A multi-round bilinear-map-based secure password hashing scheme

Csanád Bertók, Andrea Huszti, Tamás Kádek, Zsanett Jámbor

University of Debrecen, Faculty of Informatics

IEEE 2nd Conference on Information Technology and Data Science

Debrecen, 2022
Topics of the presentation

1. Introduction
2. Off-line attacks
3. Preliminaries
4. The proposed scheme
5. Security and efficiency analysis
Password usage

- Authentication: password (+salt) $\rightarrow$ hash (+salt)
- Key generation: PAKE, PBKDF

Multiple attacks against weak or not correctly stored passwords.

- 1Password (2017)
- Passwordstate (2021)
Our contribution

To protect users, services, ... against attacks, several password hashing schemes/functions have been proposed and used.

- PBKDFv2
- Argon2 (winner of PHC 2015)
- bcrypt

We construct a secure PHS based on bilinear pairing with the following properties:

- Multi-round
- Adjustable cost factor
Off-line attacks

- (Mostly) salt (and hash) independent attacks
  - Brute force
  - Dictionary

- Attacks against hashes (mostly mitigated by salt)
  - Rainbow-tables

\[
p_{i,1} \xrightarrow{H} c_{i,1} \xrightarrow{R} p_{i,2} \xrightarrow{H} c_{i,2} \xrightarrow{R} p_{i,3} \rightarrow \cdots \rightarrow p_{i,k} \xrightarrow{H} c_{i,k}
\]
Admissible bilinear map

Let $\mathbb{G}$ be an additive and $\mathbb{G}_T$ a multiplicative group of order $p$ for some large prime $p$. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an admissible bilinear map if it satisfies the following properties:

1. **Bilinear:** We say that a map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

2. **Non-degenerate:** The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$. Since $\mathbb{G}, \mathbb{G}_T$ are groups of prime order, if $P$ is a generator of $\mathbb{G}$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_T$.

3. **Computable:** There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}$. 

---
For elliptic curve based cryptography usually

- $\mathbb{G}$ is an elliptic curve group (a subgroup of the $r$-torsion)
- $\mathbb{G}_T$ is the roots of unity in a finite field

Associated problem:

**Computational Diffie-Hellman Problem**

Let $\mathbb{G}$ be a cyclic group with generator $G \in \mathbb{G}$ and let $xG, yG \in \mathbb{G}$. The Computational Diffie-Hellman Problem is to compute $xyG$. 
Preliminaries

Mapping into elliptic curves

- $q \equiv 3 \pmod{4}$ prime
- $E : y^2 = x^3 + ax$ over $\mathbb{Z}_q$

$$tr : \mathbb{Z}_q \longrightarrow E(\mathbb{Z}_q)$$

$$x \mapsto \left( \varepsilon(x) \cdot x, \varepsilon(x) \sqrt{\varepsilon(x) \cdot (x^3 + ax)} \right),$$

where $\sqrt{\cdot}$ is the square root over $\mathbb{Z}_q$ and $\varepsilon(x) = \left( \frac{x^3 + ax}{q} \right)$, where $\left( \frac{\cdot}{q} \right)$ is the Legendre symbol.
The proposed scheme

Requirements based on PHC

- Password length between 0 and 128 bytes
- Salt length 16 bytes
- Output length minimum 32 bytes
- Configurable time and/or memory cost

Our algorithm fulfills all the criteria, the configurable parameter is the time \( (t_{\text{cost}}) \) which can be adjusted by increasing / decreasing the number of rounds.
The proposed scheme

Algorithm The proposed algorithm

**INPUT:** password

**OUTPUT:** PswStore, \( S \)

1: Initialize \( E(\mathbb{Z}_q) \)

2: Initialize \( S \)

3: \( PswStore \leftarrow \text{Convert}(\text{password}) \)

4: for \( i = 0 \) up to number of rounds do

5: \( R \leftarrow \text{hashToCurve}(PswStore) \)

6: \( PswStore \leftarrow \text{TatePairing}(R, S + iG) \)

7: \( PswStore \leftarrow \text{Convert}(PswStore) \)

return \( (PswStore, S) \)
The following security requirements were considered:

- Pre-image resistance (bilinear pairing is one-way)
- Second pre-image resistance
- Collision resistance

Pre-image resistance

Let \( \hat{e} : G \times G \rightarrow G_T \) be a bilinear map. Let \( \langle G \rangle = G \) and \( \langle g \rangle = G_T \) be any elements such that \( \hat{e}(G, G) = g \). If the CDH problem is infeasible for \( g, g^a, g^b \in G_T \) with any \( a, b \in \mathbb{Z}_q \), then \( \hat{e} \) is a one-way pairing.

Thus CHD hard \( \implies \) pre image resistance.
Collision resistance $\implies$ second pre-image resistance

<table>
<thead>
<tr>
<th>Collision resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Bilinear pairing considered over torsion groups of $E$</td>
</tr>
<tr>
<td>■ The $r$-torsion has $r + 1$ cyclic groups</td>
</tr>
<tr>
<td>■ Same subgroup $\implies$ same result</td>
</tr>
<tr>
<td>■ Probability of collision for our curve and prime $\sim 10^{-48}$</td>
</tr>
</tbody>
</table>
## Efficiency analysis - running time

Comparing with bcrypt and RSA (running time measured in seconds)

<table>
<thead>
<tr>
<th># of rounds</th>
<th>bcrypt</th>
<th>Our algorithm</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.0030458</td>
<td>0.5346845</td>
<td>0.009764</td>
</tr>
<tr>
<td>32</td>
<td>0.0037977</td>
<td>0.8726219</td>
<td>0.0090346</td>
</tr>
<tr>
<td>64</td>
<td>0.0069453</td>
<td>1.7379774</td>
<td>0.0251674</td>
</tr>
<tr>
<td>128</td>
<td>0.0130193</td>
<td>1.5386831</td>
<td>0.0334561</td>
</tr>
<tr>
<td>256</td>
<td>0.023243</td>
<td>3.5953085</td>
<td>0.0638214</td>
</tr>
<tr>
<td>512</td>
<td>0.0431535</td>
<td>5.3515215</td>
<td>0.1371731</td>
</tr>
<tr>
<td>1024</td>
<td>0.087049</td>
<td>10.3966082</td>
<td>0.2013071</td>
</tr>
<tr>
<td>2048</td>
<td>0.167253</td>
<td>20.9222832</td>
<td>0.452279</td>
</tr>
<tr>
<td>4096</td>
<td>0.3439718</td>
<td>46.5067361</td>
<td>0.7515071</td>
</tr>
<tr>
<td>8192</td>
<td>0.6667411</td>
<td>86.7408044</td>
<td>1.3365767</td>
</tr>
</tbody>
</table>
Efficiency analysis - memory usage, LoC

Memory usage - limited to 1 second of runtime

Python memory profiler module

- Argon2 → 20, 1 MiB
- bcrypt → 20, 2 MiB
- Our algorithm → 22, 0 MiB

For the number of lines of code (LoC) our algorithm is between bcrypt and Argon2, however this is not a factor which can be measured precisely.
THANK YOU FOR YOUR ATTENTION!