Password Hashing Algorithms - From Past to Future

[1] Tran Song Dat Phuc, [*1,] Changhoon Lee
[1]*Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, Korea, datphuc_89@yahoo.com*
[*1,] *Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, Korea, chlee@seoultech.ac.kr*

Abstract

Password is a simple technique users do use to protect and authenticate their private information. It is employed popularly as an effective way in most computer applications and services in the early information security. Afterwards, due to the weakness of length and some bad habits of users make password be vulnerable to attacks, the key stretching and especially, password hashing techniques were presented to fix the problems and provide a stronger protecting. Both have hash functions within as major elements. However, hash functions just focus on hashing and storing password, not security at all, that leads to the need of more effective and secure techniques. Several advanced password hashing schemes were proposed, such as PBKDF2, bcrypt, and scrypt. Recently, Password Hashing Competition (PHC) was occurred to give more supplementary choices, besides the three mentioned schemes above, that are suitable for widespread adoption of modern information era. Along with other effective algorithms, the final PHC winner, Argon2, is a remarkable secure design which shows high performance and high resistance against most known attacks. In this paper, we give a general overview about password hashing techniques from beginning to present, measures and evaluates security aspects of recent password hashing schemes (through Password Hashing Competition), and also makes a prediction of future password hashing in information era so far.

***Keywords**: Password Hashing, Password Hashing Competition (PHC), PBKDF2, bcrypt, scrypt, Argon2, GPU Attack, FPGA/ASIC Attacks, Timing Attack, Side-channel Attack.*

1. Introduction

With the appearance of the Internet and the rapid development of computer technology, the demand to transfer information is increasing more and more. Since a large amount of information is exchanged constantly, security of information becomes a major interest. The users' private information need to be secure protected. At the beginning of information security, password is used widely in most computer applications and services by users as an effective way to protect and authenticate sensitive information. However, length of password provided by providers is limited (8 to 12 bytes in standard) in some cases together with bad habits of users when setting up passwords (too short or too easy to guess) make password be vulnerable to attacks. In addition, service providers who process and manage users' passwords just authenticate and verify their clients by comparing a pair id/password at server database with a pair id/password put in by users, the simplest way without any safety.

To avoid these problems and provide a stronger protecting, key stretching and password hashing techniques were proposed. Key stretching is a technique helps password (or key) be difficult to be exploited by attackers. A new key is an enhance-size version of the initial key will be produce through a certain cryptography algorithm, makes it unfeasible to break by attacks methods, especially brute force. In common, a hash function is used as that cryptographic algorithm to output a fix-length key. Password hashing technique is a one-way transformation turns password into another string, "hashed

* Corresponding Author
Received: Nov. 21, 2015, Revised: Jan. 12, 2016, Accepted: Jan. 17, 2016

password", with fix-length, and combines these concepts while the hash functions are iterated several times to provide a secure password (key).

Unfortunately, while the hash functions just care about hashing and producing a fix-length "hashed password" for service providers to store it at server database, no security method is provided enough to protect them. Somehow, if attacker obtains hashed value used, he can use it to expose user's password by some attack methods (such as dictionary attack, guessing attack, and "rainbow tables" attack). This is seriously dangerous!

Some advance password hashing schemes were proposed to overcome those attacks, with PBKDF2 [8], bcrypt [9], and scrypt [7]. Those give a much better protection for password security. However, thanks to the great development of parallel computing and hardware technologies, attackers are able to effectively employ their attacks. The attempts per time unit are increased in parallel on GPUs, FPGA/ASIC attacks that enhance the chance to expose user's information.

All those limits lead to the Password Hashing Competition (PHC) occurred in 2013. This is an open competition in order to employ and develop new designs for secure password hashing schemes. Until the deadline, there were total 24 algorithm candidates submitted to the competition. After the 2nd round, 9 candidates were remaining. And the final PHC algorithm, Argon2, from University of Luxembourg was announced as winner in July, 2015. This along with 4 other outstanding algorithms, Catena, Lyra2, Makwa, and yescrypt, provide high performance as well as high resistance against most known attacks.

This paper makes a survey on password hashing algorithms from past until present. We give a general overview on basic concepts of password hashing techniques, schemes such as PBKDF2, bcrypt and scrypt. Then, several outstanding PHC candidates are evaluated and measured on some major security aspects (such as trade-off analysis, GPU attack resistance, FPGA/ASIC attacks resistance, and side-channel attack resistance). The direction of future password hashing techniques is also discussed. Finally, the briefly conclusion is given to summary and complete our paper.

2. Password Hashing in the Past

This chapter presents base concepts on password hashing at the beginning. We cover during a period of time when the more demand of security information increases, the more necessary of having new secure techniques are to protect and verify. The initial classical password hashing technique will be reviewed in general. Then, some standard and advance password hashing algorithms, PBKDF2, bcrypt, and scrypt, are also rated respectively in order to give a more comprehensive view of the information security development.

