executed.

A *static* RTM measures the entire boot chain. However, this is complicated and the chain has many elements:

[1] Platform Configuration Registers (PCR) inside the TPM. These are special and have only 1 operation: extend.

- Patching means new expected values need to be managed.
- Sometimes the order of execution at boot changes.
- Performance may be impacted as there are a lot of things to measure, and this must all happen at once.

A *dynamic* RTM provides a special CPU instruction. On execution, a fixed piece of code is used to measure and launch a nominated piece of software. This significantly simplifies the process - and still permits a hardware measurement of each piece of software - so providing a similar guarantee of integrity.

As Martin notes [1]:

> "any program, at any point in the chain, can gain confidence in (i) its own integrity, and (ii) the integrity of the components it relies upon, by comparing the stored values with the ones it was expecting"

TPMs, previously a hardware feature for enterprise computing, are now built into consumer PCs and required by Windows 11. The assurances provided by a TPM could help to build confidence in DPS solution code running on such a system.

However, reliance on a TPM also comes with a number of risks:

- The implementation can be complex. ie. The chains can be long and software/firmware is often updated (and this is why dynamic root of trust for measurement is sometimes preferred).
- Trusted computing, even when behaving fully correctly, could still be exploited. ie. A piece of software can have full integrity, but may still be manipulated through malicious inputs to give undefined/unwanted behaviours.
- It's hard to know if a *remote* machine is tainted or not, as a hardware attack with sufficient resources could compromise it.
- Many national import/export controls require that the TPM exposes only asymmetric encryption capabilities.
- Drawbacks for law enforcement (these are the same objections seen with all privacy protecting technologies). The TPM specification requires that there be no back door into devices.
  - The tussle between privacy technologies and law enforcement continues today - and the arguments in 2015 paper *Keys Under Doormats* [2] are just as strong now as they were at time of writing.
- Vulnerabilities, once discovered, require manual intervention to patch.
  - For example, ROCA (CVE-2017-15361) [3] was a vulnerability found in Infineon TPM 1.2 chips in 2017, that used a weak process to generate RSA keys.
  - This impacted the Estonian digital ID system at the time - which quickly led them to move to a new infrastructure with fresh keys.
  - NCSC have published information [4] about how to determine if a PC is vulnerable to ROCA, and how to patch it if it is. Patching like this relies on awareness and engagement from hardware owners.

SGX

Trusted Execution Environments offer opportunities to run code and work with memory that is protected from other code running on the system. Code running in a TEE is expected to behave correctly and to retain its secrets, even if the operating system, and all other software on the system, is compromised.

To achieve this, SGX (available on many Intel CPUs) provides a number of hardware capabilities:

- Memory pages are reserved from the system's physical RAM, and encrypted. These are not addressable from the operating system[2].
- Enclaves are protected memory regions where an application can store code and secrets.
- SGX applications are divided into *untrusted* code (normal application code), and *trusted* code - which is granted access to the enclave, and is expected to process application secrets.
- When an application creates an enclave and calls a trusted function inside it, the CPU switches to a special mode and during this time no other processes can run.
- This gives the trusted part of the application plaintext access to the memory it needs to run without fear of exposure.

[2] In fact, attempts to access these pages are denied, even from privileged system code, the OS, VMM, BIOS, SMM, etc.

Working with SGX is not without risk. In *SGX Explained*, 2016, [5], Costan and Devadas review existing literature around SGX and its implementation - offering a warning that SGX does not seem to be (at the time of writing) an appropriate tool for managing trusted enclaves.

> "our security analysis reveals that the limitations in SGX's guarantees mean that a security conscious software developer cannot in good conscience rely on SGX for secure remote computation"

At time of writing it is 2021, and a lot can happen in 5 years, but it's worth noting that any solution that exploits SGX for its security properties may also need to provide a fallback capability when running on machines that cannot provide SGX hardware.

- Such a system is still vulnerable to denial of service attacks - and DPS systems are particularly sensitive to loss of availability.
- Networks of machines operating SGX or other TEE can mitigate this to some degree, but to participate in the network requires SGX support. (This is improving as new chips roll out, but if there's a vulnerability in a legacy SGX implementation that then can exclude a proportion of participants.)
- SGX support is more common now amongst Intel CPUs.
- Software and data in SGX is inaccessible to malware scanning applications - which means that malicious inputs might still be able to exact an unplanned behaviour from the software (eg. an image processing enclave could be attacked with an image that contains an exploit for the processing code). If this happens, it may still be possible to persuade an SGX enclave to leak data or perform undesirable operations.

