IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 1, Feb-Mar, 2014
ISSN: 2320 - 8791
www.ijreat.org

Strong Password Hash Secure Scheme Using Image Fusion

[1]Ms.Snehal A. Mahajan, [2]Ms.Anjali S. Mahajan, [3]Ms.Richa Sharma

[1]PG Student,CSE Department,
G.H.Raisoni Academy College of Engineering & Technology, Nagpur

[2]HOD, CSE Department,
Priydarshani Institute of Technology, Nagpur

[3]Professor, CSE Department,
G.H. Raisoni Academy College of Engineering & Technology, Nagpur

Abstract:
This paper presents a robust and secure image hash algorithm. The algorithm extracts robust image features in the Radon transform domain. A randomization mechanism is designed to achieve good discrimination and security. The hash value is dependent on a secret key. We evaluate the performance of the proposed algorithm and compare the results with those of one existing Radon transform- based algorithm. We show that the proposed algorithm has good robustness against content preserving distortion. It with stands JPEG compression, filtering, noise addition as well as moderate geometrical distortions. Additionally, we achieve improved performance in terms of discrimination, sensitivity to malicious tampering and receiver operating characteristics. We also analyze the security of the proposed algorithm using differential entropy and confusion/diffusion capabilities. Simulation shows that the proposed algorithm well satisfies these metrics.

Keywords:
Hash Value, Hash Strength, Image Fusion, Visual Cryptography, Permutation, PSNR, MSE

1. Introduction:

Password strength is a measure of the effectiveness of a password in resisting guessing and brute-force attacks. In its usual form, it estimates how many trials an attacker who does not have direct access to the password would need, on average, to guess it correctly. The strength of a password is a function of length, complexity and unpredictability. Using strong passwords lowers overall risk of a security breach, but strong passwords do not replace the need for other effective security controls. The effectiveness of a password of a given strength is strongly determined by the design and implementation of the authentication system software, particularly how frequently password guesses can be tested by an attacker and how securely information on user passwords is stored and transmitted. Risks are also posed by several means of breaching computer security which are unrelated to password strength.

A strong password:

- has at least 15 characters;
- has uppercase letters;
- has lowercase letters;
- has numbers;
- Has symbols, such as `! " ? $? % ^ & * ( ) _ - + = { [}] : ; @ ' ~# | \ < , > . ? /

- Is not like your previous passwords;
- is not your name;
- is not your login;
- is not your friend's name;
- is not your family member's name;
- is not a dictionary word;
- is not a common name;
- is not a keyboard pattern, such as qwerty, asdfghjkl, or 12345678.

1.1 Password Strength:

There are two factors to consider in determining password strength: the average number of guesses the attacker must test to find the correct password and the ease with which an attacker can check the validity of each guessed password. The first factor is determined by how long the password is how large a set of characters or symbols it is drawn from and whether the password is created randomly or by a more predictable process. Users of password-protected resources often have control of this factor. The second factor is determined by how the password is stored and used. This factor is determined by the design of the password system and beyond control of the user. The rate at which an attacker can submit guessed passwords to the system is a key factor in determining system security. Some systems impose a time-out of several seconds after a small number (e.g., three) of failed password entry attempts. In the absence of other vulnerabilities, such systems can be effectively secure with relatively simple passwords. However the system must store information about the user passwords in some form and if that information is stolen, say by breaching system security, the user passwords can be at risk.

1.2 Password Creation:

Passwords are created either automatically (using randomizing equipment) or by a human. While the strength of randomly chosen passwords against a brute force attack can be calculated with precision, determining the strength of human-generated passwords is challenging and the latter case is more common. Typically, humans are to choose a password, sometimes guided by suggestions or improvements in computing technology keep increasing the rate at which guessed passwords can be tested.

1.3 Password Guess Validation:

Systems that use passwords for authentication must have some way to check any password entered to gain access. If the valid passwords are simply stored in a system file or database, an attacker who gains sufficient access to the system will obtain all user passwords, giving the attacker access to all accounts on the attacked system, and possibly other systems where users employ the same or similar passwords. One way to reduce this risk is to store only a cryptographic hash of each password instead of the password itself. Standard cryptographic hashes, such as the Secure Hash Algorithm series, are very hard to reverse, so an attacker who gets hold of the hash value cannot directly recover the password. However, knowledge of the hash value lets the attacker quickly test guesses offline. Password cracking programs are widely available that will test large number of trial passwords against a purloined cryptographic hash. Crack a 10 letter single-case password in one day. Note that the work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs. Special key stretching hashes are available that take a relatively long time to compute, reducing the rate at which guessing can take place. Although it is considered best practice to use key stretching, many common systems do not. Another situation where quick guessing is possible is when the password is used to form a cryptographic key. In such cases, an attacker can quickly check to see if a guessed password successfully decodes encrypted data. For example, one commercial product claims to test 103,000