2.1. Basic Password Hashing

Password can be a string, word, or character provides effective way for users to protect and authenticate their information. Authentication servers will process data from users, manage and store them to databases. Before being transfer to database to be stored, password changes its initial form to another (with salts), in ordinary generates cryptographic key. The work can be done with a Key Derivation Function (KDF) helps to derive one or more sub-keys from a master key (password). The servers verify user identification in such a simple way of comparing values on database and values input by user. This way has arisen many problems in security, making password to be weak and information be easy to be exploited.

Hash function is a technique to create a fix-length output from any-length input. This is useful in managing, storing and securing while making password much longer (32 or 64 bytes). However, with same input at initial of the function will give you same output as a same result that causes difficulty in verification, authentication as well as defense against attacks (dictionary attack, or track-off attack using rainbow tables).

Key stretching is a typical technique applied to password using the hash function within. It enhances resistance probability against attacks by extending key size through a cryptographic algorithm (normally hash function). However, the disadvantage exists when the hash function is iterated many times to produce new enhance-size keys (hashed passwords) that may result in slow performance of user experience.

Password hashing schemes have became the cryptographic primitive and the most secure solution for password protection. While combining with KDFs, some advance widely-used constructions can be concerned are PBKDF2 (standard), bcrypt and scrypt.

Figure 1. shows a basic password hashing scheme.

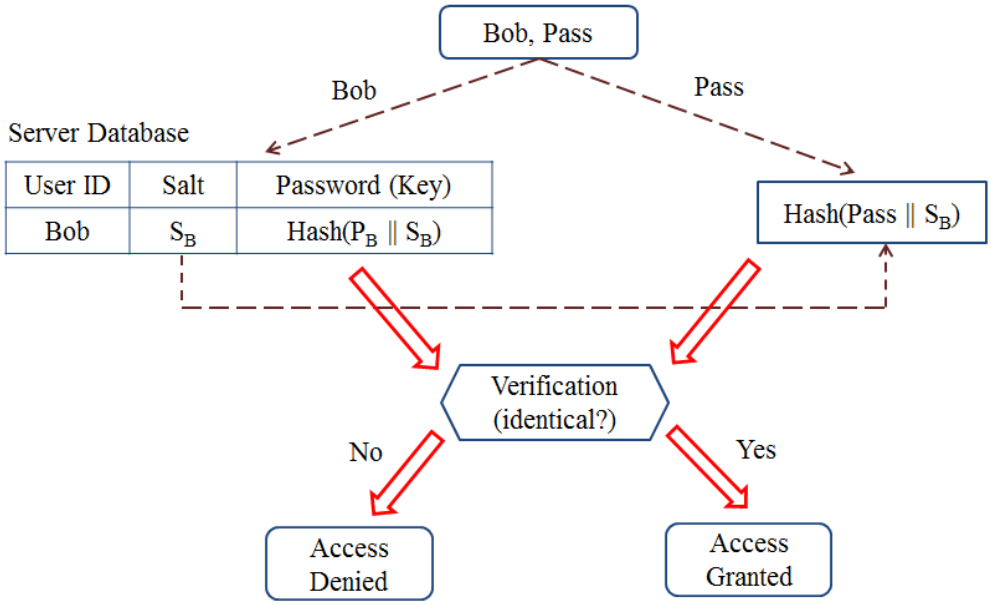

Figure 1. Basic Password Hashing Scheme.

2.2. PBKDF2

PBKDF2, stands for "Password-Based Key Derivation Function, Version 2", is an improved version replaces previous standard, PBKDF1 while the earlier has limit of key size support (only up to 160 bits). The structure is a part of the RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, published as an official standard in PKCS #5 [9] and RFC 2898 [8] documents.

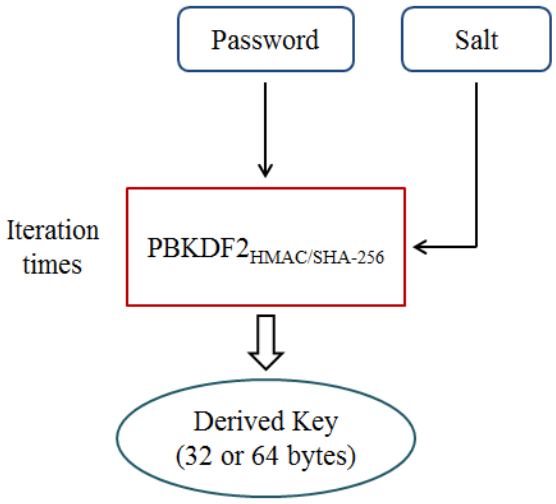

Figure 2. Standard PBKDF2 Scheme.