1. Martin A. The ten-page introduction to trusted computing. 2008.

2. Abelson H, Anderson R, Bellovin SM *et al.* Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity* 2015;**1**:69–79.

3. Nemec M, Sys M, Svenda P *et al.* The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. *24th ACM Conference on Computer and Communications Security (CCS'2017).* ACM, 2017, 1631–48.

4. ROCA: Infineon TPM and secure element RSA vulnerability guidance. 2017. https://www.ncsc.gov.uk/guidance/roca-infineon-tpm-and-secure-element-rsa-vulnerability-guidance

5. Costan V, Devadas S. Intel sgx explained. *IACR Cryptol ePrint Arch* 2016;**2016**:1–18.

Appendix G - design and risk tables

This appendix presents tables used in the design and risk analysis process for a DPS including:

* the derived requirements for a DPS (Table 01),
* the stakeholders for a DPS (Table 02),
* the generic components of a DPS (Table 03),
* assets, impact, and threat actor tables for a generic DPS (Tables 04-06), in the context of investigative journalism, and
* the tables supporting a full risk analysis process applied to the micro-services solutions proposed in the thesis body (Tables 07-13).

List of tables

Appendix G	Table 01 - requirements for a DPS
Requirement	**Description**
Confidentiality	A DPS must keep a user's secret confidential until it determines it is appropriate to release the secret.
Awareness	A DPS must have a mechanism to check the subject's well-being, and a method to release the secret if the subject fails this test.
Timing	A DPS should not release the subject's secret early (ie. whilst they are still alive and well), late (ie. too long after the subject has become unresponsive), or never.
Resilience	A DPS must assume the existence of, and protect against, attempts by hostile threats to compromise the secret's confidentiality and integrity, and be resilient against attacks intended to compromise its availability.
Affordability	A DPS should use affordable technologies to provide its functionality, not relying on resources beyond the means of the target subject group.
Durability	A DPS should be expected to remain operational for a significant amount of time (eg. the lifetime of a subject), with continued maintenance (including security patches), support, or upgrade pathways if the technologies in use become obsolete.
Explainability	A DPS must be presentable as a model that the target subject group can understand and trust.
Visibility	A DPS must be able to present evidence that it is operational, to contribute to a deterrent effect.

Appendix G	Table 02 - stakeholders for a generic DPS
Actor	**Role**
Subject	Owner of a specific DPS. Inputs the initial secret, provides evidence of own well-being. Depends on a reliable release of the secret if they are not. Depends on the reliability of the DPS. Incentivised to protect their own safety, and to take steps to ensure the reliability of the DPS.
Operator	Owner of the DPS system itself, incentivised in some way to manage the resilience of the various components in the system.
Platform provider	Owner of the hardware and infrastructure that the DPS system operates on. Incentivised to provide a reliable operating system and hardware on which the DPS system can run.
Participant	(Optional. Required by some designs.) Member of a group tasked with determination of the state of the subject's well-being. Incentivised to protect their own safety, and the safety of the subject. Motivated towards appropriate behaviour in some way (eg. through loyalty to the subject, or a reward scheme).

Appendix G	Table 03 - components of a generic DPS
Component	**Role**
Initialiser	Component that handles the secret initially provided by the Subject. Arranges safe communication and storage of the secret, and establishes a method for the DPS to check the well-being of the subject.
Secret store	Component that stores the subject's secret in such a way that it can be extracted only when the right conditions are met.
Aliveness checker	Component that can monitor evidence of the subject's well-being, and is able to suppress the secret extractor if the subject is alive and well (or activate it if not).
Secret extractor	Component that can retrieve the secret from the secret store, and publish it to the publishing medium.
Publishing medium	Either a public medium where the secret can be published, or direct communication of the secret to specific recipients.

