Protection against Significant Online Attacks on Login Password

Srinivasa Rao Morla, Padavala Sai Prasad

Dept. of CSE, Eluru College of Engineering and Technology, Eluru, AP, India

Abstract
Users are typically authenticated by their passwords. Because people are known to choose convenient passwords, which tend to be easy to guess, authentication protocols have been developed that protect user passwords from guessing attacks. Brute force and dictionary attacks on password-only remote login services are now widespread and ever increasing. Enabling convenient login for legitimate users while preventing such attacks is a difficult problem. Automated Turing Tests (ATTs) continue to be an effective, easy-to-deploy approach to identify automated malicious login attempts with reasonable cost of inconvenience to users. These proposed protocols, however, use more messages and rounds than those protocols that are not resistant to guessing attacks. In this paper, we propose a new Password Guessing Resistant Protocol (PGRP), derived upon revisiting prior proposals designed to restrict such attacks. While PGRP limits the total number of login attempts from unknown remote hosts to as low as a single attempt per username, legitimate users in most cases (e.g., when attempts are made from known, frequently-used machines) can make several failed login attempts before being challenged with an ATT. By adding Public Key Encryption algorithms (e.g., RSA) we can achieve good performance. We analyze the performance of PGRP with two real-world data sets and find it more promising than existing proposals.

Keywords
Public Key Encryption, Brute Force, Remote Login, Automated Turing Tests, RSA

I. Introduction
Identifying users is an indispensable element of computer security and, because auxiliary devices such as smart-card are not likely to be ubiquitous in the foreseeable future, users have to be authenticated through their passwords. Brute force and dictionary attacks on password-only remote login services are now widespread and ever increasing. