

# The Extinction of Password Authentication

By Fernando Alonso – ISSA member, Argentina chapter

**In the near future, password authentication will become either extremely insecure or extremely difficult to implement among human users. This article focuses on the problem from an analytic perspective.**

The usage of *username* and *password* has been the most common means of authentication in the past years. Technology evolution is becoming a threat to this paradigm. In the near future, password authentication will become either extremely insecure or extremely difficult to implement among human users. This article focuses on the problem from an analytic perspective.

## Password authentication mechanism

In secure environments, the most common password authentication mechanism is through hash value comparison. A hash function is a computational procedure that turns some kind of data into a relatively small set of characters. It is a one-way process, which means that there is no way to determine the original data from it.

**Data → Hash Function → Hash Value**

Boca Juniors rule soccer  
in Latin America

H76s\$JGT8@4RmkG(7

Hashes possess several characteristics making it the suitable mechanism in password authentication:

- There is no way to discover the data from the hash value
- The hash length does not imply the data length
- Source data approximation from pattern or similarity is impossible to achieve
- Hashing methods and algorithms are publicly known, supporting its assurance

Passwords are not stored in plain text (at least on secured systems). The password's hash value is stored instead. Therefore, password authentication uses hashing as its core procedure.

The target systems have a record of all its users and hash values of their password. After the user types the password at the terminal, the system computes the hash value and compares it to the one it has stored.

Other systems implement the hash calculation on the terminal side. Then, only the hash value travels through the communication channel. This technique has its drawbacks. One of them is that once the target system changes its hashing method or authentication procedure, it must be also be set in all the clients. Another is that the hash value could be sniffed and captured during transit, allowing the one sniffing to authenticate with it. Specific password knowledge is no longer required.

## Password complexity combination

The average user often complains about implemented password policies, even in a low-security environment. The user is typically forced to conform to a minimum password length and complexity, periodic password change enforcement every *N* days, account lock-out after a maximum number of failed login attempts, exclusion of previous passwords and dictionary words, etc.

These and other policies often upset the users despite the fact that there are legitimate security reasons for them: because hash values are not reversible, the only way to technically obtain a password (without considering social engineering) is by guessing it.

The number of possible passwords is based on models of combinatory mathematics. Table 1 shows an example of the possible combinations a password policy could generate (based on the English alphabet and international keyboard). This amount of possible combinations (a.k.a. work factor) is the cost an attacker must bear to break into a system or crack a cryptosystem.

Password Policy	Min. # of characters	# of values a single character can take	Formula	# of possible combinations
At least 3 characters, only alphabetic	3	26	$26^3$	17,576
At least 5 characters, alphanumeric only	5	36	$36^5$	60,466,176
At least 7 characters, alphanumeric, with forced upper-cases	7	62	$62^7$	$3.5 \times 10^{12} = 3,521 \text{ billion}$
At least 7 characters, alphanumeric, with forced upper-cases, and symbols	7	94	$94^7$	$64.477 \times 10^{12} = 64,847 \text{ billion}$

**Table 1 – Password combinations**

Based on a situation where a computer tries to guess billions of combinations and a lock-out policy of five minutes after five failed attempts, there will be an average of one guess per minute. As a consequence, instead of milliseconds, we are dealing with billions of minutes to guess a password. It can be translated in years, decades, or even centuries. If we finally add the fact that the system forces the user to change the password every 90 days, it will be mathematically impossible for the cracker to find it.

## Password cracking

Password cracking is the process of discovering the value of the real password from the encrypted or hashed value. There are mainly two methods of password cracking:

- **Dictionary attacks:** Based on known possible passwords, such as dictionary words, names, calendar dates, etc.
- **Brute force attacks:** Trying all the possible character combinations based on known parameters.

Cracking a password is not an issue that could only be achieved by a professional hacker. Many effective, easy-to-use tools are freely available on the Internet (L0phtCrack, John the Ripper, LM Crack, etc). Numerous rookies and script kiddies possess the tools and knowledge to crack passwords.

A large number of security professionals consider that the hash collision<sup>1</sup> of the cracking process will occur before half of the combinations have been tried. This theory succeeds when working with several hashes to be discovered and dictionary attacks used at the beginning, having more logical than empirical probability to match and is based on the probabilistic exponential distribution, where the cumulative probability is distributed at the beginning of the sample.