WPA PSK passwords per second. If a password system only stores the hash of the password, an attacker can pre-compute hash values for common passwords variants and for all passwords shorter than a certain length, allowing very rapid recovery of the password once its hash is obtained. Very long lists of pre-computed password hashes can be efficiently stored using rainbow tables. This method of attack can be foiled by storing a random value, called a cryptographic salt, along with the password. The salt is combined with the password when computing the hash, so an attacker precomputing a rainbow table would have to store for each password its hash with every possible salt value. This becomes infeasible if the salt has a big enough range, say a 32-bit number. Unfortunately, many authentication systems in common use do not employ salt and rainbow tables are available on the Internet for several such systems.

1.4 Problem Definition:

Sequence of characters letters, numbers, symbols used as a secret key for accessing a computer system or network. Passwords are used also for authentication, validation, and verification in electronic commerce.

2. Literature Survey:

2.1 Human-generated passwords:

People are notoriously remiss at achieving sufficient entropy to produce satisfactory passwords. Some stage magicians exploit this inability for amusement, in a minor way, by divining supposed random choices (of numbers, say) made by audience members. Thus, in one analysis of over 3 million eight-character passwords, the letter "e" was used over 1.5 million times, while the letter "f" was only used 250,000 times. A uniform distribution would have had each character being used about 900,000 times. The most common number used is "1", whereas the most common letters are a, e, o, and r. Users rarely makes full use of larger characters sets in forming passwords. For example, hacking results obtained from a MySpace phishing scheme in 2006 revealed 34,000 passwords, of which only 8.3% used mixed case, numbers, and symbols. Note that the full strength associated with using the entire ASCII character set (numerals, mixed case letters and special characters) is only achieved if each character in the password is chosen randomly from that set. Capitalizing a letter and adding a couple of numbers and a special character to a password will not achieve the same strength. If the numbers and special character are added in predictable ways, say at the beginning and end of the password, they could even lower password strength compared to an all letter random password of the same length.

2.2 NIST Special Publication 800-63:

NIST Special Publication 800-63 suggests the following scheme to roughly estimate the entropy of human-generated passwords:

- The entropy of the next seven characters are two bits per character;
- The entropy of the first character is four bits;
- The ninth through the twentieth character has 1.5 bits of entropy per character;
- Characters 21 and above have one bit of entropy per character.

A "bonus" of six bits is added if both upper case letters and non- alphabetic characters are used. A "bonus" of six bits is added for passwords of length 1 through 19 characters following an extensive dictionary check to ensure the password is not contained within a large dictionary. Passwords of 20 characters or more do not receive this bonus because it is assumed they are pass-phrases consisting of multiple dictionary words. Using this scheme, an eight-character human-selected password without upper case letters and non-alphabetic characters is estimated to have 18 bits of entropy. The NIST publication concedes that at the time of development, little information was available on the real world selection of passwords. Later research

into human-selected password entropy using newly available real world data has demonstrated that the NIST scheme does not provide a valid metric for entropy estimation of human-selected passwords.

2.3. Usability and Implementation Considerations:

Because national keyboard implementations vary, not all 94 ASCII printable characters can be used everywhere. This can present a problem to an international traveler who wished to log into remote system using a keyboard on a local computer. *See* keyboard layout. Many hand held devices, such as tablet computers and smart phones, require complex shift sequences to enter special characters. Authentication programs vary in which characters they allow in passwords. Some do not recognize case differences (e.g., the upper- case "E" is considered equivalent to the lower-case "e"), others prohibit some of the other symbols. In the past few decades, systems have permitted more characters in passwords, but limitations still exist. Systems also vary in the maximum length of passwords allowed.

2.4 Examples of Weak Passwords:

As with any security measure, passwords vary in effectiveness (i.e., strength); some are weaker than others. For example, the difference in weakness between a dictionary word and a word with obfuscation (i.e., letters in the password are substituted by, say, numbers— a common approach) may cost a password cracking device a few more seconds– this adds little strength. The examples below illustrate various ways weak passwords might be constructed, all of which are based on simple patterns which result in extremely low entropy, allowing them to be tested automatically at high speeds.

