**Seminar**
Presentation

**Presented By:**
Mark Vainer

# Password Security and Rainbow Tables

# Password Storage

Storing passwords in a database in plain text makes the passwords vulnerable.

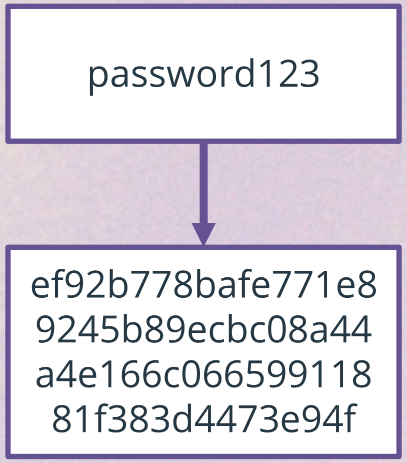
To protect them, a cryptographic hash function is used.

Some well-known hash functions include:

- MD5 (Message Digest 5) – widely known but now considered broken/insecure.
- SHA-1 (Secure Hash Algorithm 1) – also considered insecure for modern use.
- SHA-2 family (e.g., SHA-256, SHA-512) – widely used and secure today.
- SHA-3 – the latest NIST standard, based on the Keccak algorithm.
- BLAKE2 – fast, secure, and widely adopted.

And more...

Those are one-way functions meaning that it is very easy to compute the hash given the password but not so easy to reverse the process.



## Passwords Can be Cracked

Although cryptographic hash functions are hard to reverse, it is still possible to find out what the password is using indirect methods.

- Brute force attack
- Dictionary attack
- **Rainbow tables attack**

The last method is known to be the most time efficient method given a rainbow table exist. Prior to an attack, a rainbow table must be generated first.

# Rainbow Tables and Their Generation

A rainbow table is a special case of time-memory trade-off (TMTO) which is a case where an algorithm or program trades increased space usage with decreased time. Here, *space* refers to the data storage consumed in performing a given task (RAM, HDD, etc.), and *time* refers to the time consumed in performing a given task.

Rainbow tables are designed for a specific subset of passwords (specific character combinations, password lengths etc.).

Rainbow tables are essentially a large collection of chains generated by utilizing two types of functions: *cryptographic hash function* and a set of *reduction functions*.

| Hash Function | Reduction Function |
|---|---|
| Converts arbitrary length password to a fixed-length hash value | Converts a hash value to a new password candidate* |

## Rainbow Tables: How They Work

**Step 1: Obtain the hash**

- Attacker gains access to a stored password hash (e.g., from a database leak).

**Step 2: Table lookup**

- Attacker searches the rainbow table for a matching chain that contains the target hash.
- Instead of recomputing hashes from scratch, they use the precomputed table.

**Step 3: Trace backwards**

- Once the chain is identified, attacker reconstructs the chain from the start until the hash matches.
- The corresponding plaintext is revealed; this is the user's password.

## Let’s Crack Some Passwords...

Or

https://forms.cloud.microsoft/e/03F3zHKxbr?origin=lprLink

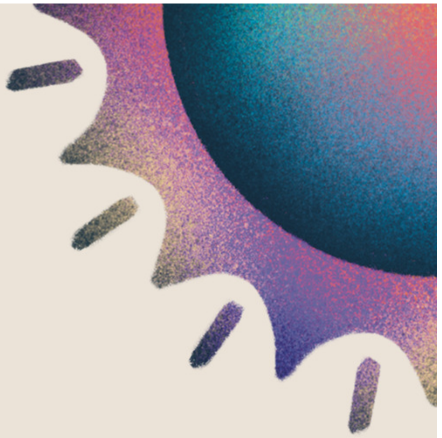
## The Problem with Rainbow Tables

For increased effectiveness, the rainbow table needs to be large to cover most of the search space. This requires a significant amount of time and computational power.

Naturally, parallel computing is perceived as an obvious way to increase computational capabilities. However, the selection of the efficient parallel algorithm is highly dependable on the research field, the considered problem and the method used.

