# Background

## Importance of strong and unique passwords and why it isn’t enough

As aforementioned, it is intended to be an extended password manager that focuses on compartmentalisation, before exploring this idea, the need for a password manager must be investigated. Typically, the way personal data is secured is through a password, which the user chooses, although a common issue of this is that many of these users tend to either use a weak or the same password throughout all of their accounts, simply because they deem it to be troublesome,­­­­­­­­­ as understandably, manually managing many complex passwords isn’t easy. This was elaborated from a research commissioned by LastPass in 2016 in which 2000 respondents were questioned about their password habits where 45% of them believed that their accounts aren’t worth a hacker’s time, where also 43% of the same 2000 respondents prioritise a password that is easy to remember than more secure (Christopherson, 2016).

Some examples of feeble passwords include dictionary words, sequential numbers and repeated characters, but the issue with these passwords isn’t just that they are easy to guess but they’ve likely been cracked before, which makes them more vulnerable to dictionary attacks with the use of precompiled passwords lists; in fact, there are “words lists” distributed online consisting of exposed passwords which attackers take advantage of for these types of attacks, similarly, “rainbow tables” are used to crack password hashes. Moreover, the length of these passwords also contribute to their vulnerability as a shorter password means that a computer would have less combinations to guess, this was demonstrated in a study where a dictionary attack with 7 billion trials per second resulted in an 8-character alphanumeric password taking 0.36 minutes to crack, but a 13-character alphanumeric would take 906,123 years (Cheswick, 2012).

Regardless of how strong a password may be, it just might not be enough as there is nothing that can be done in the case of a data breach, which is entirely possible on any database no matter how secure they are, and in such cases, there is little for these users left as a simple advice they are given is just to simply just “change those old passwords”, which doesn’t solve anything. This was further explored in an article in “NordPass” where a total of 9,517 unsecured databases were identified across 20 different countries, with notable ones being Facebook and Amazon records, the latter which didn’t even require a password to access (Webster, 2020), but this isn’t all as between January and September of 2019 alone there were over 7.9 billion data records exposed, consisting of personal information such as names, email addresses, physical address and much more (Bekker, 2020).

## Proposed encryptions – AES

While a password may not be enough for your *privacy*, it does majorly assist in security against general attacks against users, such as the aforementioned brute forcing and dictionary attack methods, it’s due to this that password managers are becoming popular and and so to understand this, the way these passwords are stored must be explored.

The proposed method of encrypting the credentials is using the popular AES (Advanced Encryption Standard) algorithm, this is a 128-bit symmetric block cipher that takes an input and returns the ciphered text of that input, both of which are 128 bits, and this is ciphered using a cipher key which can be 128/192/256 bits. The way it works is by using the cipher key in multiple rounds as a “round key” for the cipher where in each round; a new round key is generated through the key expansion routine and 4 different functions (specifically, SubBytes, ShiftRows, MixColumns, AddRoundKey (Dworkin, et al., 2001)) are used to substitute and permutate the input in a table of bytes, and at the end, the round key is mixed in with this and proceeds to the next round. The number of rounds is determined by the length of the key as shown below, which is slower, but the encryption is stronger.

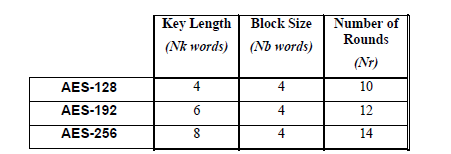


Figure 1 (Key-Block-Round Combinations. (Dworkin, et al., 2001))

To decrypt the ciphered text, the ‘Inverse Cipher’ must be used, in which the above-mentioned functions are inversed and used on the ciphered text, which still requires the same cipher key for it to work as it’s a symmetrical cipher. The result showing of a plaintext before and after its encryption using AES (128 bit key) is shown below:

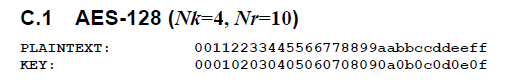


Figure 2 Plaintext and key for AES-128 (Dworkin, et al., 2001)



Figure 3 Output of AES-128 encryption (Dworkin, et al., 2001)

This shows the strength of this algorithm as it completely obfuscates the plaintext as it shows no correlation between the two. Brute forcing an AES-256 ciphered password would take roughly 1,100 followed by 75 zeroes (Chernev, 2020). This encryption’s still complex, however, there are libraries available that encapsulate its operations, making it easier to implement AES, such as the ‘AesCryptoServiceProvider’ class given in the ‘Cryptography’ namespace in C#, as well as the “aes” crate available in Rust.