This scheme is based on KDFs, takes a password and a salt as an initial input through a pseudorandom function, such as cipher, SHA-256 hash, and HMAC, to produce a derived key by repeating the process many times (**Figure 2.**). The salt used in PBKDF2 can be up to 8 bytes size, makes trade-off attacks methods (dictionary attack, "rainbow tables" attack) being less feasible to exploit. But PBKDF2 is not secured enough. The drawback is carried out when implemented, including it does not support parallelism (only a part of resources available in a CPU can be employed for defensive use, slows down defender), it does not require memory needs (small circuit with low RAM can be also possible), and the number of process iteration demanded (min is 1000) is not appropriate to current GPUs. Obviously, attackers may easier to deploy cheap attacks methods, like brute force or guessing attack to explore the key (password).

2.3. bcrypt

bcrypt is a KDF scheme based on Blowfish algorithm [32], proposed at USENIX 1999. The salt is 16 bytes for brute force and "rainbow tables" resistance. The password can up to 56 bytes, while the iteration is a power of 2. Although having some difference for better performance, this still exist similar weaknesses to PBKDF2. bcrypt does not support parallelism that affects to defense probability. Low memory needs (only 4KB) are not capable to avoid attackers from effective cheap attacks (no need to provide more RAMs). ASIC/GPU attacks resistance may be better than PBKDF2, but not secure enough. Luckily, in some case, bcrypt's memory access pattern requirements help it fairly oppose against FPGA attacks.

2.4. scrypt

scrypt was officially published by IETF as an Internet Draft in 2012, then intended as a standard RFC in 2015 [33]. Upgraded from two previous schemes above which have no memory needs or low memory demands, scrypt requires large amounts of memory. Thence, it provides stronger resistance against time-memory trade-off attacks. Example, with brute force, the cost to employ is 4000 times larger than bcrypt and 2000 times larger than PBKDF2. But, the large memory needs (more RAMs used) leads to the poor scalability when many processes are run concurrently. This point allows attackers can employ DoS attacks on servers which have to handle a huge task frequently and constantly. The general construction of scrypt algorithm is given in **Figure 3**.

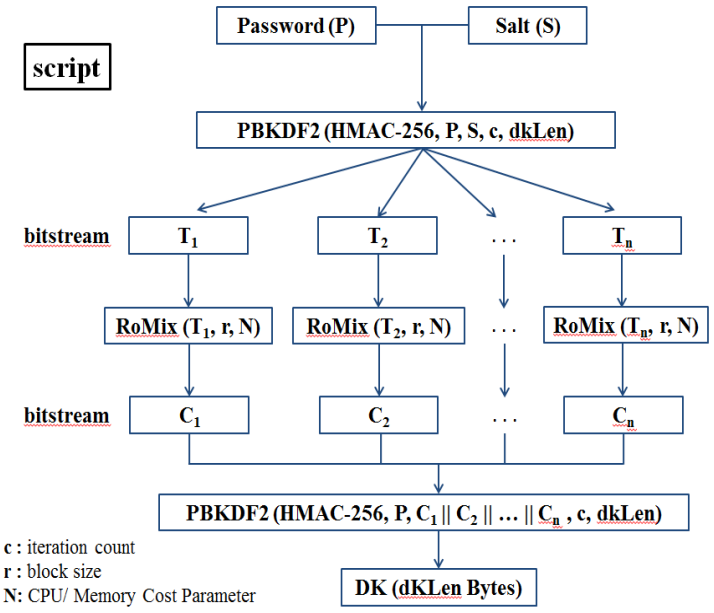

Figure 3. The scrypt algorithm.

We assume the strong security rate of password schemes is numbered from (1) to (7) in increasing order of reliable secure. General security evaluation of some schemes is defined in **Table 1**.

Table 1. Password hashing schemes' security evaluation.

Scheme Name	*Orig. Country*	*Related Cryptography*	*Memory Demands*	*GPU resist.*	*FPGA/ASIC resist.*	*Trade-off resist.*
PBKDF2-HMAC-SHA-1	USA	HMAC-SHA-1	-	(1)	(1)	-
PBKDF2-HMAC-SHA-256	USA	HMAC-SHA-256	-	(2)	(2)	-
Sha256crypt	USA	SHA-256	-	(3)	(3)	-
PBKDF2-HMAC-SHA-512	USA	HMAC-SHA-512	-	(4)	(4)	-
Sha512crypt	USA	SHA-512	-	(5)	(5)	-
bcrypt	USA	Blowfish	4KB	(6)	(6)	(6)
scrypt	Canada	PBKDF2, Salsa20/8	1GB	(7)	(7)	(7)

3. Password Hashing in the Present

This chapter focuses on the recent Password Hashing Competition (PHC), a main competition which attracted and promoted more researchers' interest to create stronger passwords. This consists of most remarkable secure password hashing algorithms at the present time. The PHC winner, Argon2, was proposed by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich from University of Luxembourg. This scheme provides high performance in hardware/software implementations and high resistance to most known attacks, along with 4 special finalists, Catena, Lyra2, Makwa, and yescrypt that offer more potential secure algorithms in password security part in further.