Appendix G	Table 04 - assets for a generic DPS	
Asset	**Name**	**Description**
A001	Subject identity	A representation of the identity of the subject, used to evaluate the evidence of well-being presented, to ensure it matches the identity of the user that owns the DPS.
A002	Secret	Representation of the subject's secret. Stored and protected by the secret store.
A003	Secret extraction information	Information that may be used to extract the secret - eg. a decryption key.
A004	Operational information	Additional information relating to the operation of the DPS - eg. a period for recurring aliveness checks, chosen behaviour should the checks fail (eg. a cooling-off period before activating the secret extractor, and information about the Publishing medium).

Appendix G	Table 05 - impact of loss for generic DPS assets			
Asset	**Asset name**	**CIA**	**Impact**	**Level**
A001	Subject identity	C	Loss of confidentiality of the subject's identity may represent a risk, as the subject may have chosen not to reveal the details of their switch (or how to locate it), to prevent it becoming an attack surface. However, to act as a deterrent, the existence of a switch must be declared - and so the positive angle is that this information may constitute evidence that the switch is operational.	Medium
A001	Subject identity	IA	The subject identity is essential for determining the well-being of the subject. If this is altered or made unavailable, the switch will not have enough information to determine the state of the subject. In this case, it must follow the safest course of action. This is not a trivial decision.	High
A002	Secret	C	Loss of confidentiality of the secret held in the switch itself can have a high impact. If the secret is released early, it can no longer be used as a disincentive to physical attack. It may also inform the attacker just how much (or how little) impact the secret could have, or may allow them to put mitigations in place to plan for the secret's publication. These all weaken the switch as a protective measure.	High
A002	Secret	IA	Alteration or deletion of the secret is a powerful attack and this grants an attacker the ability to 'defuse' the switch entirely, removing its disincentive to physical attack, and rendering it useless.	High
A003	Secret extraction information	CIA	Confidentiality, Integrity, Availability | The information required to extract the secret from the switch should remain entirely under the control of the switch. If this information becomes known, it offers insights into the secret itself (see the impact of loss of confidentiality for the secret). If it can be tampered with, it directly affects the availability of the secret too. If the switch loses the ability to extract and publish the secret it holds, it is rendered useless.	High
A004	Operational information	C	Operational information, such as the frequency and chosen publishing details of the switch are of some value to an attacker. They show the intended frequency of aliveness checks, and details of the recipients (or medium of publication). In turn, this information could be used to improve the quality of an attack against the switch, subject, or recipients - increasing the risk that individuals involved will be attacked, or the switch disabled. However, as with the subject identity, releasing information to the effect that the switch is operational may contribute to the switch's deterrent effect.	Medium
A004	Operational information	IA	Ability to alter or remove the operational information could render the switch inoperable (or impractical - for instance, if the period for aliveness checks were increased beyond a desirable limit).	High

Appendix G	Table 06 - threat actors for a DPS (investigative journalism)	
Threat actor	**Description**	**Capability**
Nation state (extra-legal means)	Nation states and their intelligence agencies are considered to have limitless resources with which to attack systems - making 0-day vulnerabilities, and expensive physical or social engineering attacks available.	Very High
Nation state (legal means)	Nation states may choose to subject the operators or platform providers of DPS services to legal pressure to halt their service or reveal stored secrets.	Very High
Organised crime	OC gangs may have various motivations for attacking a switch - either to prevent information about themselves being released, or to cause the release of information about their own adversaries. Criminals may also act unpredictably, displaying behaviours such as revenge.	High
Activists	Some activists may not approve of practises that withhold incriminating information for any reason. They may perceive it as cowardly, irresponsible, or even blackmail. To uncover and release this information, activists may use technical or social engineering attacks. Activism is not believed to be as well funded as organised crime or national security organisations.	Medium
Law enforcement	Considered separate from nation states, law enforcement bodies that perceive DPS activity as blackmail (or another activity at odds with the law) may attempt to shut down such systems, or arrest those operating it. Some law enforcement bodies have a limited range of legal activities that they can engage in, or may be called upon to enforce decisions made by local justice systems.	Medium
Script kiddies	A high volume threat on the internet, script kiddies are people who, for reasons such as boredom, curiosity, or malice, continuously test endpoints for vulnerabilities (without much thought to the nature of the endpoint) - exploiting them when found.	Low