**Some proposed solutions:**

1. Vainer, M., Kačeniauskas, A., & Goranin, N. (2025). Parallelization of rainbow tables generation using message passing Interface: A study on NTLMV2, MD5, SHA-256 and SHA-512 cryptographic hash functions. Applied Sciences, 15(15), 8152. https://doi.org/10.3390/app15158152
2. Li, P., Zhu, W., Chen, J., Yao, S., Hsu, C., & Xiong, G. (2023). High-speed implementation of rainbow table method on heterogeneous multi-device architecture. Future Generation Computer Systems, 143, 293–304. https://doi.org/10.1016/j.future.2023.01.022



## Main Results

Using Message Passing Interface (MPI)

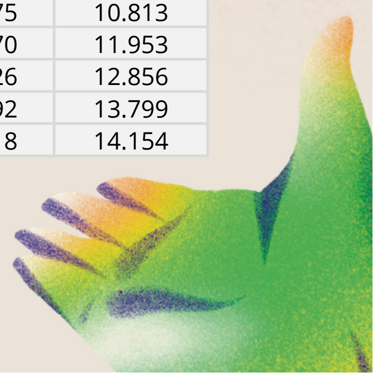
- High efficiency of 95–99%
- Near-linear speedup

Medium Table

| Nodes Count | SHA-256 | SHA-512 | MD5 | NTLMv2 |
|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 |
| **2** | 1.957 | 2.008 | 1.983 | 1.972 |
| **3** | 2.967 | 2.968 | 2.850 | 2.983 |
| **4** | 3.945 | 3.968 | 3.868 | 3.967 |
| **5** | 4.965 | 5.017 | 4.923 | 4.979 |
| **6** | 5.934 | 5.952 | 5.966 | 6.016 |
| **7** | 6.886 | 7.037 | 6.834 | 7.078 |
| **8** | 7.934 | 8.050 | 7.910 | 8.112 |
| **9** | 9.012 | 9.024 | 8.691 | 9.025 |
| **10** | 9.864 | 10.081 | 9.915 | 9.890 |
| **11** | 11.060 | 11.079 | 10.830 | 11.107 |
| **12** | 11.774 | 12.032 | 11.733 | 12.237 |
| **13** | 12.807 | 12.862 | 12.800 | 13.127 |
| **14** | 14.038 | 13.987 | 13.803 | 14.156 |
| **15** | 14.897 | 14.920 | 14.666 | 15.041 |

Large Table

| Nodes Count | SHA-256 | SHA-512 | MD5 | NTLMv2 |
|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 |
| **2** | 2.045 | 2.001 | 2.007 | 1.970 |
| **3** | 3.029 | 2.997 | 2.998 | 2.924 |
| **4** | 3.857 | 3.953 | 3.996 | 3.921 |
| **5** | 5.073 | 5.035 | 4.981 | 4.984 |
| **6** | 5.925 | 6.030 | 5.895 | 5.981 |
| **7** | 7.104 | 7.075 | 7.021 | 6.973 |
| **8** | 8.146 | 8.037 | 8.070 | 7.854 |
| **9** | 8.959 | 9.077 | 8.884 | 8.835 |
| **10** | 10.134 | 10.034 | 10.047 | 9.983 |
| **11** | 11.222 | 11.095 | 10.875 | 10.813 |
| **12** | 11.679 | 11.983 | 11.870 | 11.953 |
| **13** | 13.147 | 12.745 | 12.626 | 12.856 |
| **14** | 14.063 | 13.983 | 13.692 | 13.799 |
| **15** | 14.715 | 15.012 | 14.618 | 14.154 |



## Implications of Quantum Computing

**There are two sides to a coin:**

***Quantum-enhanced rainbow table attacks:***

Quan et al. (2024) and Khajeian (2025) proposed improvements to rainbow table attacks using quantum concepts and algorithms (Grover's Algorithm).

1. Quan, L. J., Ye, T. J., Ling, G. G., & Balachandran, V. (2024). QIris: Quantum Implementation of Rainbow Table Attacks. arXiv.org. https://arxiv.org/abs/2408.07032
2. Khajeian, M. (2025). A hybrid Classical-Quantum rainbow table attack on human passwords. arXiv.org. https://doi.org/10.22059/jac.2025.398812.1237

***Threat to rainbow table attacks?***

A quantum computer could potentially brute-force faster then rainbow table generation but due to the current limitations this is not practical today

## Thank You for Listening!

Q&A

Mark Vainer

mark.vainer@vilniustech.lt