3.1. Password Hashing Competition (PHC)

Password Hashing Competition (PHC), continuing a series of several previous cryptographic competitions such as AES, eSTREAM, SHA-3, and even CEASAR, has been initiated organized in 2013 in order to identify new cryptographic primitives as standard for strong password protection. It covers a wide range of applications, services and devices. This is a part in attempts of constructing and designing the most ideal password hashing schemes, giving complementary secure password methods besides 3 classical common choices: PBKDF2, bcrypt, and scrypt. Due to the deadline in January, 2014, there were total 24 candidates submitted [34] and then 9 of them were selected as finalists in December, 2014. The PHC winner, Argon2, was announced in July, 2015, along with 4 other special outstanding schemes: Catena, Lyra2, Makwa, and yescrypt, which provides not only high performance in implementations but also high known attacks resistance.

Some general password hashing scheme characteristics and properties are required, include:

- An input password can be any length from 0 to 128 bytes
- A salt size is 16 bytes.
- An output can be up to 32 bytes.
- One or more cost parameters (effectiveness).
- Construction should be a random function (one-way transformation).
- Trade-off attacks resistance (brute force attack, dictionary attack, "rainbow tables" attack, etc.,)
- GPU, FPGA/ASIC attacks resistance.
- Client-independent updates (the ability of server to increase security parameters and update password hashes).

- Server relief technique (the most calculation of password hashing clients do, the least resource server use to reduce server load, DoS defense, maintain security of hash computation).
- Effective improvement in implementations (fast, easy).
- Flexible and scalable design.

3.1.1 Argon2

Argon2 [4] scheme was proposed by Biryukov, A., et al. from University of Luxembourg. It has two variants: Argon2d depends on memory access, be suitable for crypto-currencies and applications, and Argon2i which uses independent-memory access, is appropriate for password hashing and KDF-based constructions. Argon2i is slower than the other one when having memory access demands enough to resist time-memory trade-off attacks.

Within design, Argon2 using XOR operations, byte-based permutation, round function of AES structure with 128 bits fixed-key in 5 rounds, padding phase and compression function G. The permutation is based on the RC4 stream cipher's permutation which is data-dependent to protect against memory-saving attacks. Additional time-cost parameter presents the time for authentication (timing attacks defense). Moreover, Argon2 supports server relief technique and client-dependent updates. The construction seems effective in CPU performance while can process up to 32 parallel threads in employing. With GPU, due to large memory access use, Argon2 makes difficult (latency) in some constructions which do not support compatible memory. Both two versions of Argon2 provide high resistance against most known attacks (side-channel attack, timing attack, garbage-collector attacks, etc.).

3.1.2 Catena

Catena [11] is password-scrambling framework, proposed by Forler, C., et al. from Germany. It adapts a wide scale of architectures and supports flexible usage in multiple environments (low-memory devices, multi-cores CPUs, etc.). There are two instantiations: Catena-BRG (Bit-Reversal Graph) and Catena-DBG (Double-Butterfly Graph). Catena uses strong hash functions, like SHA-512 or BLAKE2b which are available for SIMD approach to provide GPU attacks resistance. It also enables client-dependent updates with addition parameters (time-cost and memory-cost parameters) and server relief technique. Above all, Catena provides high resistance to most known attacks (side channel attacks, timing attack, and GPU, FPGA/ASIC attacks).

3.1.3 Lyra2

Lyra2 [12] is a designed algorithm from Brazil, proposed by Simplicio Jr. M.A, et al. This scheme is password-scrambling framework, employing hash functions based on duplex sponge technique, and appropriate for KDF-based structures. The hash function used is BLAKE2b algorithm. Lyra2 requires large memory access (RAM) up to 1GB to store password hashing process. Additional parameter "basil" is also used as additional salt to resist collision attack. Lyra2 allows to process parallel instances on multi-cores CPUs, provides ability of attacks defense (brute force, password guessing, timing attack, etc.).

3.1.4 Makwa

Makwa [13] is designed suitable for password-hashing and KDF. This was proposed by Canadian researcher Pornin, T. It is based on big number of modular squaring like in RSA algorithm. To employ KDF, Makwa uses standard HMAC_DRBG combined with SHA-256 hash function. Additional time parameter is also presented while memory access demands are reasonable. The scheme supports server relief technique in processing big instances to increase attacks cost, and client-dependent updates. Makwa also seems provide high resistance against known attacks (side-channel attacks, CPU, FPGA/ASIC attacks, etc.).

