Bcrypt at 25: A Retrospective on Password Security

May 22, 2023

Opinion

Authors:
Niels Provos

Article shepherded by:
Rik Farrow

As one of the creators of bcrypt back in 1997, I find it somewhat surprising that, 25 years later, we still rely heavily on passwords. My initial encounter with password security took place in 1993 at the University of Hamburg. Having just discovered Unix, I was fascinated by a service called Yellow Pages that allowed anyone to dump encrypted

password databases. When I reached out to my friends at other universities, requesting their password dumps to run crack on them, some of the system administrators were quite displeased with me. They found themselves victims of weak system security and an insecure password hashing algorithm. While I can't claim that this directly inspired my future work, it's clear that bcrypt ultimately helped to rectify some of the wrongs committed during my innocent youth.

The concept of computer passwords dates back to the early days of multi-user computing systems when users required a method to authenticate themselves and protect their data. In the 1960s, the Compatible Time-Sharing System (CTSS), developed at MIT, became one of the first systems to implement password-based authentication for user accounts. This came over the strenuous objections of Richard Stallman who famously tried to resist the introduction of passwords at MIT in the 1970s (Levy, 1984). However, the true origins of modern password security can be traced back to the development of Unix and its password hashing function, crypt.

Unix, developed in the 1970s by Ken Thompson and Dennis Ritchie, introduced a more robust password hashing function called crypt. The original version of crypt in Unix (6th Edition) was implemented by Robert Morris and based on the Hagelin cipher machine (M-209). Even back then, the quick computation using crypt was known to be a problem. In Unix (7th edition), the crypt function was refined by introducing a 12-bit salt and iterating the Data Encryption Standard (DES) cipher to create a hash from a user's password. This use of salt essentially created a family of 2^12 distinct hash functions, with each user randomly drawing from this pool for their password. The purpose of the salt was not only to ensure uniqueness, making each hash distinct even for identical passwords, but also to significantly impede precomputed hash attacks. The hashed password, along with the salt, was stored in the password file, which allowed the system to authenticate users without storing their plaintext passwords. While crypt was considered secure for its time, short-sighted US export policies and advances in computational power paved the way for new

password hashing algorithms such as MD5crypt, introduced in 1994 by Poul-Henning Kamp (Kamp, 2012).

Despite the emergence of new algorithms, concerns about their security and resistance to attacks led to the development of bcrypt (Provos & Mazieres,1999). The concept of adaptive hashing, which made brute force and dictionary attacks more and more computationally demanding, was the brainchild of David Mazières. I was lucky to have had the privilege of collaborating with him on this new algorithm. Being a German citizen residing in Germany at the time, I was unhindered by US export restrictions on cryptography, and this allowed me to take the lead in implementing bcrypt. Since its introduction in June 1997 as part of OpenBSD 2.1 and publication by USENIX in 1999, bcrypt has had a profound impact, shaping the landscape of password security over the past quarter-century.

In this short retrospective, I will examine the history of password hashing, key developments since the introduction of bcrypt, and where I see the field of security 25 years later.

Password Guessing

Password guessing tools are like toddlers: they'll try anything, and they won't give up easily. - Google's Bard AI.

Before delving deeper into bcrypt and secure password hashing, let me explain why secure password hashing matters. In the past, the theft of password databases from compromised systems was a common problem. With the introduction of crypt, passwords were no longer stored in plaintext. To learn the password for an account from an encrypted password dump, it was necessary to guess at a potential password, hash it and then compare it against the encrypted password database. To nobody's surprise, humans were, and still are, predictable in their password choices. Consequently, various tools emerged to guess common passwords and compare them with the hashed passwords in the user database. These tools typically employ a

combination of dictionary attacks, brute force and other techniques to guess potential passwords and check them against stored hashes (Bonneau, 2012). John the Ripper, L0phtCrack, Hashcat, and Hydra are some popular examples of password cracking tools.

When attackers successfully recover passwords from a database dump, they can then try them on different servers and sites, often gaining access to email accounts or other sensitive user data. Over the years, password cracking tools have become increasingly effective. John the Ripper, developed by Solar Designer, led the charge in technological innovation for password guessing. Later, with the advent of powerful GPUs, Hashcat was able to harness their power to further accelerate password guessing. I will discuss this in more detail later in this retrospective.

