

SHA512 vs. Blowfish and Bcrypt

[closed]

Asked 15 years, 2 months ago Modified 6 years, 5 months ago

Viewed 143k times



236



As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, [visit the help center](#) for guidance.

Closed 12 years ago.

I'm looking at hashing algorithms, but couldn't find an answer.

- Bcrypt uses Blowfish
- Blowfish is better than MD5
- Q: but is Blowfish better than SHA512?

Thanks..

Update:

I want to clarify that I understand the difference between hashing and encryption. What prompted me to ask the

question this way is [this article](#), where the author refers to bcrypt as "adaptive hashing"

Since bcrypt is based on Blowfish, I was led to think that Blowfish is a hashing algorithm. If it's encryption as answers have pointed out, then seems to me like it shouldn't have a place in this article. What's worse is that he's concluding that bcrypt is the best. What's also confusing me now is that the phpass class (used for password hashing I believe) uses bcrypt (i.e. blowfish, i.e. encryption). Based on this new info you guys are telling me (blowfish is encryption), this class sounds wrong. Am I missing something?

security

encryption

passwords

hash

Share

Improve this question

Follow

edited Jul 21, 2018 at 8:44



Peter Vandivier

671 ● 1 ● 10 ● 37

asked Oct 13, 2009 at 15:56



Chris

8,816 ● 18 ● 51 ● 56

2 It's not wrong; see the updates to my answer for an explanation of how bcrypt works and why it serves the same purpose as a hash-based "one-way" algorithm. – [erickson](#)
Oct 13, 2009 at 21:21

3 `bcrypt` simply has a higher "work factor" by default. SHA is assumed not to...unless if you use passhash9, which can use

either along with a work factor. why is this question closed?
it's far from answered yet very important. – user1382306 Oct 3, 2013 at 5:11

1 Link in question is down..... – [Pacerier](#) Dec 22, 2014 at 0:36

5 Answers

Sorted by:

Highest score (default)



333



It should suffice to say whether bcrypt or SHA-512 (in the context of an appropriate algorithm like PBKDF2) is *good enough*. And the answer is yes, either algorithm is secure enough that a breach will occur through an implementation flaw, not cryptanalysis.



If you insist on knowing which is "better", SHA-512 has had in-depth reviews by NIST and others. It's good, but flaws have been recognized that, while not exploitable now, have led to the the SHA-3 competition for new hash algorithms. Also, keep in mind that the study of hash algorithms is "newer" than that of ciphers, and cryptographers are still learning about them.

Even though bcrypt as a whole hasn't had as much scrutiny as Blowfish itself, I believe that being based on a cipher with a well-understood structure gives it some inherent security that hash-based authentication lacks. Also, it is easier to use common GPUs as a tool for attacking SHA-2–based hashes; because of its memory requirements, optimizing bcrypt requires more specialized hardware like FPGA with some on-board RAM.

Note: bcrypt is an algorithm that uses Blowfish internally. It is not an encryption algorithm itself. It is used to irreversibly obscure passwords, just as hash functions are used to do a "one-way hash".

Cryptographic hash algorithms are designed to be impossible to reverse. In other words, given only the output of a hash function, it should take "forever" to find a message that will produce the same hash output. In fact, it should be computationally infeasible to find any two messages that produce the same hash value. Unlike a cipher, hash functions aren't parameterized with a key; the same input will always produce the same output.

If someone provides a password that hashes to the value stored in the password table, they are authenticated. In particular, because of the irreversibility of the hash function, it's assumed that the user isn't an attacker that got hold of the hash and reversed it to find a working password.

Now consider bcrypt. It uses Blowfish to encrypt a magic string, using a key "derived" from the password. Later, when a user enters a password, the key is derived again, and if the ciphertext produced by encrypting with that key matches the stored ciphertext, the user is authenticated. The ciphertext is stored in the "password" table, but the derived key is never stored.

In order to break the cryptography here, an attacker would have to recover the key from the ciphertext. This is called a "known-plaintext" attack, since the attack knows

the magic string that has been encrypted, but not the key used. Blowfish has been studied extensively, and no attacks are yet known that would allow an attacker to find the key with a single known plaintext.

So, just like irreversible algorithms based cryptographic digests, bcrypt produces an irreversible output, from a password, salt, and cost factor. Its strength lies in Blowfish's resistance to known plaintext attacks, which is analogous to a "first pre-image attack" on a digest algorithm. Since it can be used *in place of a hash algorithm* to protect passwords, bcrypt is confusingly referred to as a "hash" algorithm itself.