3.1.5 yescrypt

yescrypt [14] was proposed by Peslyak, A., based on scrypt algorithm with tweaks. It upgrades some weaknesses from scrypt while increasing more parallel instances in processing, and supports SIMD. The ROMix algorithm is utilized for better implementation performance. A new algorithm (pwx-form) is replaced to the Salsa20/8 algorithm. yescrypt requires reasonable memory access needs, up the attacks complexity. Similar to scrypt, yescrypt provides high resistance against known attacks (side-channel attacks, CPU, FPGA/ASIC attacks, etc.).

3.1.6 battcrypt

battcrypt (Blowfish All The Things) [15] is based on utilized Blowfish algorithm combined with SHA-512 hash function. It was proposed by Thomas, S., which targeted server-side applications and suitable for web services. Additional memory-cost and time-cost parameters are executed. This scheme provides high performance while implemented on CPUs may faster than on GPUs. battcrypt also supports server relief technique, and presents attacks resistance (timing attacks, side-channel attacks, etc.)

3.1.7 Pufferfish

Pufferfish [18] is a password hashing scheme based on Blowfish and bcrypt's Eksblowfish algorithms, proposed by Gosney, J.M. It supports KDF, using HMAC-SHA-512 and password-dependent Sboxes. Additional memory-cost and time-cost parameters increase memory access demands and attacks complexity. Due to HMAC and SHA-512 within structure, Pufferfish enhances protection against known attacks (timing attacks, side-channel attacks, etc.).

3.1.8 POMELO

Pomelo [17] was proposed by Wu, H. from Singapore, has simple design with 3 main functions: one non-linear feedback function and two others provide random memory access demands. Additional memory and time cost parameters increase memory size and attacks complexity. This also supports client-dependent updates. Pomelo provides resistance against known attacks (side-channel attacks, timing attack, pre-image attack, low memory attack, GPU attack, etc.).

3.1.9 Parallel

Parallel [16] is a password hashing scheme, designed based on PBKDF algorithm with SHA-512 hash function. This was proposed by Thomas, S. for low memory applications. It does not require memory access demands (or low) that enables attackers to employ cheap effective attacks to exploit. This algorithm is very simple, easy to implement. Parallel also provides collision attack defense.

3.1.10 Gambit

Gambit [22], proposed by Pinter, K., is a sponge based, memory hard KDF scheme. This uses duplex sponge construction with large state and standard Keccak hash function. Gambit supports server relief technique and client-dependent updates, increasing attacks cost. It is not a strong secure algorithm since still need to be developed more.

3.1.11 Lanarea DF

Lanarea Derivation Function [23] is KDF-based construction with BLAKE2b hash function. It was proposed by Mubarak, H. from USA, targeted to make execution on GPGPUs and ASICs parallel to be hard. The output can be multiple of 32 bytes. Lanarea DF is suitable for server-client

environment, stream and long term encryption. This scheme is also a now strong secure design while it slows down memory usage.

3.1.12 MCS_PHS

MCS_PHS [24] is a password hashing and KDF-based scheme based on MCSSHA-8 hash algorithm, proposed by Maslennikov, M. from Russia. To employ KDF, it uses PBKDF_MCS algorithm- a combination of PBKDF and MCSSHA-8. It requires memory access demands of 1KB, with additional time-cost parameter to increase iteration count. Although having some upgrades from PBKDF, this algorithm is nearly similar to since the security mainly based on MCSSHA-8 which is not strong secure at all.

3.1.13 Omega Crypt (ocrypt)

Omega Crypt [25] is a password hashing scheme with KDF, proposed by Enright, B. This structure is based on scrypt algorithm, while the memory access demands is identified by ChaCha stream cipher output, instead of hash function. Another CubeHash algorithm is used to derive keys at the final step. The password is used both generate keys for stream cipher and initialize internal state. ocrypt provides resistance against attacks (GPU, FPGA/ASIC attacks, side-channel attack, etc.). However, due to predictable data-dependent branches and memory access needs, the performance and security is not clear and secure enough.

3.1.14 EARWORM

EARWORM [21] was proposed by Franke, D., targeted to server-side applications. This is designed for low-time cost applications. It is based on utilized AES round function along with PBKDF2-HMAC-SHA-256 algorithm. EARWORM requires large memory access demands, makes it un-appropriate to low-memory environments. It is not a KDF-based support. In addition, this algorithm enables to process parallel multi-cores on CPUs, but low performance with GPUs while large memory needs lead to high latency. EARWORM does not provide strong protection (not resist second pre-image attack, not construct as KDF-based).

3.1.15 PolyPassHash

PolyPassHash [26] is not a password scrambling framework, construct password hashing scheme with PolyHashing algorithm. This was proposed by Cappos, J., targeted to server-side applications. It is designed to protect an individual password hash among a lot of password hashes produced. PolyPassHash requires memory access of 1KB. It recovers symmetric keys to encrypt password but, not support KDF-based. Similar to EARWORM, this algorithm does not provide strong protection.