	Security	**Adaptable Work Factor**	**Memory-hardness**	**Year Introduced**
CRYPT	Not recommended	No	No	1970s
MD5crypt	Not recommended	No	No	1994
BCRYPT	High	Yes	No	1999
PBKDF2WithHmacSHA1 (RFC 2898, 2000)	Moderate to high	Yes	No	2000
Scrypt (Percival, 2013)	High	Yes	Yes	2009
Argon2 (Biryukov & Dinu, 2016)	High	Yes	Yes	2015

Table 1: Comparison of password hashing functions and their security over time. Adaptable work factor means that CPU cost can be increased. Memory-hardness means that the hashing algorithm also scales in memory in addition to CPU consumption.

The Password Hashing Landscape Post-Bcrypt

Back in 1997, I had little idea that bcrypt would be a significant innovation in password security. With the Internet's rising popularity, data breaches involving hashed passwords were becoming commonplace, revealing shortcomings in the password hashing algorithms of the time. This underscored our motivation to create bcrypt, an algorithm designed to resist the rapid advancements in computational power.

David Mazieres' key contribution to bcrypt was the adjustable cost factor, an innovation that introduced adaptive hashing, making brute force and dictionary attacks increasingly resource-intensive. This critical feature allowed bcrypt to keep pace with computing power advancements, maintaining its robustness against evolving threats. Bcrypt's influence stimulated the development of new algorithms and research to counter password cracking techniques. Its concept of adaptive work factors has been adopted and expanded upon by newer algorithms like PBKDF2 and scrypt, incorporating advanced features such as memory-hardness and adjustable parallelism. The Password Hashing Competition in 2013 further stimulated innovation in the field, leading to the creation of Argon2, which is gaining increased adoption as a bcrypt alternative.

Both memory-hardness and adjustable parallelism were meant to counter password guessing on specialized hardware such as GPUs. Memory-hardness increases the password guessing cost by requiring a

substantial amount of memory for each guess. This is especially effective in deterring attackers who use parallel computing resources to speed up password cracking. Memory-hardness was an idea first introduced by Abadi et al. in 2003 and applied to passwords by Percival in 2009. Adjustable parallelism allows the algorithms to make use of multiple processing cores, further increasing the computation cost and reducing efficiency gains through guessing passwords in parallel.

Data Breaches With Stolen Passwords Post-Bcrypt

There are only two types of companies: Those that have been hacked and those that will be hacked. – Robert S. Mueller III, former Director of the FBI.

I don't need a password manager, I just use the same password for everything. - Unknown

Initially, I believed that generating secure password hashes might be considered a solved problem. Sadly, industry adoption hasn't been as swift as I had hoped. Even after introducing bcrypt and other more secure password hashing algorithms, numerous high-profile data breaches have occurred involving passwords hashed with weaker algorithms. These incidents highlight the importance of strong password hashing algorithms but also expose the sluggish pace at which the industry embraces new security technologies. Here are a few notable examples:

- **LinkedIn (2012)**: In 2012, LinkedIn was hacked, and approximately 117 million password hashes were stolen. The passwords were stored using the SHA-1 hashing algorithm, which is no longer considered secure.
- **Yahoo (2013 and 2014)**: In 2013 and 2014, Yahoo was hacked, and a total of 3 billion user accounts were compromised. The accounts included names, email addresses, passwords, and birth

dates. The passwords were stored using a variety of hashing algorithms, including MD5, SHA-1, and bcrypt.

- **Adobe (2013)**: In 2013, Adobe was hacked, and 153 million user accounts were compromised. The accounts included names, email addresses, passwords, and credit card numbers. The passwords were stored using a custom encryption method rather than a proper hashing algorithm, making it easier for attackers to crack them. The password hints were stored using the SHA-1 hashing algorithm.
- **MySpace (2016)**: The MySpace breach was reported in 2016, but it is believed to have occurred around 2013. In this incident, 360 million password hashes were stolen. The passwords were stored unsalted using the SHA-1 hashing algorithm.

