Sub-Optimal Cryptographic Wallets for Large Key State Spaces

Bashar Zahalka

June 2023

1 Abstract

In this study, we propose an improved approach for identifying sub-optimal cryptocurrency wallets within the context of a large number of cryptographic keys. Building upon the seminal work "On Cryptocurrency Wallet Design" by Ittay Eyal, we leverage their methodology and analysis as our main reference, tackling an open question raised in the referenced paper.

2 Introduction

The security of cryptocurrency assets relies on the secure storage of cryptographic keys by their owners. The unavailability of these keys can restrict asset access and potentially lead to attacks. The referenced paper provides significant insights into the wallets of up to 8 keys, which are of immediate importance to users in the blockchain technology field. We aim to extend these findings to accommodate the use of larger entities holding crypto assets, allowing for further scaling and applicability.

2.1 Cryptocurrency Wallets

A cryptocurrency wallet is a predicate that depends on the cryptographic keys held by the owner. For instance, if Alice has two keys, k1 and k2, one wallet configuration requires availability of either key (k1 OR k2) for access. While this allows Alice to access her wallet in a scenario where one key is lost or stolen, it also poses a risk of unauthorized access. Alternatively, a different configuration could demand both keys (k1 AND k2) for access. Although this mitigates unauthorized access if one key was lost, it presents a challenge for Alice as the loss of either key would render her unable to access the wallet.

2.2 The Model

In the model proposed in the referenced paper loss, theft, leakage, and safety are the possible states of cryptographic keys.

A Scenario is a vector representing the states of the keys the owner possesses. For example (SAFE, SAFE, LOSS, LEAK) is a scenario where k1 and k2 are safe, k3 is lost and k4 is leaked. If we assume that the safety, loss, theft, and leakage probabilities of each key are independent, we can calculate the probability of the scenario by considering the individual probabilities of each key.

A wallet is successful in a scenario if and only if the owner can access the wallet and the adversary cannot. We can calculate wallet success probability using the formula:

$$P_{success}(w) = \sum_{\sigma \in \Sigma^n} Pr[\sigma] \times 1_w(\sigma_O) \times 1_{\neg w}(\sigma_A)$$

$Pr[\sigma]$ - The probability of given scenario happening.
$1_w(\sigma_O)$ - 1 if the owner can access wallet w in given scenario, 0 otherwise.
$1_{\neg w}(\sigma_A)$ - 0 if the aversary can access wallet w in given scenario, 1 otherwise.

3 Methodology

The authors of the referenced paper employed a genetic algorithm to discover optimal cryptographic wallets. The algorithm begins with a random population of wallets and iteratively applies selection (choosing the fittest individuals) and perturbation (introducing mutations to the selected individuals' genome). The fitness function is utilized to determine the fittest individuals by evaluating their success probability. The wallets are perturbed by randomly adding or removing keys from the combinations of keys.

Evaluating the success probability of a wallet is computationally demanding, as it necessitates iterating through the entire scenario space. This limitation becomes more pronounced as the number of keys increases since the scenario space size is $4^{numberofkeys}$.

3.1 Our Approach

To address this computational challenge, we propose a lightweight fitness function for the genetic algorithm: heuristic success. Instead of calculating success probability over the entire scenario space, we calculate the success probability over a Uniformly At Random (UAR) selected sample of scenarios.

The new fitness function the algorithm uses would be:

$$H_{success}(w) = \sum_{\sigma \in \Sigma *} Pr[\sigma] \times 1_w(\sigma_O) \times 1_{\neg w}(\sigma_A)$$

Such that $\Sigma *$ is a UAR selected subset of Σ^n.

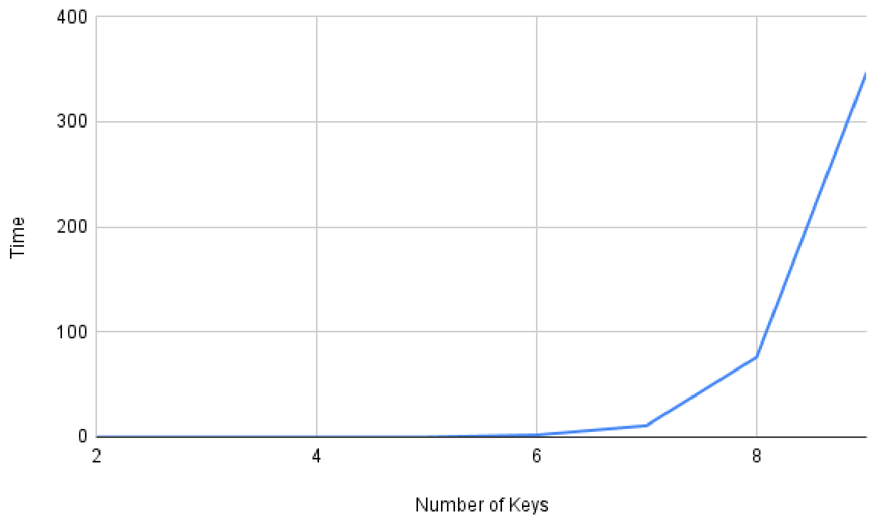
By introducing our improved fitness function, we have successfully enhanced the efficiency and scalability of the genetic algorithm for finding secure wallets. However, it is important to acknowledge the trade-off associated with this improvement. The resultant wallets generated by our approach may be sub-optimal compared to the original method, leading to lower success probabilities.