Appendix G, Table 07 - risk analysis combination tables

Likelihood of success		Capability		
		L	**M**	**H**
Vulnerability level	**L**	L	L	M
	M	L	M	H
	H	M	H	H

Threat level		Frequency of attempt		
		L	**M**	**H**
Likelihood of success	**L**	L	L	M
	M	L	M	H
	H	M	H	H

Risk level		Impact		
		L	**M**	**H**
Threat level	**L**	L	L	M
	M	L	M	H
	H	M	H	H

Risk priority		Impact		
		L	**M**	**H**
Threat level	**L**	9	7	4
	M	8	5	2
	H	6	3	1

Appendix G	Table 08 - additional assets and impacts, micro-services design				
Asset	**Name**	**Description**	**CIA**	**Impact**	**Level**
A005	Architecture	Data controlling the design of the system, creation of micro-services, databases, trust zones, rules for permitted connections between micro-services.	I	If altered, the system's behaviour could be adjusted, resulting in the loss of confidentiality of subject secrets, or loss of availability of the system.	H
A006	Administration account	Account and credentials required to construct or modify system architecture.	C	If confidentiality of the credentials is compromised, any attacker can alter the system's architecture and change any part of its behaviour.	H
A006	Administration account	Account and credentials required to construct or modify system architecture.	IA	If access to the account is lost (ie. if the bill cannot be paid, the credentials are modified without knowledge of the operator, or the cloud hosting provider prevents access for some other reason), either: 1. the service will be shut down (in which case, switches become unavailable), or 2. it becomes difficult to patch or update the service. Eventually exploits will be found that can be used to attack it.	H
A007	DMZ services, input zone services, data zone services	Web services in the DMZ responsible for creating new switches, and for accepting evidence of subject aliveness; services in the input zone that coordinate creation of switches and recording of subject aliveness evidence; services in the data zone that store operational and secret data, and share it with the check zone.	IA	If services in any of these zones are compromised, it may be possible to alter the behaviour of the service - either to make it unavailable, or to reject user input. It may also be possible for a compromised service to record the user's credentials and use them to play back false aliveness evidence at a later date.	H
A008	Subject aliveness evidence secret	A secret that the subject shares with their aliveness assertion.	C	If this becomes known, an attacker may use it to submit false aliveness evidence.	H
A009	Data zone services	Services in the data zone that store operational and secret data, and share it with the check zone.	C	If the data in the secret store is released, it will be encrypted with a key retained in the HSM, and so of little use to an attacker. However, operational information (pseudonyms and data used to verify aliveness information) may be uncovered and this could be used to manipulate the switch into accepting false aliveness information.	M
A010	HSM	The HSM in the data zone contains keys required to decrypt subject secrets.	C=A	If the HSM is compromised it is expected to render itself inoperable. This results in user secrets becoming unavailable. If this can be detected, users will be forced to resubmit their secrets in order to restore operation.	M

A011	Check zone services	Services in the check zone responsible for checking aliveness, extracting secrets, and publishing if necessary.	IA	If the secret extractor is compromised, it could lead to loss of confidentiality of the user's secret. If either service is altered or disabled, it could lead to the service becoming unable to carry out secret extraction and publishing if the subject becomes unavailable.	H
A012	Operator	The person or persons that operate the system	IA	If the operator is attacked or compromised in some way, the could be persuaded to release secrets permitting an attacker to take control of the system, learn subject secrets, or shut down the service.	H