Considering that the combination of characters will be dealt with equilibrated odds among each individual sample, the most suitable analysis would be based on normal distribution with a mean of 0 and a deviation of 1:

$$N(\mu = 0, \sigma^2 = 1)$$

<sup>1</sup> Collision is the moment where the obtained hash equals the pretended one. Related to Birthday Attacks.

Considering then a standard Gaussian distribution with an accuracy of  $2\sigma$ , we deem that the password will be cracked before 97.5% of attempts have been tried.

As it was detailed previously, lock-out tools will prevent intruders from using constant guesses until collision. However, there is a known weakness. The hashes could be intercepted and obtained in transit while they travel through the network to the target system in a distributed or client/server environment. Or even worst, they can be obtained from the file that stores them in the target system if the system is compromised by an attacker.

It is imperative to explain that many systems, even the most popular ones, store the password hashes in a file that has reading access to all registered users (reading access only, they can only be modified by the system's administrator or root.) The file is called `/etc/passwd` file in standard UNIX systems or SAM databases in Windows Server architectures.

Once an intruder obtains the hash value or list of values, she can copied them for herself and the lock-out hurdle will be avoided: the intruder will try to crack the hash value in her own environment, with several parallel CPUs and without the lock-out penalty. However, the combination value is still so high, that it could take the cracker weeks to guess it. That is the reason why periodical password change is so important.

Security professionals are aware of this matter, which is why they continuously try to enforce strong password policies. A strong password policy implementation will not only force crackers to use brute force attacks on obtained hashes but also force users to change passwords as often as possible/feasible. This issue (password crackers versus password policy enforcers) is a constant battle that seems to have no end, especially when users try to involuntarily help attackers with their complaints.

Password protection is being threatened by an evident fact: computers evolve constantly at exponential rates. They actually have multiple processors with multiple cores (dual-core, quad-core, octo-core, etc.). Nowadays, an affordable computer could have 16 or 32 processing units in it. In the upcoming years, bus transmissions in printed circuits will evolve from copper analog to optical signals (prototypes are already working), increasing computational processing speed drastically.

Such processing power will reduce password cracking time from years to months, days, hours, or even minutes. This time reduction will boost password policies to restriction levels that will be impossible to implement, at least for human beings, e.g., think of a policy that force users to generate and memorize passwords of at least 256 characters, using 16 digits, 24 upper cases, 32 symbols and must be changed 10 times a day.

## Evolution of CPUs

One of the parameters to be calculated is the progression of the CPUs speed in the past, in order to project it into the future. We will assume that the first recognized computer-oriented CPU was the Intel 4004 in 1971 (UNIVAC, ENIAC, and other vacuum tube-based macro computers will be skipped).

The first step in estimating the progression is to set the parameters in a Cartesian graph. In this case, the abscissa axis will be time evolution expressed in years, and the ordinate axis will be the cycles per second capability for a single processor expressed in KHz.

Based on 1971 as the first “CPU year,” each following year will be measured relative to it. This assumption will make the progression formula more readable. For example, the Intel 8086, developed in 1978, will be represented as 7 (1978 – 1971 = 7). Therefore, the years in the x-axis will be an integer offset number from the 1971 origin.

Based on this, a brief processing evolution would be:

Processor	Year	Offset	Cycles (in KHz)
Intel 4004	1971	0	745
Intel 8008	1972	1	800
AM9080	1974	3	2,000
Intel 8080	1974	3	2,000
Intel 8086	1978	7	5,000
IBM 801	1980	9	15,000
Intel 80286	1982	11	12,000
Intel 80386	1986	15	16,000
Intel 80486	1989	18	50,000
AM386	1991	20	33,000
IBM 6150 RT	1991	20	20,000
Intel Pentium	1993	22	60,000
AM486	1994	23	66,000
AM586	1995	24	133,000
AMD K5	1996	25	133,000
AMD K7	1997	26	550,000
Intel Pentium II	1997	26	233,000
AMD Athlon	1998	27	750,000
Intel Xeon	1998	27	800,000
Intel Pentium III	1999	28	450,000
AMD Duron	2000	29	950,000
Pentium IV	2000	29	1,300,000
AMD Sempron	2002	31	2,000,000
AMD Athlon 64	2003	32	2,200,000
Pentium M	2003	32	2,260,000
AMD Turion 64	2005	34	2,400,000
Pentium Core	2006	35	2,330,000
Pentium Dual Core	2006	35	3,000,000
AMD Opteron	2007	36	2,800,000
AMD Turion Ultra	2008	37	3,000,000

