Random secret based Hash Password Generators (RSHPG) and their advantages over Password Managers and Passkeys

Leonardo Balardini
Student,
tomopas602@gmail.com
https://github.com/tomopas

Abstract

Random secret based Hash Password Generators are able to generate passwords without relying on cloud storage. The user only needs to remember his master secret and a website secret to authenticate with a secure, irreversible password.

Keywords: Password, Random, Hash, Secret, Password Manager, Passkey.

1. Introduction

Secure passwords should only be used once per website but should also be long and random. Password Managers and Passkeys are two solutions to this problem, but they both have the disadvantage of storing data with a third party or having to deal with inconvenience. This is due to having to synchronize each new registration with the password managers database. RSHPG are a third solution that combines the advantages of both Password Managers and Passkeys while avoiding their disadvantages of needing a third party database.

2. Passkeys and Password Managers

Password Managers store and generate passwords for the user. They are secured with a master secret. They can either be stored in the cloud and automatically synchronized or stored on device which results in more inconvenience because the user has to synchronize new passwords manually.

Passkeys are a new technology which still isn't supported by all websites. They generate a private key for each website and authenticate by signing a challenge from the website with the users private key. The website verifies the signature with the users stored public key. The problem here is that there is also a need to synchronize the passkeys across devices which comes with the same inconveniences or security issues as password managers.

3. Methodology

Random secret based Hash Password Generators (RSHPG) solve the problem of relying on a third party. The RSHPG generates a password by combining a master secret, a random value and a website specific secret and then hashing all three values together. The resulting hash is then used as the password.

4. Example

- Master Secret: Password123 (the user should remember this and use it for every website, it should also be a secure long password which is easy for the user to remember)
- Random Value: fwh78GFWGF!g/f16f7w1fwDgw7f16f16f5wwf (the random value is generated once and stored on the device, the same random value is used for every website, this can be any compared to the private key in Passkeys but it is used for every website and not generated for each website)
- Website Secret: Google -> oo (the user chooses a pattern which is always the same for every website, this pattern generates a secret from the website information, for example the user can choose to always use the second and third letter of the websites name)
- Hash: oo | Password123 | Random Value -> Hash (the Website Secret, Master Secret and Random Value are combined and hashed together resulting in a password, a secure hashing function like SHA-256 should be used)

5. Comparison

Compared to Passkeys and Password Managers there is no need to store data on a third party. The Random Value can be shared with another device and used that way for example by using a QR code. The user only needs to remember the Master Secret and the Website Secret (the way he retrieves the Website Secret).

6. Pseudocode

```
MasterSecret = input("Enter your Master Secret: ")
RandomValue = device.get(RandomValue)
if RandomValue == None:
  RandomValue = generateRandomValue()
WebsiteSecret = input("Enter your Website Secret: ")

Hash = HashingFunction(WebsiteSecret + MasterSecret + RandomValue)
```

7. Criticism

The main disadvantage of RSHPG is that once a password is compromised, the user can't easily change it. If he would change it he would need to change the Website Secret Algorithm or the Master Secret or the Random Value, each of which would require a change of every other password used.

Another solution would be to use an additional password manager for websites which were once compromised making the RSHPG solution less convenient. This could be achieved by implementing an additional list for passwords which were compromised and will be hashed with an additional secret. A browser extension could then compare if a visited website is in the list of compromised websites and autofill the password from its password list like a password manager. This would still require the user to authenticate but would require synchronization between devices once a password is compromised.

A more convenient implementation would be for the website to send an additional website random value to the user which gets saved with the username and hashed password. This way if a password is compromised the website can tell the user to set a new password and generate a new website random value for the user. Also an attacker wouldn't know which accounts are active because the website could send a random value for each username entered even if an account doesn't exist and delete these credentials after a certain time. The attacker would thereby know if an account is fake or real after a certain time. This solution requires websites to implement a completely new authentication system which is not very convenient.

7.0.1 Web references

This Work will first be published on GitHub: https://github.com/tomopas/RSHPG including an open source Browser Extension implementation.

8. Conclusions

Random secret based Hash Password Generators (RSHPG) are a secure and convenient way to generate passwords without relying on third party storage. They combine the advantages of Password Managers and Passkeys while avoiding their disadvantages. RSHPG only require the user to remember a Master Secret and a Website Secret pattern, making them easy to use across multiple devices without the need for synchronization with a third party.