Appendix G	Table 09 - vulnerabilities, micro-services design			
Vuln ID	**Vulnerability**	**Assets affected**	**Description**	**Vuln level**
V001	Human weakness	A012 (operator)	Staff at the cloud platform provider, or the operator, may be vulnerable to bribery, threat, or disgruntled staff may act on malice. Humans are vulnerable to duress, and also make mistakes like having guessable passwords.	M
V002	Configuration errors	A005 (architecture), A007 (services), A009 (services), A010 (HSM), A011 (services)	Mistakes or deliberate errors in the configuration for security appliances (such as the firewall), any backups, micro-services or the HSM.	M
V003	Coding flaws	A007 (services), A009 (services), A011 (services)	Weaknesses in the micro-services themselves.	M
V004	Unpatched software vulnerabilities	A007 (services), A009 (services), A010 (HSM), A011 (services)	Managed micro-services are considered to be at low risk from OS patching. However, the micro-services themselves will need regular patching as new vulnerabilities are discovered.	M
V005	Default passwords	A006 (admin acct), A010 (HSM)	Accounts used to manage infrastructure with default passwords.	H
V006	Physical vulnerability of data centre	A005 (architecture), A007 (services), A009 (services), A010 (HSM), A011 (services)	The data centres administered by large cloud providers are considered to have adequate security in place to ensure only authorised staff may enter for good reasons.	L
V007	Capacity limits of architecture	A007 (services)	It may be possible to overwhelm the service's public facing endpoints.	M
V008	Legal compulsion	A012 (operator)	Staff at the cloud platform provider, or the operator, are more likely to choose cooperation with a legal request over a hefty fine or a custodial sentence.	H

Appendix G	Table 10 - threats, micro-services design							
Threat ID	**Threat**	**Actor**	**Motive**	**Cap.**	**Target Assets**	**Method**	**Freq.**	**Notes**
T001	Denial of service	Nation state (extra-legal means)	Prevent the service from operating, to protect their own secrets	H	A007 (services)	Overwhelm the front-end services with requests.	L	Considered Low frequency in some democracies, but far higher (or even automatic) in some countries.
T002	Denial of service	Organised crime	Prevent the service from operating, to protect their own secrets	H	A007 (services)	Overwhelm the front-end services with requests.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.
T003	Denial of service	Activists	Prevent the service from operating because it is perceived as unethical to withhold evidence of wrongdoing	M	A007 (services)	Overwhelm the front-end services with requests.	M	This could form a part of an ongoing campaign if activists are sufficiently motivated.
T004	Exfiltrate secrets	Nation state (extra-legal means)	Learn secrets stored by dissidents or investigative journalists investigating government wrongdoing	H	A001-A004 (informatic A009 (data zone), A011 (check zone), A012 (operator), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	M	This is an attractive target for an intelligence agency that wishes to control dissidents and investigative journalists.
T005	Exfiltrate secrets	Organised crime	Learn secrets stored by enemies or people they wish to blackmail / Sell the information or accept payment not to release it	H	A001-A004 (informatic A009 (data zone), A011 (check zone), A012 (operator), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.
T006	Exfiltrate secrets	Activists	Embarrass the service, and leak secrets because it is perceived as unethical to withhold evidence of wrongdoing	M	A001-A004 (informatic A009 (data zone), A011 (check zone), A006 (admin acct)	Capture and decrypt information held in the secret store: either through learning or guessing the administrator credentials, or by exploiting vulnerabilities in the micro-services.	L	Activists are considered to have limited resources to do this repeatedly, and may not need to - as the goal of reputational damage to the service would be achieved if sufficiently many subject's secrets were compromised.
T007	Destroy secrets	Nation state (extra-legal means)	Prevent evidence of government wrongdoing from being released	H	A001-A004 (informatic A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	Considered Low frequency in some democracies, but far higher (or even automatic) in some countries with less regard for rule of law.