2.5 Guidelines for Strong Passwords:

Guidelines for choosing good passwords are designed to make passwords less easily discovered by intelligent guessing. Common guidelines include:

- A minimum password length of 12 to 14 characters if permitted
- Generating passwords randomly where feasible
- Avoiding passwords based on repetition, dictionary words, letter or number sequences, usernames, relative or pet names, romantic links (current or past), or biographical information (e.g., ID numbers, ancestors' names or dates).
- Including numbers, and symbols in passwords if allowed by the be changed at installation time): password, default, admin, guest, etc. Lists of default passwords are widely available on the internet
- Dictionary words: chameleon, RedSox, sandbags, bunnyhop!, IntenseCrabtree, etc., including words in non- English dictionaries.
- Words with numbers appended: password1, deer2000, john1234, etc., can be easily tested automatically with little lost time.
- Words with simple obfuscation: p@ssw0rd, l33th4x0r, g0ldf1sh, etc., can be tested automatically with little additional effort. For example a domain administrator password compromised in the DigiNotar attack was reportedly Pr0d@dm1n.
- Doubled words: crab crab, stop stop, treetree, passpass, etc.
- Common sequences from a keyboard row: qwerty, 12345, asdfgh, fred, etc.
- Numeric sequences based on well known numbers such as 911314159...), or 27182...etc.
- Identifiers: jsmith123, 1/1/1970, 555–1234, your username, etc.
- Anything personally related to an individual: license plate number, Social Security number, current or past telephone number, student ID, address, birthday, sports team, relative's or pet's names/nicknames/birthdays/initials, etc., can easily be tested automatically after a simple investigation of person's details. There are many other ways a password can

be weak, corresponding to the strengths of various attack schemes; the core principle is that a password should have high entropy (usually taken to be equivalent to randomness) and *not* be readily derivable by any "clever" pattern, nor should passwords be mixed with information identifying the user. On-line services often provide a restore password function that a hacker can figure out and by doing so bypass a password. Choosing hard to guess restore password questions can further secure the password.

2.6 Password Policy:

A password policy is a guide to choosing satisfactory passwords. Some are controversial. They are usually intended to:

- assist users in choosing strong passwords
- ensure the passwords are suited to the target population recommendations to users with regard to the handling of their passwords a requirement to change any password which has been lost or compromised, and perhaps that no password be used longer than a limited time some policies prescribe the pattern of characters which passwords must contain.
- For example, password expiration is often covered by password policies. Password expiration serves two purposes:
- 1) if the time to crack a password is estimated to be 100 days, password expiration times fewer than 100 days may help ensure insufficient time for an attacker.
- 2) if a password has been compromised, requiring it to be changed regularly should limit the access time for the attacker Some argue that password expirations have become obsolete, since:
- Asking users to change passwords frequently encourages simple, weak passwords. If one has a truly strong password, there is little point in changing it. Changing passwords which are already strong introduces risk that the new password may be less strong. A compromised password is likely to be used immediately by an attacker to install a backdoor, often via privilege escalation. Once this is accomplished, password changes won't prevent future attacker access. Mathematically it doesn't gain much security at all. moving from never changing one's password to changing the password on every authenticate attempt (pass *or* fail attempts) only doubles the number of attempts the attacker must make on average before guessing the password in a brute force attack - one gains *much* more security just increasing the password length by one character than changing the password on every use.

2.7 Creating and Handling Passwords:

The hardest passwords to crack, for a given length and character set, are random character strings; if long enough they resist brute force attacks (because there are many characters) and guessing attacks (due to high entropy). However, such passwords are typically the hardest to remember. The imposition of a requirement for such passwords in a password policy may encourage users to write them down, store them in PDAs or cellphones, or share them with others as a safeguard against memory failure. Some people consider each of these user resorts to increase security risks. Others suggest the absurdity of expecting users to remember distinct complex passwords for each of the dozens of accounts they access.

- A training program. Also, updated training for those who fail to follow the password policy (lost passwords, inadequate passwords, etc.).
- Rewarding strong password users by reducing the rate, or eliminating altogether, the need for password changes (password expiration). The strength of user-chosen passwords can be estimated by automatic programs which inspect and evaluate proposed passwords, when setting or changing a password.
- Displaying to each user the last login date and time in the hope that the user may notice unauthorized access, suggesting a compromised password.

- Allowing users to reset their passwords via an automatic system, which reduces help desk call volume. However, some systems are themselves insecure; for instance, easily guessed or researched answers to password reset questions bypass the advantages of a strong password system.
- Using randomly generated passwords that do not allow users to choose their own passwords, or at least offering randomly generated passwords as an option.

2.8 Memory Techniques:

Password policies sometimes suggest memory techniques to assist remembering passwords:

2.8.1 mnemonic passwords: Some users develop mnemonic phrases and use them to generate high-entropy (more or less random) passwords which are nevertheless relatively easy for the user to remember. For instance, the first letter of each word in a memorable phrase. Silly ones are possibly more memorable. Another way to make random-appearing passwords more memorable is to use random words or syllables instead of randomly chosen letters.