## Proposed encryptions – bcrypt

The other proposed method of encryption is bcrypt, this is an adaptive password hashing algorithm which uses the Blowfish keying schedule, and unlike AES, it isn’t symmetric, meaning that it doesn’t use a key at all; it takes in a 128-bit salt and encrypts a 192-bit value (Mazières, 1999).

While bcrypt isn’t symmetric, it heavily relies on Blowfish, which similarly to AES, it is a symmetric block cipher as it takes in a key (32 to 448 bits), and Blowfish uses numerous rounds to encrypt its input where in each round the input is modified by shifting its bytes using the cipher key. Bcrypt uses this by producing the cipher key for Blowfish by using its ‘ExpandKey’ function with the cost, salt, and the initial password as its parameters, and this ‘cost’ factor is what makes the encryption strong as it greatly slows the hashing by multiplying the amount of rounds the encryption goes through. By making the process of hashing slower, de-hashing will be just as slow, which is very effective against any attack, and in addition to this, bcrypt keeps up with Moore’s law where only the work rate needs to be increased to make the hash slower as the computers get faster (Coda Hale, 2010).

The salt parameter is simply an addition to the hash function which modifies the output so that the encrypted hash can change regardless if the initial password stays the same. However it is worth knowing that the salt becomes pointless against dictionary or brute force attacks especially if the salt is known by the attacker, although it does work effectively against rainbow tables (which contains lists of precompiled hashes) which are rendered useless considering a password hashed a salt will have a different hash in comparison to the same password without a salt.

Below shows an example output of a hash string generated with bcrypt:

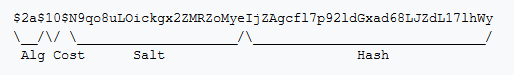


Figure 4 Bcrypt example hash string (contributors, 2020)

As mentioned before, bcrypt specifically doesn’t have a key to input as it’s generated based on its parameters, but this means it might be better to use it in synergy with another encryption method to encrypt the master password itself where its hash is for an example passed onto the AES algorithm as the cipher key, so that the credentials may never directly link to the plaintext master password although it is still used to encrypt the credentials. There are libraries such as ‘BCrypt.Net’ for C# as well as the ‘pwhash’ crate for Rust, both of which allows easy implementation of this algorithm.

## Similar applications – LessPass

Unlike many other password managers, LessPass is very unique, because it introduces a way of non-synchronisation by allowing the offline generation of passwords for sites mainly by using the site’s URL, the user’s email address and name. This also means that you don’t require a LessPass account to generate these logins as the only benefit of such an account is to save the user’s logins; this introduces a second layer of security as without an account, there is no possible way of your login details for these sites being compromised when using this system, this requires a potential attacker to know all of the attributes beforehand to access a password for a site. Below shows how the password generation operates:

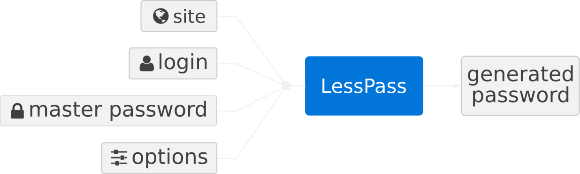


Figure 5 LessPass password generation function (Guillaume, 2016)

If the user chose to create a master account and save their logins, upon generating a password and saving it, it creates a profile in a JSON format containing all of the attributes used to generate the password for the site, such a profile is shown below:

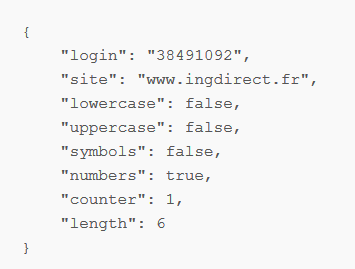


Figure 6 A LessPass profile (Guillaume, 2016)

Notice that this doesn’t include the master password, because this is only included upon the usage of the profile when logging into the site, and so this makes the system very simple as it removes the idea of saving passwords in a database which also eliminates synchronisation, and it is also open source, which allows anyone to check and contribute to its codebase as well as self host it for the user’s master account.

However, an issue with LessPass is that you can only save just a password for a site, and not allowing you to store notes or any other fields which may be site-specific such as a double password (ProtonMail has this feature), and this is important because the password generation uses the site’s URL, and so if the URL remains the same throughout various forms in the same site, it will only allow just one password to be generated. Furthermore, you cannot import existing passwords nor manually set a password, and the only way to change it is by changing changing the “counter” field that it provides as an option.