T008	Destroy secrets	Organised crime	Prevent evidence of their own wrongdoing from being release, or make it possible to attack an enemy	H	A001-A004 (informatic A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, by obtaining the administrator credentials through duress, or by exploiting vulnerabilities in the micro-services.	L	This may be considered infrequent, as organised crime are likely to have one-off reasons for doing this.
T009	Destroy secrets	Activists	Prevent the service from operating because it is perceived as unethical to withhold evidence of wrongdoing	M	A001-A004 (informatic A009 (data zone)	Delete information held in the secret store: either through learning or guessing the administrator credentials, or by exploiting vulnerabilities in the micro-services.	L	Activists are considered to have limited resources to do this repeatedly, and may not need to - as the goal of reputational damage to the service would be achieved if sufficiently many subject's secrets were deleted.
T010	Install malware	Nation state (extra-legal means)	Make service inoperable, with plausible deniability	H	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	L	This is difficult to do on a managed micro-services infrastructure - and once done, there would be little benefit to doing it again, as the service would be disabled.
T011	Install malware	Organised crime	Exploit computation resource to make some money, or ransom back the capability to operate	H	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	M	This is difficult to do on a managed micro-services infrastructure, but organised crime are incentivised to do it repeatedly if they can.
T012	Install malware	Script kiddies	Excitement, reputation	L	A007 (services)	Exploit vulnerabilities in the micro-services to install malware on the various hosted services.	M	This is difficult to do on a managed micro-services infrastructure, but persistent *attempts* are to be expected by any service.
T013	Weaken secret protections by altering architecture	Nation state (extra-legal means)	Learn about future secrets and users	H	A005 (architecture), A006 (admin acct)	Adjust the architecture to remove protections for secrets, or install additional accounts that can be used to obtain secrets in future, by learning or guessing the administrator password (or obtaining it through duress); or by making imperceptible changes to the architecture designs and waiting for the operator to refresh the system from this data.	L	This activity can be detected, and is not considered the easiest way to achieve the same weakening of the service's secret protections. However, it may serve a purpose and could be quicker to implement than legal compulsions.
T014	Weaken secret protections by mandating a backdoor	Nation state (legal means)	Learn about future secrets and users	H	A005 (architecture), A006 (admin acct)	Take the operator or platform service provider to court, or threaten to, to incentivise them to install a backdoor (ie. an additional account), or alter the micro-services, to make it possible for a government body to obtain the secrets stored in the DPS.	M	Wide-reaching mass surveillance programmes (such as Prism) have been revealed in recent years. Data collection was achieved by legal (and other) means. New programmes would be equally hard to discover. It is likely that major providers are collaborating with law enforcement, or intelligence agencies, to *some* degree.

T015	Weaken secret protections by altering architecture	Organised crime	Learn secrets stored by enemies or people they wish to blackmail / Sell the information or accept payment not to release it	H	A005 (architecture), A006 (admin acct)	Adjust the architecture to remove protections for secrets, or install additional accounts that can be used to obtain secrets in future, by learning or guessing the administrator password (or obtaining it through duress); or by making imperceptible changes to the architecture designs and waiting for the operator to refresh the system from this data.	L	Organised crime may not care if their changes are discovered provided they can exploit them reasonably swiftly.
T016	Shut down service with court order	Nation state (legal means)	Prevent use of the switch for a variety of reasons	H	A001-A004 (informatic A006 (admin acct), A012 (operator)	Take the operator, or the cloud platform provider, to court (or threaten to) to incentivise them to halt the account hosting the DPS.	L	Shutting down the entire service seems less useful to a nation state than exploiting it to gather information.
T017	Fines for switches used for blackmail	Law enforcement	Enforce blackmail legislation	H	A012 (operator)	Take the operator, or the cloud platform provider, to court for alleged use of the DPS for blackmail.	M	This could happen. It's likely that individual cases would be dealt with, ie. an individual DPS might attract attention of law enforcement, and the court case may require only the removal of an individual switch.
T018	Fines or jail for refusing to release secrets	Law enforcement	Support criminal investigations, incentivise release of secrets stored by people engaged in illegal activity	H	A012 (operator)	Threaten the operator with fines, or a custodial sentence, to incentivise them to release secrets belonging to a person of interest.	L	This is a more extreme case, where law enforcement wish to learn the information kept in the switch.

Appendix G	Table 11 - risks, micro-services design											
Risk ID	**Threat IDs**	**Vuln IDs**	**Asset IDs**	**Risk**	**Capability**	**Vuln level**	**Likelihood of success**	**Freq. of attempt**	**Threat level**	**Impact**	**Risk level**	**Risk priority**
R001	T001, T002, T003	V007	A007	DoS attack by overwhelming the public facing endpoints.	H	M	H	L	M	M	M	5
R002	T004, T005, T007, T008	V001, V002, V005	A001-A004, A006	Exfiltrate or destroy secrets by guessing or learning the administrator credentials.	H	H	H	M	H	H	H	1
R003	T004, T005, T007, T008	V002, V003, V004	A001-A004, A009, A011	Exfiltrate or destroy secrets by exploiting vulnerabilities in the micro-services.	H	M	H	M	H	H	H	1
R004	T004, T005, T006, T007, T008, T009	V001	A001-A004, A012	Exfiltrate or destroy secrets by learning the administrator credentials through duress.	H	M	H	M	H	H	H	1
R005	T010, T011, T012	V002, V003, V004	A007	Install malware by exploiting vulnerabilities in the micro-services.	H	M	H	M	H	H	H	1
R006	T013, T015	V002, V003	A005	Adjust the architecture to remove protections for secrets by making imperceptible changes to the architecture files and waiting for the operator to refresh the infrastructure.	H	M	H	L	M	M	M	5
R007	T013, T015	V001, V005	A005, A006	Adjust the architecture to remove protections for secrets by learning or guessing the administrator credentials.	H	H	H	L	M	M	M	5
R008	T013, T015	V001, V005	A005, A006, A012	Adjust the architecture to remove protections for secrets by learning the administrator credentials through duress.	H	H	H	L	M	M	M	5
R009	T014	V008	A005, A006	Weaken secret protections by mandating a back door through legal means.	H	H	H	M	H	M	H	2
R010	T016	V008	A001-A004, A006, A012	Shut down the service by threatening the operator with court (or taking them to court and threatening them with a file or custodial sentence).	H	H	H	L	M	H	H	1