These breaches underscore the importance of using strong password hashing algorithms like bcrypt, scrypt, and Argon2. While bcrypt had already been introduced and was known to offer stronger protection, many organizations continued to use weaker hashing algorithms, leaving user data vulnerable to attackers. These breaches have motivated the increased adoption of more secure password storage methods and the development of more advanced hashing algorithms to safeguard user information. Alas, even now, we should expect more breaches where weak password hashing algorithms were used. As I will argue later, security is no longer a technical problem — rather it is a people and incentives problem.

Password Hashing Performance Over Time

Over the last 30 years, we have seen password cracking techniques and hardware evolve significantly, resulting in substantial improvements in password guessing performance. Given the performance improvements in CPUs and GPUs, it has become obvious how important it is to employ work-cost adaptation for password hashing algorithms. To demonstrate this progression, I've compiled a

list of password guessing scenarios over time, highlighting the number of password guesses per second for various algorithms and tools. Please keep in mind that these figures serve as an overview of general trends and might not be directly comparable due to differences in hardware, software, and configurations used.

- 1978: PDP-11/70 (M-209 simulated crypt) → 800 passwords/second
- 1988: VAX 8600 (Morris-worm optimized des-crypt) → 45 passwords/second
- 1994: 60MHz Pentium (MD5-based crypt) → 29.41 passwords/second
- 1999: John the Ripper (bitsliced DES-crypt) → 214,000 passwords/second
- 1999: John the Ripper (bcrypt with work factor 5) → 62.5 passwords/second
- 2018: Hashcat (des-crypt on a GPU-based rig) → 1.7 billion passwords/second
- 2018: Hashcat (MD5 hashes on a GPU-based rig) → 45.4 billion passwords/second
- 2018: Hashcat (SHA-1 hashes on a GPU-based rig) → 14.6 billion passwords/second
- 2018: Hashcat (bcrypt with work factor 5 on a GPU-based rig) → 47.2 thousand passwords/second
- 2018: Hashcat (scrypt on a GPU-based rig) → 1.4 million passwords/second
- 2022: Hashcat (des-crypt on an RTX 4090 GPU) → 6.3 billion passwords/second
- 2022: Hashcat (bcrypt with work factor 5 on an RTX 4090 GPU) → 184 thousand passwords/second

Over the course of 34 years, we've witnessed an extraordinary leap in the password guessing speed for DES-crypt; it's accelerated from a mere 45 passwords per second to an astounding 6.3 billion passwords per second. This advance doesn't just surpass Moore's law — it pulverizes it, tripling the password cracking speed every two years.

Bcrypt's Enduring Performance Over the Past 25 Years

Personally, I think that bcrypt and SHA-crypt are quite close to becoming obsolete in favor of something new, yet right now I would still recommend moving to bcrypt as the most suitable pre-existing password hashing method. - Solar Designer, June 2012 on Bugtraq mailing list

Throughout my 25-year journey with bcrypt, I've seen it withstand challenges posed by rapid advancements in computing power and the development of new password cracking techniques. Surprisingly, it still remains an effective and reliable choice for password hashing.

We designed the adaptable work factor in bcrypt, primarily to keep pace with increasing CPU performance, but also included factors to make it more robust against custom ASICs or GPU acceleration, e.g. by heavily relying on instructions that generic CPUs execute efficiently. A crucial aspect of bcrypt's resistance to GPU optimization lies in its memory access pattern spanning 4KB during the key setup phase, which is inefficient to parallelize on GPUs with small L1 caches (Malvoni et al., 2014). That said, NVidia increased the L1 cache significantly to 16MB for the RTX 4090 and bcrypt’s 4KB is a tiny amount of memory these days.

Bcrypt's endurance can be attributed to several other factors beyond our intentional algorithm design. Its wide availability in open-source implementations has facilitated widespread adoption and integration into various systems. According to Wikipedia, there are implementations of bcrypt in C, C++, C#, Embarcadero Delphi, Elixir, Go, Java, JavaScript, Perl, PHP, Python, and Ruby. Moreover, bcrypt's focus on computational cost scaling makes it an attractive choice for large Internet services compared to newer algorithms like Argon2, which also scale in memory consumption. As large-scale Internet services handle most password checking and need to optimize resource consumption for economic reasons, bcrypt's robust security, without excessive memory demands, proves beneficial. Modern

approaches like Argon2, with their memory-hardness, can be less compelling in some cases, as their increased memory consumption may not align with resource optimization goals, contributing to bcrypt's continued relevance in password security.