Table 2 – Evolution of CPU processor speeds

In a dotted Cartesian system we obtain the following graph:

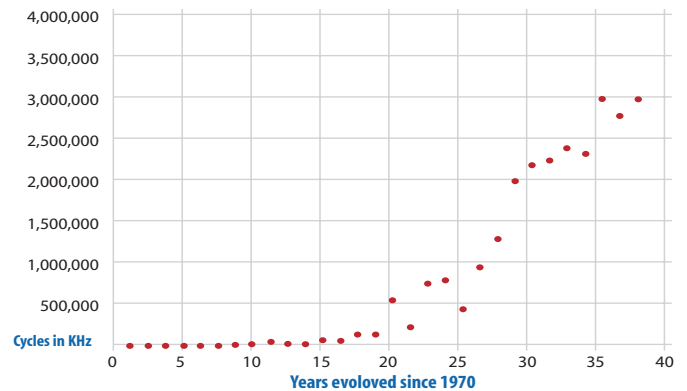


Figure 1 – Cartesian dotted expression of evolution of CPUs

This result can be compared to Moore’s law.<sup>2</sup> Although Moore’s theory referred to the increase of transistors since the creation of integrated circuits, his theory determines that processing capabilities will grow exponentially. There is another Moore’s law, compatible with his previous theory and this article as well, which states that the cost to acquire such technologies will decrease accordingly. This means that a capable technology to crack a password will someday be available to anyone at a Radio Shack store.

If we consider adding a trend line with a power regression, we will obtain the graph in Figure 2. The R<sup>2</sup> precision for such trend line is 0.9165, a quite confident value.<sup>3</sup>

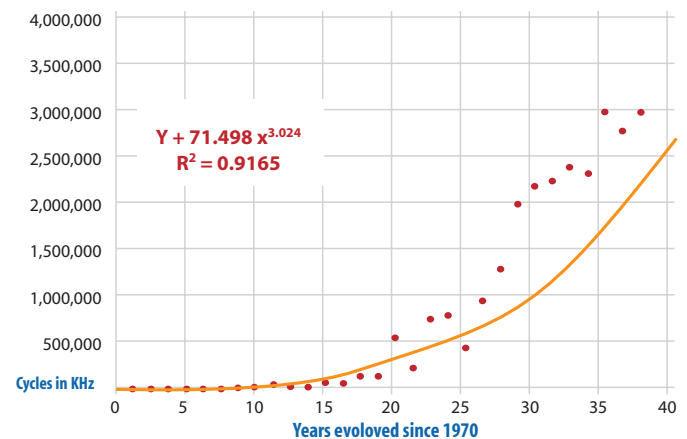


Figure 2 – Power regression

The determined regression function is:  $f(x) = 71.5x^3$ ,  $x$  being the number of years since 1971 and  $f(x)$  representing the CPU speed. For instance, if we would like to estimate a CPU’s speed in the year 2010, then we should make the following calculation:

$$2010 - 1971 = 39 \Rightarrow x = 39$$

$$f(x) = 71.5 \times 39^3 = 4241308.5 \text{ KHz} \approx 4.25 \text{ GHz}$$

On the other hand, if we would like to find out the approximate moment

2 Moore’s Law was stated by Intel Co-founder Gordon Moore in a 1965 paper.  
 3 An exponential regression would be more precise considering the R<sup>2</sup> deviation. However, when  $x$  is greater than 30, it starts to behave asymptotically. Such infinite value would be quite difficult to analyze with real facts.

in time where CPUs will reach speeds of up to 5GHz, then the formula would be:

$$\sqrt[3]{\frac{5000000\text{KHz}}{71.5}} = 41.2$$

which estimates that it will occur in February-

March of 2012. This formula shouldn't be taken so precisely, though. We must bear in mind that we are dealing with statistical trends with deviations and human developments influenced by social behaviors.

## Projected time of password cracking

In order to generate a hash value from source data, CPUs must step into several mathematical and logical operations. Once obtained, the comparison against the real hash is performed. This calculation and comparison is accomplished through several logical operations. Once finished, the CPU continues with the next attempt. Considering the SHA-1 and MD5 hashing standards, it takes approximately 64 standard CPU cycles to obtain a hash.

Let's figure out the case of a fully dedicated PC performing this task with a 3 GHz processor. Bear in mind the fact that the whole CPU resources will not be dedicated to this task. There are other matters to attend, like basic operating systems services or daemons, kernel requests, context switches, interruption requests, etc. Therefore, we will assume 50% of the CPU cycles will be dedicated to the password cracking activity.

