

# Update on Argon2

Dmitry Khovratovich

University of Luxembourg

Evernym, Inc.

April 30th, 2017

Recall why we need Argon2

Keyless password authentication:

- User registers with name  $I$  and password  $p$ ;
- Server selects hash function  $H$ , generates salt  $s$ , and stores  $(I, H(s, p))$ ;
- User sends  $(I, p')$  during the login;
- Server matches  $(I, H(s, p'))$  with its password file.

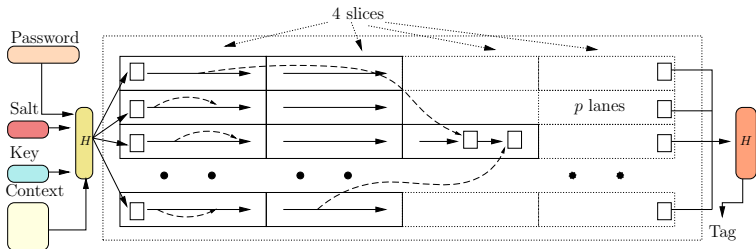
Problems:

- Password files are often leaked unencrypted;
- Passwords have low entropy ("123456");
- Regular cryptographic hash functions are cracked on GPU/FPGA/ASIC.



# Argon2, the winner of Password Hashing Competition

# Specification of Argon2



Three variants: Argon2d, Argon2i, Argon2id.

- Select the amount of memory  $M$ , number of passes  $T$ , level of parallelism  $l$ .
- Argon2d uses data-dependent addressing – side-channels;
- Argon2i uses data-independent addressing – lower attack costs due to tradeoffs (up to factor 4);
- Argon2id, best of two worlds – Argon2i for the first half-pass, Argon2d for the rest.

# Updates on Argon2 since July 2016

## Analysis:

- No new attack, only a slight improvement in the Alwen-Blocki attack on Argon2i (see later).

## Code:

- 65 commits, mostly refactoring and bug fixes;
- 112 total forks, 1567 stars.
- Bindings: Javascript, PHP, Python, Rust, Go.

## Adoption:

- Packages: Debian, Ubuntu, NetBSD,...
- Projects: EXT4/fscrypt filesystem, Yandex authentication.
- Libraries: libsodium, PassLib.
- Password managers: KeePass;
- Web frameworks: Django;
- Proof of work: Dynamic, Lemon Coin.

On the road: Mozilla, Dropbox, DOVECOT.

Time to wrap up and finalize Argon2  
as a standard!

Which parameter set is the best?

Naive approach 1:

- Fix affordable memory  $M$ , take  $T$  passes that minimizes the impact of the tradeoff attack.

Naive approach 2:

- Fix affordable time  $MT$ , take  $T$  that minimizes the impact of the tradeoff attack.



Which parameter set is the best?

Naive approach 1:

- Fix affordable memory  $M$ , take  $T$  passes that minimizes the impact of the tradeoff attack.

Naive approach 2:

- Fix affordable time  $MT$ , take  $T$  that minimizes the impact of the tradeoff attack.

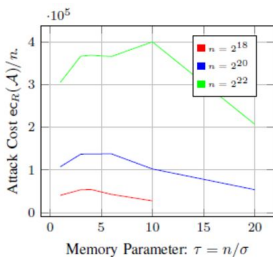
Best choice:

- Fix time  $MT$ , take  $(M, T)$  that maximizes the attack cost.

Costs of hash evaluation using  $M$  memory and  $T$  passes on hardware (time-area product) using a time-memory tradeoff of quality  $Q(M, T)$ :

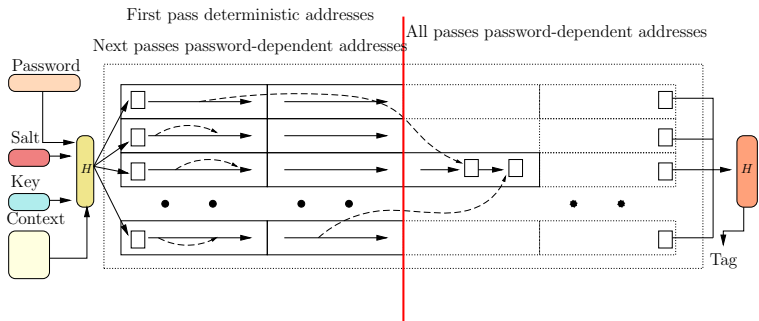
$$\frac{M^2 \cdot T}{Q(M, T)}$$

Best tradeoff attacks by Alwen-Blocki:



The  $T = 3$  is close to optimal.

# Argon2id



- Side-channels are possible after the half of the first pass only;
- Smart time-memory tradeoffs apply to the first half only.

Tradeoff attack on  $T = 1$ :

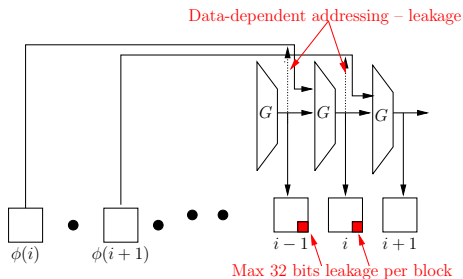
- Can not do better than attack the first half as Argon2i and the second half as Argon2d – total factor less than 2.5;
- No attack on 2 and more passes.

Tradeoff attack on  $T = 1$ :

- Can not do better than attack the first half as Argon2i and the second half as Argon2d – total factor less than 2.5;
- No attack on 2 and more passes.

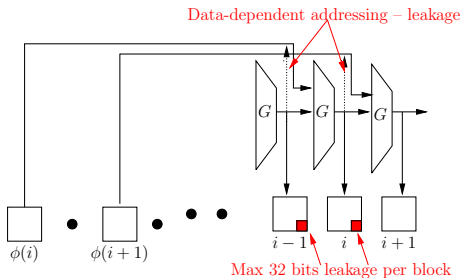
What can we infer from the side-channel analysis?

Blocki (2017): suppose that the adversary has learned all data-dependent memory addresses.



Can we learn information on  $B[\phi(i)]$  given the output of  $G$ , a wide Blake2b-based permutation?

Blocki (2017): suppose that the adversary has learned all data-dependent memory addresses.



Can we learn information on  $B[\phi(i)]$  given the output of  $G$ , a wide Blake2b-based permutation? Our experiments show no statistically significant correlation.



- ① Argon2id is the primary variant;
- ②  $T = 1$  pass is recommended.

Questions?

If no, approve the RFC draft.