To address this limitation, we decided to incorporate random restarts into the algorithm. As a result, the success probabilities achieved by our approach became only marginally lower than those obtained by the original algorithm.

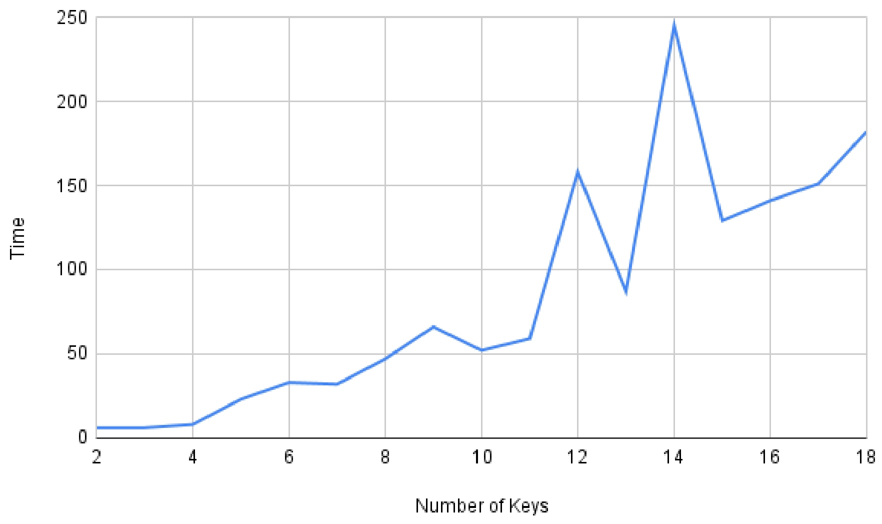
Despite this trade-off, the advantages of reduced computational burden and improved scalability outweigh the slight decrease in success probabilities.

4 Results

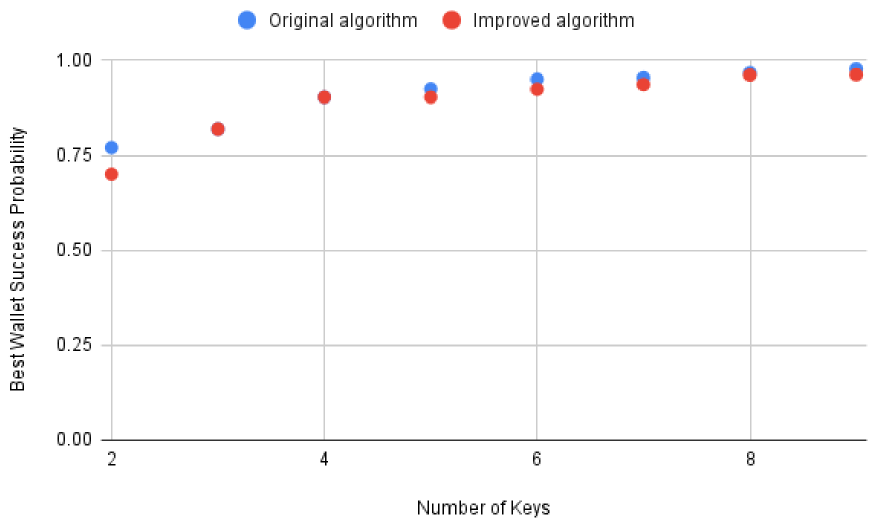
4.1 Original algorithm runtime:



4.2 Improved algorithm runtime:



4.3 Best wallet success probability:



Figures 4.1 and 4.2 illustrate the relationship between runtime and the number of keys for the original and improved algorithms, with 10 random restarts included in the runtime of the improved algorithm. Figure 4.3 demonstrates that the improved algorithm yields success probabilities that are comparable to those obtained by the original algorithm.