As a result, considering a 3GHz processor at 50% of it usage means 1,500,000,000 cycles per second. If it takes 64 cycles per hash value, then our CPU will be able to obtain 23,437,500 hashes per second. In order to crack a password which offers 65,000 billion combinations, today's CPU will take 32 days to collide it (considering of course 97.5% of confidence from Gaussian distribution) – a long time.

What would happen, however, if the cracker uses 10 parallel PCs in order to break a tenth of the combination spectrum each one? In this case, the cracking time should be considered from just a tenth of the combinations. That means 6,500 billion attempts. With a 97.5% confidence, a single CPU will crack it in three days.

What would happen if crackers use servers with two parallel CPUs? Parallel multiprocessors (which differs from a single processor with multiple cores) is a common environment in computers today. Even though we are not dealing with low-cost environments – such a server configuration costs around \$10,000 USD, quite expensive for a script kiddy but extremely cheap if the target is an international bank or government agency – such a configuration will reduce the cracking time by half. If the cracker uses a single one, it will take 16 days; if 10, an average password will be cracked in a day and a half.

What would happen if we use 5GHz processors instead of 3GHz, using the previous configuration? In this case, it will take only 22 hours to find it. If we use the previous estimation formula:

$$\sqrt[3]{\frac{5000000\text{KHz}}{71.5}} = 41 \Rightarrow 1971 + 41 = 2012$$

By the year 2012, strong passwords will be vulnerable to cracking in 22 hours or less.

## Conclusion

Password authentication efficiency seems to be in danger. Even considering policies forcing users to change them every 10 days, the time will come when they will be cracked in less than one day, obtaining nine days to develop the silent attack without being discovered by intrusion detection tools due to its "legitimate user" natural behavior. So, the real concern here is what to do about it. Are passwords really going to die? Can they be replaced by passphrases or tokens? Even if we can predict CPUs speed in the upcoming years, it is impossible to predict the future with certainty per se. The fact is that passwords are endangered. They may disappear, or perhaps some alternatives may arise.

- Beverstock, D. 2003. Passwords are dead, long live the passwords. SANS Institute - [http://www.sans.org/reading\\_room/whitepapers/authentication/1144.php](http://www.sans.org/reading_room/whitepapers/authentication/1144.php).
- Duncan, R. 2003; An overview of different authentication methods and protocols. SANS Institute - [http://www.sans.org/reading\\_room/whitepapers/authentication/118.php](http://www.sans.org/reading_room/whitepapers/authentication/118.php).
- Fisher, A. 2001. Authentication and Authorization: The Big Picture with IEEE 802.1X. Penn State University – College of Information Science and Technology - <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.2762>.
- Graham, D. 2003. It's all about authentication. SANS Institute - [http://www.sans.org/reading\\_room/whitepapers/authentication/1070.php](http://www.sans.org/reading_room/whitepapers/authentication/1070.php).
- Kaufman, K. 2003. Identity Management. GIAC – Global Information Assurance Certification - [http://www.giac.org/certified\\_professionals/practicals/gsec/3150.php](http://www.giac.org/certified_professionals/practicals/gsec/3150.php).
- Lee, J. Wook. 2001. Strong Authentication and Authorization model Using PKI, PMI, and Directory. GIAC – Global Information Assurance Certification - [http://www.giac.org/certified\\_professionals/practicals/GSEC/1304.php](http://www.giac.org/certified_professionals/practicals/GSEC/1304.php).
- Loveland, N. 2003. Single sign-on through password authentication. Security Docs - <http://www.securitydocs.com/library/1006>.
- Rallo, K. 2002. Clear text passwords and authentication. SANS Institute - [http://www.sans.org/reading\\_room/whitepapers/authentication/113.php](http://www.sans.org/reading_room/whitepapers/authentication/113.php).
- Wilson, S. 2002. Combating the lazy user: An examination of various password policies and guidelines. Security Docs - <http://www.securitydocs.com/library/1005>.

## About the Author

*Fernando O. Alonso is an independent security consultant and researcher. He possesses both a Bachelor of Science and a Master of Science in Computer Information Systems. Educated and trained in the U.S., he is now based in Buenos Aires, Argentina. After working in different U.S. and international companies, he settled on his own venture advising companies in the fields of network security, operations security, and backup strategies. Fernando can be reached at [feralonso@gmail.com](mailto:feralonso@gmail.com).*