R011	T017, T018	V008	A001-A004, A006, A012	Shut down an individual subject's switch by taking the operator or cloud platform provider to court for aiding blackmail, or other legislation in criminal investigations.	H	H	H	M	H	L	M	7

Appendix G	Table 12 - responses, micro-services design						
Risk ID	Risk	Risk level	Risk priority	Avoid	Accept	Transfer	Reduce
R001	DoS attack by overwhelming the public facing endpoints.	M	5	☐	☐	☐	☑
R002	Exfiltrate or destroy secrets by guessing or learning the administrator credentials.	H	1	☐	☐	☐	☑
R003	Exfiltrate or destroy secrets by exploiting vulnerabilities in the micro-services.	H	1	☐	☐	☐	☑
R004	Exfiltrate or destroy secrets by learning the administrator credentials through duress.	H	1	☐	☐	☐	☑
R008	Install malware by exploiting vulnerabilities in the micro-services.	H	1	☐	☐	☐	☑
R009	Adjust the architecture to remove protections for secrets by making imperceptible changes to the architecture files and waiting for the operator to refresh the infrastructure.	M	5	☐	☐	☐	☑
R010	Adjust the architecture to remove protections for secrets by learning or guessing the administrator credentials.	M	5	☐	☐	☐	☑
R011	Adjust the architecture to remove protections for secrets by learning the administrator credentials through duress.	M	5	☐	☐	☐	☑
R012	Weaken secret protections by mandating a back door through legal means.	H	2	☑	☑	☐	☐
R013	Shut down the service by threatening the operator with court (or taking them to court and threatening them with a file or custodial sentence).	H	1	☑	☑	☐	☐
R014	Shut down an individual subject's switch by taking the operator or cloud platform provider to court for aiding blackmail, or other legislation in criminal investigations.	M	7	☐	☑	☐	☐

Appendix G	Table 13 - controls, micro-services design								
Risk ID	**Risk**	**Control ID**	**Control**	**Strategy**	**Tactic**	**Operational**	**Cost**	**Use**	**Justification**
R001	DoS attack by overwhelming the public facing endpoints.	C001	Implement load balancing and DoS resilience with a service such as cloudflare.	Reduce	Correct	Technical	M	☑	This is an accepted method of developing resilience to DoS attack.
R002	Exfiltrate or destroy secrets by guessing or learning the administrator credentials.	C002	Implement strong password requirements.	Reduce	Correct	Technical	L	☑	This makes it much harder to guess an account.
		C003	Require 2FA for all accounts.	Reduce	Correct	Technical	L	☑	This makes it much harder to compromise an account remotely, even if the password is guessable.
		C004	Implement a backup regime.	Reduce	Correct	Technical, Procedural	L	☑	Being able to restore state quickly in the event of loss is good - although worth recognising that the backups represent an increased attack surface.
R003	Exfiltrate or destroy secrets by exploiting vulnerabilities in the micro-services.	C005	Implement code reviews, testing policies, and regularly apply dependency analysis to code. Regular pen-testing.	Reduce	Correct	Procedural	L	☑	This is a very affordable way to improve code quality and catch vulnerabilities (especially in 3rd party libraries) as soon as possible.
		C006	Apply monitoring software for unexpected activity	Reduce	Detect	Technical	M	☑	It's important to understand when the service has been compromised, even if the damage is irreversible it informs future solutions and allows the service to warn the users.
R004	Exfiltrate or destroy secrets by learning the administrator credentials through duress.	C007	Hire bodyguards for all operator staff.	Reduce	Correct	Physical	H	☐	This is considered too expensive for staff. Other means can reduce the risk sufficiently.
		C008	Rent an office that provides physical security guards. Add personal safety training, and subsidise home security equipment.	Reduce	Correct	Physical	M	☑	This is an accepted method of increasing personal security for staff.
R008	Install malware by exploiting vulnerabilities in the micro-services.	C009	*As controls for R003*						
R009	Adjust the architecture to remove protections for secrets by making imperceptible changes to the architecture files and waiting for the operator to refresh the infrastructure.	C010	Engage the development team to regularly review the code for unexpected changes.	Reduce	Detect	Technical	L	☑	This is a very affordable way to detect these changes.
R010	Adjust the architecture to remove protections for secrets by learning or guessing the administrator credentials.	C011	*As controls for R002*						
R011	Adjust the architecture to remove protections for secrets by learning the administrator credentials through duress.	C012	*As controls for R004*						