3.1.6 Centrifuge

Alvarez, R. proposed Centrifuge [20] as a password hashing scheme, based on a pseudorandom number generator, SHA-512 hash function, and a substitution S-box. Within design, this structure uses AES-256 algorithm as internal function in CFB mode. Additional memory-cost and time-cost parameters are given, can be higher than 64. It provides resistance against GPU, FPGA/ASIC attacks. However, the random memory access needs give slow performance in memory usage.

3.1.17 AntCrypt

AntCrypt [19] was proposed by Durmuth, M., et al., based on SHA-512 hash function. The salt size is 16 bytes. Additional memory-cost and time-cost parameters define consumed memory needed and iteration count, respectively. It supports client-dependent updates. AntCrypt requires memory access demands of 32KB. This scheme provides resistance against CPU, GPU attacks, FPGA/ASIC attacks. However, due to float-point arithmetic reduces portability and

reproducibility, and un-appropriate data-dependent branches make AntCrypt seem be not strong protection.

3.1.18 Rig

Rig [27] is a password hashing scheme, proposed by Chang, D., et al., based on BLAKE2b hash function. This supports client-dependent updates and server relief technique within structure. It requires large memory access needs (15MB), some case be impractical in some environments. Rig provides resistance against attacks (timing attack, memory-free attack, DoS attack, etc.). Unfortunately, the specification and code of this is at low quality (errors, bugs).

3.1.19 Schvrch

Schvrch [28] is a password scrambling scheme, proposed by Vuckovac, K. It designs a new class of hash functions. This requires memory access demands of 8MB. Additional memory-cost and time-cost parameters are integrated to internal state. Schvrch employs as a Cellular Automation. However, it does not provide strong protection for password.

3.1.20 Tortuga

Tortuga [29] was proposed by Teath Sch., based on a sponge structure in Feistel network (Turtle algorithm). It also supports KDF-based scheme, achieves memory-hardness. The algorithm and implementation of Tortuga have not tested in practice, so cannot evaluate the security rate for password protection.

3.1.21 TwoCats

TwoCats [30] is a password hashing scheme proposed by Cox, B., based on various hash functions (BLAKE2b, BLAKE2s, SHA-256, SHA-512, etc.). The structure is similar to scrypt. This support KDF-based construction, server relief technique and client-dependent updates. TwoCats is designed for mobile, web server, and embedded constrained environments. It provides resistance against attacks (GPU attack, FPGA/ASIC attacks, etc.). However, the design is not totally clear, makes difficult to understand and implement.

3.1.22 Yarn

Yarn [31] is a memory-hard function, proposed by Kapun, E., which based on AES round function and BLAKE2b hash function. Additional memory-cost parameters are provided to adjust memory parallelism. Due to large memory consumed, Yarn presents low performance on GPUs, FPGAs, and ASICs. This makes this scheme is less secure for password protection.

3.2. Security Evaluation

Strong password hashing scheme need to support KDF-based construction, random one-way transformation function (hash functions, stream functions, etc.), most known attacks resistance (trade-off attacks defense, CPU, GPU attacks, FPGA/ASIC attacks, side-channel attacks, timing attack, etc.).

Table 2. defines general security evaluation of PHC candidates.

Table 2. PHC candidates' security evaluation.

Scheme Name	*Orig. Country*	*Memory Demands*	*GPU resist.*	*FPGA/ASIC resist.*	*Trade-off resist.*
Argon2	Luxembourg	1KB/1GB	✓	✓	✓
Catena	Germany	8MB	✓	✓	✓
Lyra2	Brazil	8MB/1GB	✓	✓	✓

Makwa	Canada	335KB	✓	✓	-
yescrypt	Russia	44KB/3MB	✓	✓	✓
AntCrypt	Germany	32KB	-	-	-
battcrypt	USA	18KB/128MB	✓	-	-
Centrifuge	Spain	56KB/2MB	✓	✓	✓
EARWORM	USA	2GB (ROM)	✓	✓	✓
Gambit	Hungary	50MB	-	-	✓
Lanarea	USA	256bytes	✓	✓	-
MCS_PHS	Russia	1KB	-	-	✓
Omega Crypt	USA	1MB/1GB	-	-	-
Parallel	USA	-	✓	✓	-
PolyPassHash	USA	1KB			
POMELO	Singapore	1KB/8GB	✓	-	✓
Pufferfish	USA	4KB/16KB	✓	-	-
Rig	India	15MB	-	-	✓
Schvrch	Croatia	4KB/8MB	-	-	×
Tortuga	Haiti	32KB	-	-	✓
TwoCats	USA	1KB/192GB	✓	✓	✓
Yarn	Russia	16bytes/192GB	×	×	×

4. Password Hashing in the Future

According to the development of password hashing techniques through the time mentioned in two previous chapters, in this chapter, we will summary and explore in part next steps of password hashing directions in the future. The contents will be discussed include some security parameters, criteria, challenges, the way how to design and construct a good and even, an ideal password hashing scheme in attempts to provide the most secure password security. At last, the prediction of future information security is given in wide point of view which is a combination of all factors and matters considered through the whole paper.