- **2.8.2 after-the-fact mnemonics**: After the password has been established, invent a mnemonic that fits. It does not have to be reasonable or sensible, only memorable. This allows passwords to be random.
- **2.8.3 password patterns**: Any pattern in a password makes guessing (automated or not) easier and reduces an attacker's work factor. For example, passwords of the following case-insensitive form: consonant, vowel, consonant, consonant, vowel, consonant, number, number (for example *pinray45*) are called Environ passwords The pattern of alternating vowel and consonant characters was intended to make passwords more likely to be pronounceable and thus more memorable.

2.9 Password Strength Advisers:

Several web sites and some standalone programs meant to be run without a network connection on a local machine, offer automated tests of password strength adequacy. They are problematic. Any network based checking necessarily involves submitting one's password to a purpose declared system somewhere. Doing so eases an attacker's problem very considerably; the relevant network traffic is identifiable as passwords saving much sifting effort, authentication of network connection problems permit authentication problems (e.g., site spoofing) which are lessened for equivalent programs running on local computers.

3.Algorithm:

1. Select Images from directory.
2. Fused an Image.
3. Select Key file
4. Segment key file into multiple segments.
5. Select Segments randomly or manually to create key file
6. Select Pixels of Fused Image as per Combined Segmented Image.
7. Convert Pixels value to Decimal Values.
8. Convert decimal Values to ASCII character Values.
9. Remove redundancy.
10. Check Password Strength
11. If Password (Hash) is strong, then go to step 12 else go to step
12. Create Backup.
13. Stop

3.1 Proposed Strong Password Generator Scheme Will Works as Follows:

1. Select input Images which may be of any type like RGB, Gray and Binary etc.
2. We perform Image fusion algorithm that combines all selected images into a single Image. Significance of image fusion algorithm is only to avoid the dependency of generated password on single image. Image fusion modifies input image pixels & at the result end, we find two images are mixed up. Let us consider the following example of two image fusion (RGB Format)
3. Once the image are fused, we will apply two shares Visual Cryptography that Encrypt

the image & converts it into unreadable format, result of visual cryptography becomes as follow:

4.The Cryptographic image is unreadable in format that's why an intruder will find difficulty in reading Plain image for password decryption. Cryptographic image contain a decimal pixel value either 0 or 255.

5. Crypto image is a input to our Proposed Password Generator Algorithm we choose the pixels from Crypto image based on the values of Keys, we suggest the multiple key selection to create more patterns of selection. Finally we assemble all these selected pixels into a single dimensional array which we will divide into 04 sections that is Digits, Characters, Special Symbols, & Special Character.

6. Strong password definition says that, Password should contain Digits, Characters, Special Symbols, & Special Characters and it should not be breakable by Non of the Brilliant Intruder easily, in proposed work; we will try to Mix up all the generated sections with permutations so that every time & in every round an Unique Password will generate.

4. Conclusion:

In this work, we propose a robust and secure image hash algorithm. The algorithm extracts image features in the Radon transform domain. A randomization mechanism is incorporated to make the hash output dependent on a secret key. It is resilient to filtering, JPEG compression, and noise addition. It is also robust to moderate geometrical distortions including rotation and cropping. The proposed algorithm achieves significant improvement to the well-known RASH algorithm.It has better discrimination and higher sensitivity to malicious tampering than RASH, which leads to a better operating characteristic. The key-dependent feature also makes it suitable for a wider range of applications. The security of the algorithm is evaluated in terms of differential entropy and confusion/diffusion capabilities. Good security is confirmed by both metrics.In the future, we plan to improve the proposed algorithm by detecting several geometric distortions (e.g. scaling and cropping) before computing the hash distance.This will further enhance robustness. More security metrics will be taken into account.

5.Reference:

[1] Venkatesan, R., Koon, S., Jakubowski, M., Moulin, P.: Robust image hashing. In:Proceedings of IEEE International Conference on Image Processing Vol.3, pp. 664666.(2000)

[2] Swaminathan, A., Mao,Y.,Wu,M.: Robust and secure image hashing. In: IEEE Transactions on Information Forensics and Security, Vol.1, No. 2. (June 2006)

[3] Mihcak, K., Venkatesan, R.: New iterative geometric methods for robust perceptual image hashing. In: In Proceedings of the Workshop on Security and Privacy in Digital Rights Management. (2001)

[4] Lefebvre, F., Macq, B., Legat, J.: Rash: Radon soft hash algorithm. In: Proceedings of the European Signal Processing Conference, Toulouse, France. (Sep. 2002)

[5] Fridrich, J., Goljan, M.: Robust hash functions for digital watermarking. In: Proceedings of the The International Conference on IT: Coding and Computing. (2000)

[6] Mao, Y., Wu, M.: Unicity distance of robust image hashing. In: IEEE Transactions on Information Forensics and Security, Vol. 2, No. 3. (Sep. 2007)