R012	Weaken secret protections by mandating a back door through legal means.	C013	Rely on a cloud service provider that operates in a country less likely to mandate this.	Reduce	Prevent	Procedural	M	☑	Careful selection of physical location for the service can lead to a far safer experience.
R013	Shut down the service by threatening the operator with court (or taking them to court and threatening them with a fine or custodial sentence).	C014	Rely on a cloud service provider that operates in a country less likely to mandate this.	Reduce	Prevent	Procedural	M	☑	Careful selection of physical location for the service can lead to a far safer experience.
R014	Shut down an individual subject's switch by taking the operator or cloud platform provider to court for aiding blackmail, or other legislation in criminal investigations.	C015	Accept this risk.	Accept	Directive	Procedural	L	☐	Acknowledge that this is not the ideal experience for an individual user, but in order to keep the service running it's necessary.
		C016	Contend the request in court to ensure that there's a good reason for this.	Reduce	Correct	Procedural	L	☑	Many countries to allow a degree of legal recourse to ensure that power such as this is not abused.
		C017	Distribute keys across multiple jurisdictions	Reduce	Prevent	Procedural	M	☑	Whilst expensive, designing the service to use multiple keys distributed across multiple jurisdictions makes it much harder to compel the service to release keys held under any one legal requirement. This increases resilience against legal compulsions.

Appendix G		Table 14 - scored evaluations proposed designs, contrasted with existing systems							
Type	Solution	Confidentiality	Awareness	Timing	Resilience	Affordability	Durability	Explainability	Visibility
Hosted	DeadMansSwitch	1	2	2	1	3	1	1	0
Hosted	DeadMan	1	2	3	1	3	0	1	0
Hosted	DeadManTracker	2	2	3	2	3	3	2	0
Hosted	Letters Cloud	2	2	2	2	3	0	1	0
dApp	KillCord	2	3	3	2	3	3	2	2
dApp	Kimono	3	0	3	2	3	3	2	2
dApp	SilentDelivery	3	0	3	3	3	3	2	2
OSS	skickar/DeadManSwitch	1	1	2	1	3	1	2	0
OSS	h313/dead-mans-switch	0	1	2	1	3	1	2	0
dApp	deadmenswitch/dms	1	3	2	1	3	3	2	1
OSS	EsmailELBoBDev2/Dead-man-s-switch	0	1	2	0	3	1	2	0
OSS	dmp1ce/DMSS	2	0	2	2	3	2	2	0
Proposed	**#1 Microservices architecture design**	2	2	3	2	3	3	3	0
Proposed	**#2 Distributed application**	3	3	3	3	3	3	2	3
Proposed	**#3 Application of witness encryption**	3	3	3	3	3	3	1	3

Scoring	0	Does not meet requirement, or no information available.
	1	Shows awareness of the requirement, with significant room for improvement.
	2	Partially meets the requirement, with some room for improvement.
	3	Meets the requirement comprehensively, with little or no room for improvement.