What's Up Argon2?

The Password Hashing Winner A Year Later

JP Aumasson, Kudelski Security

Password Hashing Competition

and our recommendation for hashing passwords: Argon2

ARGON2 | PHC | CONTACT

Password hashing is everywhere, from web services' credentials storage to mobile and desktop authentication or disk encryption systems. Yet there wasn't an established standard to fulfill the needs of modern applications and to best protect against attackers. We started the Password Hashing Competition (PHC) to solve this problem.

PHC ran from 2013 to 2015 as an open competition—the same kind of process as NIST's AES and SHA-3 competitions, and the most effective way to develop a crypto standard. We received 24 candidates, including many excellent designs, and selected one winner, Argon2, an algorithm designed by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich from University of Luxembourg.

password-hashing.net

Nobody cared about password hashing research before PHC

Now we've got **Argon2**, the best password hash ever

Secure, simple, easy to use

Argon2: the memory-hard function for password hashing and other applications

Designers: Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich
University of Luxembourg, Luxembourg

How Argon2 works, super high-level

1. H := **Hash**(password, salt, all parameters)
2. Fill a 2-dimension array B of ***MemParameter*** 1024-byte **blocks**
   - Fill column by column, with sequential dependency
   - Blocks B[i][0] and B[i][1] depend on H
   - Other blocks B[i][j] depend on B[i][j–1] and on **another block**
   - "depend on X" = "are a BLAKE2-based **hash** of stuff including X"
3. Repeat 2 ***TimeParameter*** times, **xoring** new blocks to old ones
4. Return as a **tag** an xor of the last column’s blocks

- **Argon2d**: “another block” **d**epends on the password
- **Argon2i**: “another block” is **i**ndependent of the password

Side-channel info on “another block” can be used to crack passwords faster ⇒ use **Argon2i** if there are side channels

But **Argon2d** gets you optimal resistance to TMTO

Processor	Threads	Argon2d (1 pass)		Argon2i (3 passes)	
		Cycles/Byte	Bandwidth (GB/s)	Cycles/Byte	Bandwidth (GB/s)
i7-4500U	1	1.3	2.5	4.7	2.6
i7-4500U	2	0.9	3.8	2.8	4.5
i7-4500U	4	0.6	5.4	2	5.4
i7-4500U	8	0.6	5.4	1.9	5.8

Table 4: Speed and memory bandwidth of Argon2(d/i) measured on 1 GB memory filled. Core i7-4500U — Intel Haswell 1.8 GHz, 4 cores

Specifically, on an i7-4500U (Haswell):

- 0.1 second to **Argon2d** using 250MB with 1 core
- 0.5 second to **Argon2i** using 1GB with 2 cores

Applications of Argon2

- Storing user **passwords**
- **Key derivation**, from low-entropy data like passwords
- **Proofs of work** (there's already an altcoin)

Get it at https://github.com/P-H-C/phc-winner-argon2

- Reference C89 code, for Linux, *BSD, Windows
- Builds static and shared libs, command-line utility
- Public domain-like license (CC0)
- Bindings for most common languages

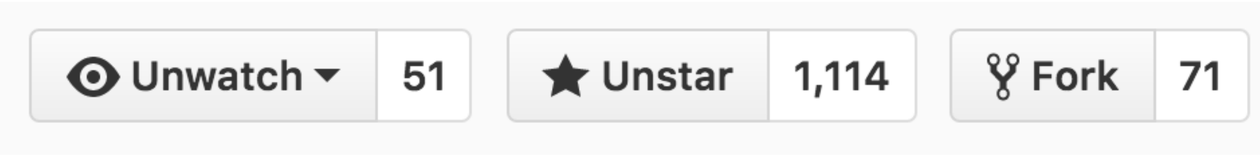
```
$ echo -n "password" | ./argon2 somesalt -t 2 -m 16 -p 4 -l 24
Type:           Argon2i
Iterations:     2
Memory:         65536 KiB
Parallelism:    4
Hash:           45d7ac72e76f242b20b77b9bf9bf9d5915894e669a24e6c6
Encoded:        $argon2i$v=19$m=65536,t=2,p=4$c29tZXNhbHQ$RdescudvJCsgt3ub+b+dWRWJTmaaJObG
0.188 seconds
Verification ok
```

Based on initial C++ code by the Argon2 designers

Initial commit

veorq committed on Oct 4, 2015

Since then, as of Jul 25:



- 463 commits, 91 pull requests, 58 issues
- Major code cleanup and lots of bugs fixed
- Continuous integration and best practices