4.1. Security Challenges

New password hashing constructions face lots of challenges come from software/hardware implementation requirements, security evaluation requirements, strong algorithm to use and specially, attackers' possibility to break security rate.

Designing a password hashing scheme needs to support Key Derivation Function (KDF-based). This exist a drawback for authentication is that to use a lot of RAM fast, it is necessary to close to full memory bandwidth. But, since many concurrent instances are run constantly, it leads to poor scalability and flexibility. In this case, we have to choose between using more RAM per instance (low performance) and using multi-cores CPU (lower RAM setting per instance).

To evaluate software/hardware implementation while design a password hashing scheme is not always an easy challenge. We have to ensure minimized efficiency on GPUs and FPGAs, maximized efficiency on CPUs. This method need to parallelism of multiple instances concurrently. How to approach implementation in general-purpose in which not slow down or present low performance when employing? And also have to care about security properties.

Cryptographic algorithms are another challenge. Which techniques, algorithms should be chosen to provide strong password protection against attacks. The scheme designed targeted to all applications from servers, clients, or both? Some hash functions are fast, but not offer strong secure, while others are slow. With different environments, algorithms need to be designed and measured deeply.

Besides classical attacks, attackers are rising more their methods and levels in breaking password. In all way, a strong password hashing scheme always have to resist and avoid as much as possible

attacks. In some cases, attackers support to find any trick or weakness from designs, help designer to handle and fix it within their structures.

4.2. A Good Password Hashing Scheme

A "good" password hashing scheme must meet 2 concurrent needs: both secure password hash and utilize computation required for secure storing password.

Some evaluation criteria can be considered:

- Minimal Input / Output Requirements (similar to PHC: 0-128bytes password, 16 bytes salt, 16 bytes hash, cost parameter, etc.)
- Pseudorandom function behavior
- Minimal speedup to exhaust password cracking
- Effective cost parameters (memory, time)
- Flexibility and scalability
- Side-channel attacks resistance
- Timing attack, low-memory attacks resistance
- Trade-off attacks resistance
- CPU,GPU,FPGA,ASIC attacks resistance
- Clarity, conciseness in specification
- High performance in implementation
- Wide range adaption on various devices, environments

4.3. Password Hashing is still the Future of Information Security?

Till now, password hashing technique is the most secure and effective to provide password protection. After PHC, we have an overview of password security with many potential algorithms achieved. Argon2, Catena, Lyra2, and other schemes present both their high performance in implementation and high resistance against attacks. It shows the importance of strong password and promotes secure password construction. A good password hashing structure can affect much in information security, dedicate to a "clean and safe" information world in the future.

5. Conclusion

In this paper, a general survey on password hashing techniques from the beginning to present has been presented to provide an overview of this information security part. We paid our main attention to the Password Hashing Competition (PHC) with its remarkable potential candidates. Argon2, the algorithm was announced as final PHC winner which seems the most secure and effective password hashing construction while offering high performance in implementations and high defense to known attacks, along with other outstanding candidates that are recognized and recommended using rather than existing password hashing schemes. Total 24 PHC submitted algorithms at initial (except 2 withdrawn candidates Catfish and M3lcrypt) were measured based on major security criteria and challenges in order to evaluate the weaknesses, limits as well as advantages of each. Finally, we contribute our works to further research by summary next steps of prospective information security in general. Since password stills plays an important role in protecting information now and later, advance password hashing techniques will be continue a reliable method helps to produce strong and "clear" passwords in the future. Besides cryptographic hashing, it is also expected to have more new solutions to fulfill the demands of the best performance on wide environments and the best protection against attackers nowadays.

6. References

[1] Forler, C., List, E., Lucks, S., Wenzel, J., "Overview of the Candidates for the Password Hashing Competition - And Their Resistance against Garbage-Collector Attacks," *Cryptology ePrint Archive, Report 2014/881*, 2014.

[2] Broz, M., "Password Hashing Competition second round candidates – Tests Report (v3)," *Technical Report, https://github.com/mbroz/PHCtest/raw/master/output/phc_round2.pdf*, 2015.

[3] Password Hashing Competition (PHC), *https://password-hashing.net*, 2013.

[4] Biryukov, A., Khovratovich, D., "Argon and Argon2: Password Hashing Scheme," *https://password-hashing.net/submissions/specs/Argon-v2.pdf*, 2015.

[5] "NIST: Recommendation for Password-Based Key Derivation," *NIST Special Publication 800-132, http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf*, 2010.

[6] Orman, H., "*Twelve Random Characters: Passwords in the Era of Massive Parallelism*" *IEEE Internet Computing, Vol. 17, Issue 5, pp. 91-94*, 2013.