That said, Facebook has successfully deployed scrypt at 16MiB memory consumption and Solar Designer has worked on an extension of scrypt called yescrypt that can leverage large amounts of site-shared read only memory to overcome the throughput concerns faced by large-scale Internet services (Solar Designer, 2017).

The Future of Passwords and Security

Like the pulsing rhythm in an EDM track, bcrypt keeps the beat of password security, forever adapting to the increasing tempo of computational power. - Activ8te, 2023.

Over the years, I've observed modern password hashing algorithms significantly reduce the effectiveness of brute-force password guessing. However, password stuffing attacks—where attackers use previously leaked credentials to gain unauthorized access—continue to be a persistent threat. On the other hand, the advent of multi-factor authentication (MFA) has shifted the focus to protecting user accounts through additional layers of verification, making passwords less critical to security.

In today's digital landscape, the adoption of cloud services has increased, with most critical data being stored and processed remotely. Brute-force password guessing against online services often encounters rate-limiting protections and captchas that require human interaction. As a result, attackers targeting cloud-based services tend to exploit vulnerabilities or use social engineering to compromise insiders to breach sensitive customer or user data.

Despite these challenges, passwords remain an integral part of our digital lives. Their ease of deployment, creation, and revocability make them a convenient choice even today. While single sign-on is now nearly ubiquitous, there is a significant risk in trusting a single identity provider (IdP) with one's entire online presence. For instance, an IdP might disable your account due to poorly designed abuse detection rules, or because your account has been erroneously flagged as a false positive. Utilizing a password eliminates the risky dependency on third-party services, which, while sad to say, is a pertinent concern. Until alternatives such as self-sovereign identity (Allen, 2016) - where users retain control over their identities — gain widespread acceptance, the reign of passwords is likely to endure.

In my experience, most security professionals consider password security a solved problem from a technical perspective. I would even go further and claim that security is no longer a problem that I consider to be primarily technical. By and large, most security challenges have technological solutions that sufficiently address them. Instead, the main problem I see with companies improving their security and reducing the frequency of data breaches, boils down to human factors and the cost of adopting existing security technologies.

Let’s talk about adopting security technologies first. Currently, there are no off the shelf solutions that lead to good security outcomes. Truly robust security is often seen only in Internet-based software engineering companies, where the executive team places a high priority on security and is prepared to invest in it. Most other companies usually seek to choose an appropriate trade off between investing in business growth and security. The incentives are such that it is the rational argument to invest in business growth and treat a data breach as an eventuality that will be dealt with when it happens. Tighter regulations, more stringent enforcement and increased liability might change that calculus.

For the few companies that achieve a mature security posture, human factors often dominate their security concerns, and insider risk

becomes the primary focus. Many of these threats stem from the vulnerability of humans to social engineering, leading them to act on behalf of others, or from disgruntled employees who, in a moment of weakness, abuse their privileges to steal and leak sensitive information. Addressing insider risk is a more complex challenge, as it involves a human component and typically lacks a straightforward technological solution.

That said, it's much easier to address human behavior when there's a strong technological security foundation in place. A robust approach to securing infrastructure from both external attacks and rogue insiders necessitates the development of strong security invariants integrated into the infrastructure to mitigate risks. This often calls for engineering bespoke solutions and employing skilled software and security engineers. Taking such an approach brings us right back to the previous point, i.e. the high cost of adopting security technologies.

It turns out that even companies who are willing to heavily invest in improving their security, face significant challenges in finding and hiring skilled engineers with security domain expertise. As such, one of the most significant challenges in addressing security issues may be the scarcity of skilled security professionals. I believe that there is an urgent need to generate more interest in the field and create a strong talent pipeline to ensure the continued development of effective security measures.

To address this scarcity of talent, I've pursued a very unconventional approach. I've embarked on a new venture as an EDM (Electronic Dance Music) producer under the artist name Activ8te, creating cybersecurity-themed EDM tracks. My goal is to captivate a younger audience and ignite their interest in security topics. Some of my recent tracks, like "Teardrop Falling" and "I Am Tracking You," explore challenging security themes such as denial of service, censorship, and the risks to our privacy posed by sophisticated spyware (Activ8te, 2022). By raising awareness and enthusiasm for the field, I hope to contribute to the expansion of a skilled security professional pipeline.