Thanks to all contributors

sneves
49 commits / 2,182 ++ / 2,508 --

khovratovich
39 commits / 552 ++ / 392 --

daniel-dinu
36 commits / 51,965 ++ / 25,077 --

technion
32 commits / 354 ++ / 154 --

flamewow
13 commits / 536 ++ / 434 --

lucab
11 commits / 585 ++ / 562 --

jedisct1
9 commits / 29 ++ / 16 --

ranisalt
8 commits / 116 ++ / 81 --

UniQP
4 commits / 4 ++ / 6 --

phxql
2 commits / 10 ++ / 6 --

0-wiz-0
2 commits / 6 ++ / 1 --

paragonie-scott
2 commits / 20 ++ / 10 --

yonas
2 commits / 2 ++ / 6 --

hynek
2 commits / 6 ++ / 2 --

mbroz
2 commits / 8 ++ / 31 --

ocharles
1 commit / 1 ++ / 0 --

angt
1 commit / 1 ++ / 1 --

quininer
1 commit / 1 ++ / 0 --

dessant
1 commit / 1 ++ / 0 --

cjlarose
1 commit / 2 ++ / 1 --

tvdburgt
1 commit / 1 ++ / 0 --

aberaud
1 commit / 1 ++ / 1 --

alipha
1 commit / 51 ++ / 16 --

dkg
1 commit / 47 ++ / 0 --

thibaultCha
1 commit / 1 ++ / 0 --

tigertoes
1 commit / 0 ++ / 1 --

Khady
1 commit / 1 ++ / 0 --

seanhussey
1 commit / 1 ++ / 1 --

PlasmaPower
1 commit / 2 ++ / 0 --

Qowyn
1 commit / 1,202 ++ / 10 --

The default password hash in **libsodium**

```
#define PASSWORD "Correct Horse Battery Staple"
#define KEY_LEN crypto_box_SEEDBYTES

unsigned char salt[crypto_pwhash_SALTBYTES];
unsigned char key[KEY_LEN];

randombytes_buf(salt, sizeof salt);

if (crypto_pwhash
    (key, sizeof key, PASSWORD, strlen(PASSWORD), salt,
     crypto_pwhash_OPSLIMIT_INTERACTIVE, crypto_pwhash_MEMLIMIT_INTERACTIVE,
     crypto_pwhash_ALG_DEFAULT) != 0) {
    /* out of memory */
}
```

https://download.libsodium.org, by @jedisct1

Source Package: argon2 (0~20160406-2)

The following binary packages are built from this source package:

argon2

memory-hard hashing function - utility

libargon2-0

memory-hard hashing function - runtime library

libargon2-0-dev

memory-hard hashing function - development files

Props to @lucabruno

Why Argon2 and not **scrypt**?

- Scrypt has no data-independent mode (like Argon2i)
- Argon2 is easier to parametrize (just 2 knobs)
- Argon2 algorithm is simpler
  - scrypt needs PKBDF2, HMAC, SHA-256, Salsa20
  - Argon2 just needs BLAKE2-like rounds

Argon2 also has a better **security analysis** …

Argon2's security (1/4): **cryptanalysis**

- Seriously? :-)

Argon2's security (2/4): **GPU/ASIC inefficiency**

- Argon2 optimized for modern x86 microarchitectures
- Exploits local parallelism and multi-core/threading
- More memory usage makes ASICs slower & costlier

Argon2's security (3/4): **side-channel resistance**

- We're concerned with **software** side channels
- Argon2i is time-constant, memory addresses-constant
- Argon2d is not

Argon2's security (4/4): **time-space tradeoffs**

- How much does it cost to hash with less memory?
- There should be no "shortcut"
- 2 excellent papers published this year…

http://eprint.iacr.org/2016/027 (Jan 2016, 53 pages)

Balloon Hashing: a Provably Memory-Hard Function with a Data-Independent Access Pattern

Dan Boneh[1], Henry Corrigan-Gibbs[1], and Stuart Schechter[2]

[1] Stanford University
[2] Microsoft Research

Rigorous analysis of memory-hard hashing

- Introduced the **balloon hashing** function
- Showed how to Argon2i with **4 times less space**
- Motivated a **tweak** of Argon2i released March 2016

http://eprint.iacr.org/2016/115 (Feb 2016, 37 pages)

Efficiently Computing Data-Independent Memory-Hard Functions

Joël Alwen
IST Austria

Jeremiah Blocki
Microsoft Research

Theoretical analysis of memory-hard hashing's cost:

- Introduces an energy measure, more realistic than AT
- Presents **asymptotic** attacks on Argon2i and Balloon
- No practical impact on Argon2, similar attacks known

Standardization efforts @ IRTF (CFRG)

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 21, 2016

A. Biryukov
D. Dinu
D. Khovratovich
University of Luxembourg
S. Josefsson
SJD AB
March 20, 2016

The memory-hard Argon2 password hash and proof-of-work function
draft-irtf-cfrg-argon2-00

Abstract

This document describes the Argon2 memory-hard function for password hashing and proof-of-work applications. We provide an implementer oriented description together with sample code and test vectors. The purpose is to simplify adoption of Argon2 for Internet protocols.

Conclusions

- We understand well Argon2's strengths and limitations
- Argon2 now has a mature reference implementation
- You can use it with most popular languages

For any support: http://password-hashing.net/#contact