[7] Percival, C., "Stronger Key Derivation via Sequential Memory-Hard Functions," *BSDCan09*, 2009.

[8] Kaliski, B., "RFC2898 - PKCS#5: Password-Based Cryptography Specification Version 2.0," *Technical Report, IETF*, 2000.

[9] RSA Laboratories, "PKCS#5: Password-Based Cryptographic Standard, v2.0," *http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-5-password-based-cryptography-standard.htm*, 2000.

[10] Provos N., Mazires D., "A Future-Adaptable Password Scheme," *USENIX Annual Technical Conference, pp. 8192*, 1999.

[11] Forler, C., Lucks, S., and Wenzel, J., "The Catena Password-Scrambling Framework," *https://password-hashing.net/submissions/specs/Catena-v5.pdf*, 2015.

[12] Simplicio Jr. M.A., Almeida, L.C., Andrade, E.R., Barreto, P.S.L.M., and Marcos., "The Lyra2 Reference Guide," *https://password-hashing.net/submissions/specs/Lyra2-v3.pdf*, 2015.

[13] Pornin, T., "The MAKWA Password Hashing Function," *https://password-hashing.net/submissions/specs/Makwa-v1.pdf*, 2015.

[14] Peslyak, A., "yescrypt – a Password Hashing Competition Submission," *https://password-hashing.net/submissions/specs/yescrypt-v2.pdf*, 2015.

[15] Thomas, S., "battcrypt (Blowfish All the Things)," *https://password-hashing.net/submissions/specs/battcrypt-v0.pdf*, 2014.

[16] Thomas, S., "PPBKDF (Parallel Password Based Key Derivation Function)," *https://password-hashing.net/submissions/specs/Parallel-v1.pdf*, 2014.

[17] Wu, H., "POMELO: A Password Hashing Algorithm," *https://password-hashing.net/submissions/specs/POMELO-v3.pdf*, 2015.

[18] Gosney, J.M., "The Pufferfish Password Hashing Scheme," *https://password-hashing.net/submissions/specs/Pufferfish-v1.pdf*, 2015.

[19] Durmuth, M., Zimmermann, R., "AntCrypt: Proposal for the Password Hashing Competition," *https://password-hashing.net/submissions/specs/AntCrypt-v0.pdf*, 2014.

[20] Alvarez, R., "CENTRIFUGE: A Password Hashing Algorithm," *https://password-hashing.net/submissions/specs/Centrifuge-v0.pdf*, 2014.

[21] Franke, D., "The EARWORM Password Hashing Algorithm," *https://password-hashing.net/submissions/specs/EARWORM-v0.pdf*, 2014.

[22] Pinter, K., "Gambit – A Sponge Based, Memory Hard Key Derivation Function," *https://password-hashing.net/submissions/specs/Gambit-v1.pdf*, 2014.

[23] Mubarak, H., "Lanarea DF," *https://password-hashing.net/submissions/specs/Lanarea-v0.pdf*, 2014.

[24] Maslennikov, M., "Password Hashing Scheme MCS_PHS," *https://password-hashing.net/submissions/specs/MCS_PHS-v2.pdf*, 2015.

[25] Enright, B., "Omega Crypt," *https://password-hashing.net/submissions/specs/OmegaCrypt-v0.pdf*, 2014.

[26] Cappos, J., "PolyPassHash: Protecting Passwords In the Event of a Password File Disclosure," *https://password-hashing.net/submissions/specs/PolyPassHash-v1.pdf*, 2014.

[27] Chang, D., Jati, A., Mishra, S., Sanadhya, S.M., "RIG: A Simple, Secure and Flexible Design for Password Hashing," *https://password-hashing.net/submissions/specs/RIG-v2.pdf*, 2014.
[28] Vuckovac, K., "Schvrch," *https://password-hashing.net/submissions/specs/Schvrch-v0.pdf*, 2014.
[29] Teath Sch., "TORTUGA – Password Hashing Based on the Turtle Algorithm," *https://password-hashing.net/submissions/specs/Tortuga-v0.pdf*, 2014.
[30] Cox, B., "TwoCats (and SkinnyCat): A Computer Time and Sequential Memory Hard Password Hashing Scheme," *https://password-hashing.net/submissions/specs/TwoCats-v0.pdf*, 2014.
[31] Kapun, E., "Yarn Password Hashing Function," *https://password-hashing.net/submissions/specs/Yarn-v2.pdf*, 2014.
[32] Schneier, B., "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Fast Software Encryption, Springer, LNCS, Vol. 809, pp. 191-204*, 1994.
[33] Percival, C., Josefsson, S., "The scrypt Password-Based Key Derivation Function," *IETF, https://tools.ietf.org/html/draft-josefsson-scrypt-kdf-02*, 2015.
[34] Password Hashing Competition (PHC): Candidates, *https://password-hashing.net/candidates.html*, (2014).