Assuming that rainbow tables have been thwarted by proper use of salt, any truly irreversible function reduces the attacker to trial-and-error. And the rate that the attacker can make trials is determined by the speed of that irreversible "hash" algorithm. If a single iteration of a hash function is used, an attacker can make millions of trials per second using equipment that costs on the order of \$1000, testing all passwords up to 8 characters long in a few months.

If however, the digest output is "fed back" thousands of times, it will take hundreds of years to test the same set of passwords on that hardware. Bcrypt achieves the same "key strengthening" effect by iterating inside its key derivation routine, and a proper hash-based method like PBKDF2 does the same thing; in this respect, the two methods are similar.

So, my recommendation of bcrypt stems from the assumptions 1) that a Blowfish has had a similar level of scrutiny as the SHA-2 family of hash functions, and 2) that cryptanalytic methods for ciphers are better developed than those for hash functions.

Share Improve this answer

edited Dec 23, 2013 at 17:29

Follow

answered Oct 13, 2009 at 16:09



erickson

269k ● 59 ● 401 ● 497


4 +1 great post. But I have two questions. Blowfish was replaced by twofish more than a decade ago, shouldn't a system utilize modern primitives? Also thousands of iterations seems wasteful in systems such as web applications where many people are logging in at any given moment. For instance PBKDF2 is only implemented in scenarios when 1 person is logging on at a time, such as a string2key function for an encrypted filesystem. I use the adage "If its too heavy for the attacker to lift, then its too heavy for your server." What do you think? – rook Nov 6, 2010 at 4:25

17 I don't think there's anything wrong with using a more modern primitive. Vulnerabilities are often discovered with the passage of time, and Twofish was developed using knowledge gained from Blowfish. However, I'm not aware of specific vulnerabilities that would invalidate use of Blowfish, so a "if-it-ain't-broke" argument could be made too. Your adage about attackers doesn't sound good to me. Even if you choose an algorithm that would require years for an attacker to test a billion passwords, it will consume a

negligible fraction of time in a legitimate application.

– [erickson](#) Nov 8, 2010 at 4:40

15 If you look at the specification of any hash function, you will not see anything about "salt". The only parameter is the message to be digested. Review the specification of any cipher, and you will see that the function is parameterized with a key. The "salt" which may (or may *not*) be used in conjunction with a hash is simply part of the message. The hash algorithm doesn't require it, doesn't treat it specially, and cannot differentiate it from the rest of the message. So, while it is true that *messages are often altered* by salting, a given message produces only one hash. – [erickson](#) Jan 14, 2011 at 19:33

1 @Andre D As a pentester I report applications that lock accounts, and i Report applications that do not prevent against brute force. Ideally an offending IP address must solve a captcha, additionally if a user name is targeted (even if that username doesn't exists) then that account should solve a captcha before authenticating. Enforcing a ratelimit of X per minute is also acceptable. Related: [security.stackexchange.com/questions/25444/...](https://security.stackexchange.com/questions/25444/) – [rook](#) May 23, 2013 at 0:16 

2 @rook : while ratelimiting applications is good practice you may assume in this case that the database has been downloaded and placed on equipment that does not have the rate limit you describe. – [Ellert van Koperen](#) Oct 21, 2015 at 9:06



50

I agree with erickson's answer, with one caveat: for password authentication purposes, bcrypt is *far* better than a *single iteration* of SHA-512 - simply because it is far slower. If you don't get why slowness is an advantage in this particular game, read the article you linked to again



(scroll down to "*Speed is exactly what you don't want in a password hash function.*").



You can of course build a secure password hashing algorithm around SHA-512 by iterating it thousands of times, just like the way PHK's MD5 algorithm works.

[Ulrich Drepper did exactly this](#), for glibc's crypt(). There's no particular reason to do this, though, if you already have a tested bcrypt implementation available.