## Similar applications – LastPass

LastPass is a password manager which is entirely online as it comes with a web interface and it was released as early as 2008. While it stores credentials as normal, what makes LastPass stand out from the others is with its distinctive features, an example being that it allows the user to change the password of a saved login which would automatically change it directly on that account to reduce time consumption. However, what’s more interesting is the authentication of LastPass where the user can create a one-time password as a temporary replacement of their master password to access the vault, which is great for when accessing it in public/untrusted places where your connection may not be encrypted, this is shown below:

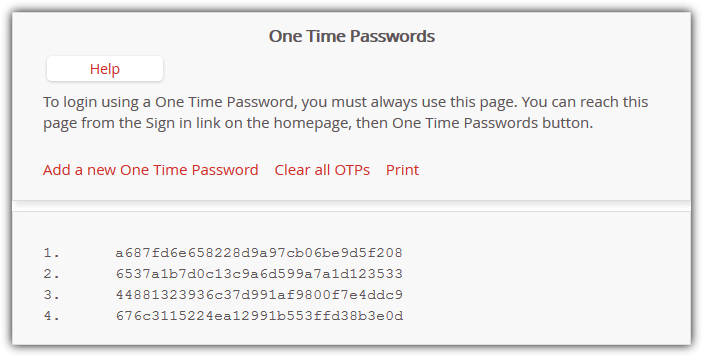


Figure 7 One Time Passwords (Raymond, 2011)

Regardless of these features, over the years, LastPass had several security incidents mainly in its browser extension, where in one of these incidents, their network was breached and thus many key attributes of these LastPass accounts were compromised (LastPass, 2015), although the encrypted data such as the passwords were not exposed as the master passwords are not stored on the server and so it’s still considered a great password manager.

A key point is that LastPass isn’t open source even regardless that it has a free plan, if it was open source and self-hostable, most users wouldn’t need to worry about these breaches as not only would these vulnerabilities be fixed with community effort but with self-hosting, it’s harder to be targeted by these attacks, regardless of the vulnerabilities. Not to forget with LastPass being mostly web-based, it introduces a lot of performance issues both on client and server sides, which may include slow login times and vault load times. Furthermore, considering LastPass has synchronisation and stores the credentials on their servers, if the user currently has no connection in their current device, if they modify one of the login details from a different device, not only would the current device not receive it but it may introduce conflicts if it’s also updated on the current device. There is also no way to opt out of storing these credentials on their servers regardless if they are also stored locally, which is another issue.

## Similar applications – Bitwarden

Similarly to LastPass, Bitwarden also has an online vault, but it also has a standalone application which can be accessed offline, it is also open source and has been audited numerous times. Bitwarden remains as a simple password manager as it simply does what the user needs by allowing the user to save credentials, but a key premium feature is that within those credentials, the user can include and allocate a TOTP (Time-based One Time Password) so that if the user has enabled two-factor authentication for those credentials, the TOTP can directly be generated and accessed, this eliminates the need of an external authenticator which ultimately saves time for the user, this is shown below:

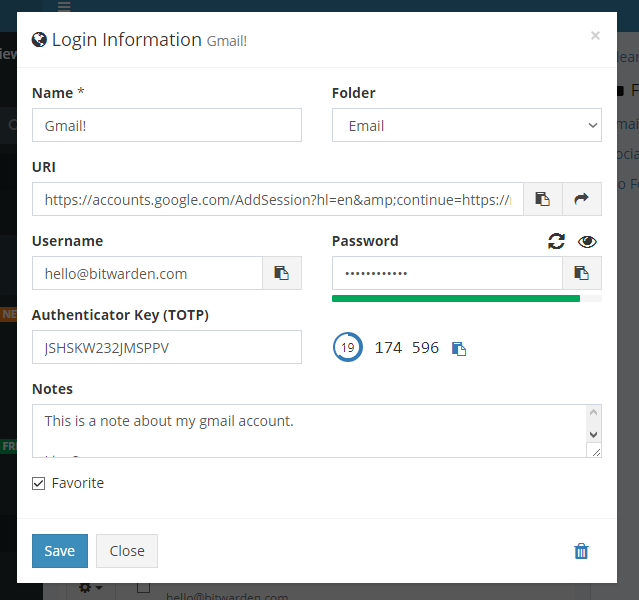


Figure 8 TOTP verification code displayed in the web vault login information view (Spearrin, 2017)

Another feature is that upon generating a password for login details, it also allows the user to generate a passphrase instead, this is an overlooked feature as users tend to think that a password is harder to remember is harder for a computer to guess, however, a passphrase which contains just dictionary words together could be stronger. The reason why a passphrase is strong is because in the attacker’s perspective if they knew for the user’s password consists of 4 words from a list of 2048 common words, the number of combinations to obtain this password would be (211)4 = 244 (with 11 calculated from log22048) (Explain XKCD, 2011), which would take a very long time to brute force.