With the goal of winning a Grammy for one of my cybersecurity-inspired tracks - a thought that never fails to amuse me - I aim not only to showcase my music, but also to create a broader interest in the field of security. It's my hope to raise awareness about the need for more proficient security talent in our constantly changing digital world. Perhaps the next retrospective, 25 years in the future, will celebrate the fact that security talent is no longer scarce, and we'll be living in a world where our data is secure and our privacy protected. Hopefully, by then, the topic of passwords will be a thing of the past.

Further Reading

Abadi, M., Burrows, M., Manasse, M., & Wobber, T. (2003). Moderately hard, memory-bound functions. NDSS Symposium..

Activ8te (2022). Cybersecurity meets EDM. **https://activ8te.io/**

Allen, C. (2016). The Path to Self-Sovereign Identity. **https://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-id...**

Biryukov, A., & Dinu, D. (2016). The Argon2 memory-hard function for password hashing and other applications. In International Conference on Fast Software Encryption (pp. 170-189). Springer, Berlin, Heidelberg.

Bonneau, J. (2012). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. 2012 IEEE Symposium on Security and Privacy.

Chiasson, S., Biddle, R., & Van Oorschot, P. C. (2006). A second look at the usability of click-based graphical passwords. In Proceedings of the 3rd Symposium on Usable Privacy and Security (pp. 1-12). ACM.

Florencio, D., & Herley, C. (2007). A large-scale study of web password habits. In Proceedings of the 16th international conference on World

Wide Web (pp. 657-666). ACM.

Kamp, P.-H. (2012). The history of md5crypt. **https://phk.freebsd.dk/sagas/md5crypt/**

Levy, S. (1984). Hackers: Heroes of the Computer Revolution
Malvoni, K., Solar Designer, and Knezovic, J. (2014). Are Your Passwords Safe: Energy-Efficient Bcrypt Cracking with Low-Cost Parallel Hardware. In WOOT.

Percival, C. (2013). Stronger key derivation via sequential memory-hard functions. Self-published. **https://sites.cs.ucsb.edu/~rich/class/old.cs290/papers/scrypt.pdf**

Provos, N. and Mazieres, D. (1999). Bcrypt algorithm. USENIX ATC, FREENIX Track.

RFC 2898. (2000). PKCS #5: Password-Based Cryptography Specification Version 2.0. IETF.

Seeley, Donn (1989). A Tour of the Worm: **https://www.cs.unc.edu/~jeffay/courses/nidsS05/attacks/seely-RTMworm-89....**

Solar Designer. (2012) Password security: past, present, future. Openwall.

Solar Designer. (2015). bcrypt: Cost, parallelism, and attacks on GPUs. Openwall.

Solar Designer (2017). yescrypt: large-scale password hashing. **https://www.openwall.com/presentations/BSidesLjubljana2017-Yescrypt-Larg...**

Ur, B., Bees, J., Segreti, S. M., Bauer, L., Christin, N., & Cranor, L. F. (2015). Measuring Real-World Accuracy and Impact of Password Strength Meters. In Proceedings of the 24th International Conference on World Wide Web (pp. 283-293).

Article Categories: Security

Last updated May 21, 2023

Authors:

Niels Provos is a German-American computer security practitioner with a PhD in computer science from the University of Michigan. His career includes roles at Google, Stripe, and Lacework. With over 15 years of experience as a people manager, Niels is dedicated to fostering well-executing and healthy teams. At Google, he managed most of the security engineering teams and played a pivotal role in creating Safe Browsing, enhancing web security for billions of users. In his role as Head of Security at Stripe, he employed his expertise to establish a world-class security posture. He has made significant contributions to OpenSSH and developed the Honeyd honeypot system. Niels has shared his management philosophy and insights on creating a healthy company culture through multiple articles.

In addition to his cybersecurity career, Niels is an accomplished swordsmith

and electronic music producer under the artist name Activ8te. Passionate about both crafts, he finds parallels between forging swords and defending against cyber threats. His music serves to create an emotional connection with his audience while educating them on challenging cybersecurity topics. One of his recent tracks, "Teardrop Falling," was inspired by his early work on preventing Distributed Denial of Service attacks at scale.

provos@gmail.com

Log in to post comments