Share Improve this answer

answered Oct 13, 2009 at 23:12

Follow




caf

239k ● 41 ● 335 ● 474

-
- 3 Hopefully my answer makes it clear that a single hash iteration is not sufficient (sadly, even this rudimentary level of knowledge cannot be assumed). "If a single iteration of a hash function is used, an attacker can make millions of trials per second using equipment that costs on the order of \$1000, testing all passwords up to 8 characters long in a few months. If however, the digest output is 'fed back' thousands of times, it will take hundreds of years to test the same set of passwords on that hardware. Bcrypt achieves the same 'key strengthening' effect by iterating..." – [erickson](#) Jun 20, 2011 at 14:16

@erickson: Yes, although I think you might have buried the lede there. The point I was trying to make is that a direct comparison of bcrypt and SHA-512 isn't really relevant, because one is a key derivation function and the other is just a cryptographic primitive, unsuitable on its own. – [caf](#) Jun 21, 2011 at 0:05

-
- 4 codahale.com/how-to-safely-store-a-password – [ripper234](#) Feb 1, 2012 at 12:38
-

- 1 Using thousands of rounds of SHA-512 is not unheard of and given its inclusion in various `crypt` implementations (including in PHP which I use), when I read the original question I even assumed that's what the OP meant when he asked about SHA-512 - that he was actually referring to thousands of rounds of SHA-512 vs bcrypt which uses hundreds or thousands of iterations itself. – [thomasrutter](#) Aug 25, 2015 at 5:56 
-



36



Blowfish is not a hashing algorithm. It's an encryption algorithm. What that means is that you can encrypt something using blowfish, and then later on you can decrypt it back to plain text.

SHA512 is a hashing algorithm. That means that (in theory) once you hash the input you can't get the original input back again.

They're 2 different things, designed to be used for different tasks. There is no 'correct' answer to *"is blowfish better than SHA512?"* You might as well ask *"are apples better than kangaroos?"*

If you want to read some more on the topic here's some links:

- [Blowfish](#)
- [SHA512](#)

Share Improve this answer

Follow

edited Mar 17, 2016 at 5:58



[temporary_user_name](#)

36.9k ● 48 ● 159 ● 240

answered Oct 13, 2009 at 16:05



Glen


22.3k ● 3 ● 63 ● 77

21 I think the question is about using bcrypt as an irreversible protection for passwords, much as hashing is used for that purpose. – [erickson](#) Oct 13, 2009 at 16:47

3 @erickson the text "Q: but is Blowfish better than SHA512?" seems pretty clear to me and shows that the OP doesn't understand the differences between the 2 algorithms – [Glen](#) Oct 13, 2009 at 17:04

1 OP here. Actually, based on Glen's answer that blowfish is an encryption algorithm (which I understand is different from hashing), I realize now that my question yes was confused. What's confusing now though is that the phpass class (used for password hashing I believe) uses bcrypt (i.e. blowfish, i.e. encryption). If blowfish is encryption, how come phpass uses it to hash passwords, seems like a flaw to me, no? Am I missing something? – [Chris](#) Oct 13, 2009 at 19:22

2 however the question asks which of apples and kangaroos are better suited to a specific task. Blowfish is a better hashing function than sha because of the time it takes to hash. Most implementations of sha i've seen are very fast. You want a slow algorithm for password hashing.
– [John Nicholas](#) Mar 27, 2010 at 15:56

This answer is correct that Blowfish is an encryption algorithm, but in this context (eg, when used in `bcrypt`) it is used as a hashing algorithm by deriving a key from the source string and using that to encrypt a magic number. This makes it irreversible, essentially a hashing function. You cannot calculate the key from a cipher, even if you know the plaintext and encrypted data. – [thomasrutter](#) Aug 25, 2015 at 6:01 



Blowfish isn't better than MD5 or SHA512, as they serve different purposes. MD5 and SHA512 are hashing

4

algorithms, Blowfish is an encryption algorithm. Two entirely different cryptographic functions.



Share Improve this answer

Follow

answered Oct 13, 2009 at 16:06



[blowdart](#)

56.5k ● 12 ● 118 ● 151



I would recommend Ulrich Drepper's SHA-256/SHA-512 based crypt implementation.

2



We ported these algorithms to Java, and you can find a freely licensed version of them at ftp://ftp.arlut.utexas.edu/java_hashes/.



Note that most modern (L)Unices support Drepper's algorithm in their /etc/shadow files.

Share Improve this answer

answered Mar 18, 2010 at 18:07

Follow



Jonathan Abbey

21 ● 1

PWDTK sourceforge.net/projects/pwdtknet uses HMAC-SHA512 however it does so over many iterations to create "slowness" aka Key Stretching as others here have been talking about. BCrypt is better than a single SHA-512 as has been mentioned, however if you use SHA-512 in something like PBKDF2 then you are well secure (As long as you are using a large crypto-random salt and enough iterations to force time to make a rainbow table) the API I just posted is built by me and will do what you want in .NET if that is what you are developing for (For future readers benefit)

– [thashiznets](#) May 10, 2013 at 3:44