A criticism of Bitwarden would be the UI, since Bitwarden is very simple, it doesn’t look as visually appealing which may be the reason why it isn’t as popular as some other password managers considering it doesn’t provide a visual “overview” of the user’s credentials together, although it was slightly compensated in the health reports for the premium plan.

## Similar applications – Dashlane

Dashlane is a popular password manager and it appears to be the most visually pleasing in comparison to the other password managers as it unifies the main elements in each section of the program in a readable fashion, and it very much seems like an upgraded password manager in comparison to Bitwarden as the layout is very similar and it stores the login details normally for each website. Below shows one of the sections in the program:

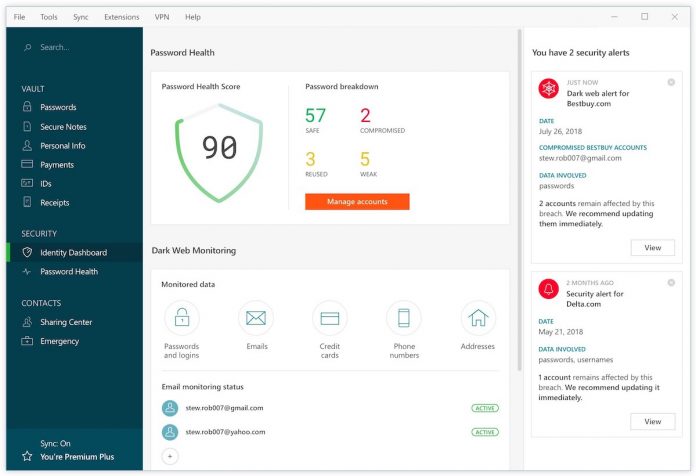


Figure 9 Dashlane identity dashboard (MacRumors, 2018)

Unlike the other password managers, Dashlane has a section dedicated just for an overview of the credentials which shows their weaknesses along with security alerts of breaches. This is very important as it highlights the result of using a password manager and the calculated “health score” provides visual assurance that they are secured which also encourages them constantly push this number up by improving their security.

A vital issue with Dashlane is that it’s not open source and it’s mostly a paid-to-use program as the free version is very limited, it also offers an in-built VPN on the premium plan, however, due to it not being open source, no one except the company can see how it operates behind the scenes. In the past there’s been many cases where such companies were caught selling data to other companies, like for an example the Hong Kong-based VPN provider ‘UFO VPN’ which was breached and had user logs leaked which can identify and track these users online activity (Bischoff, 2020). This would be even worse if it happens to be the case with Dashlane as it’s still a password manager at its core and so in the worst-case scenario these users would be leaving a money trail which makes them more vulnerable than if they weren’t using this system at all.

## Privacy and importance of compartmentalisation

The previously mentioned password managers had their strengths and weaknesses, although what they all lacked was the idea of securing the user’s privacy, where none of them had addressed the issues with the identities associated with the user’s credentials. The main aim of this system is to tackle is the issues of non-compartmentalisation online, and so it’s crucial to understand what the causes and effects of this are, as well as what could be done to minimise these effects in the future.

An online identity is simply a persona of a user, which comprises of who they are, what they do, and how they present themselves, which in other words, it’s the sum of a user’s characteristics and interactions online (Tan, 2016). These identities are constantly being tracked and used by big companies where data is harvested from them and they are targeted with ads to make them money, which in a way, is selling data, this was evident in a 2017 study of 144 million pages, it was revealed that 77% of those pages loaded with some tracker where Google was receiving data from 64% of these page loads (Eddy, 2018). While companies harvesting data may seem harmless, there are other aspects of online identities which can affect users, an example of such as a case is through exposed public information where a user may have posted something on social media which they regret, and this could damage their reputation and contribute to factors of getting a job in the future, in fact, an article written in “Forbes” revealed that in a survey conducted by CareerBuilder.com with 2,303 hiring managers that 65% of them incorporate social media into the hiring process sheerly to see if the job seeker presents himself professionally (Smith, 2013). A user’s activity online doesn’t necessarily need to relate to what they do in real life, however, it’s unavoidable when they use their same personal identity throughout all accounts.

Another case is in either a data breach or a compromised account, where in this instance its effect on the user way be significant, this is due to various ways in which a user’s data can be misused by an attacker, for instance, identity theft, where the attacker may have access to the user’s personal information such as their names and physical addresses, which may be due to poor privacy and security practices as seen from a survey conducted by Pew Research Center with 1,002 respondents revealed that only 13% would give inaccurate information about themselves to be less visible online (Rainie, et al., n.d.). Furthermore, these details can be linked with the user’s other accounts, making the user more vulnerable; to show how easy it is do this, assuming a user’s email addressed was exposed, it can be used to search through hundreds of breached websites using the site ‘haveibeenpwned.com’, from which an attacker can search for and find further information about the user, this is also known as ‘doxing’.

There are ways to minimise these issues, and the most important one is compartmentalisation; assuming the typical user works, and also studies, they would have a profile for each environment; one for work, another for study, and then another for social media and general browsing, ideally, these should be converted into identities where they do not overlap nor interfere with each other, and this is most commonly achieved by the use of pseudonyms as shown below:

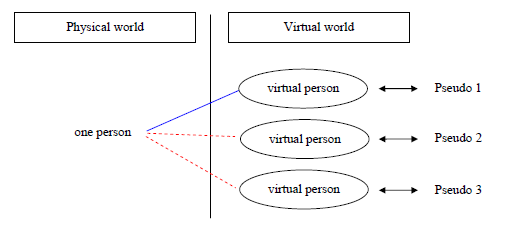


Figure 10 Linkable and unlinkable pseudonyms (WP2, 2008)

This shows demonstrates the idea of the interaction between physical and virtual world where one person may have several virtual identities, although they may be distinguished and unrelated using its pseudo details. These pseudo details may range from attributes such as fake names, temporary emails and fake passwords, all of which can be generated with ease, and this enhances the user’s privacy by providing anonymity where if a user was to sign up for a service or a website with one of these identities, their other identities are isolated away from it which also assists against targeting ads as they are based on these details, the figure below describes this isolation:

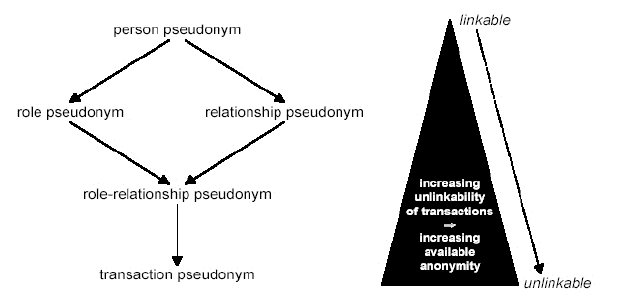


Figure 11 Linking pseudonyms (WP2, 2006)

The figure shows that increasing the number of pseudonyms behind a transaction for an example, increases the available anonymity, and thus the transaction becoming less linkable with the user’s real identity.

This most importantly minimises the effects of a compromised account or a data breach where the attacker would be unable to dig deeper into the user’s personal details as they are isolated by the identity created for that service or environment (work, social, etc.).

## Improved requirements gathered from background research

Throughout the background research, a list of requirements was formed covering some of the ways to make final system better by evaluating the existing solutions and unifying some of their best features as well as improvements:

* Use bcrypt to encrypt the master password and AES alongside to encrypt the credentials using the hashed master password.
* Allow the creation of identities and base the credentials around them by associating them with their respective identities.
* Provide a user-friendly UI that gives the user assurance through an overview displaying the user’s current strength of security and privacy.
* Use web crawlers to search for currently available public information in search engines, given the created online identities
* Allow the user to generate passphrases as an alternative to passwords for better security and memorisation.
* Ensure it is non-synchronised to prevent online threats to the user’s vaults by eliminating the use of servers and implementing a local master account.

## Requirements resulting from GDPR compliance

Considering this system is a password manager, it will involve the use and storage of the user’s personal data, therefore, the user is given control over this data to comply with the general data protection regulation using a set of rights obtained from ICO’s guide to GDPR (Information Commissioner's Office, 2018), this includes the following:

* Right of access – Display why and how their personal data will be used upon the actions they perform.
* Right to rectification – Enable the user to modify their inputs in the system which includes the credentials and other personal data.
* Right to erase or to be forgotten – Allow the option to delete any if not all the data inputted into the system, and this process must be easy to do.
* Right to restriction of processing – Provide the user with a choice in the processing of their data, which may be prevalent in the identity creation.
* Right to data portability – Implement the ability to export the user’s vaults at any given time in a formal manner.

The rights to withdraw consent and to object are handled by the rights to erase and restrict processing, as well as the right to related to automated decision making which is handled by the right to access, moreover, the right to lodge a complaint is not applied as this system is not intended to provide continuous customer support due to its offline usage.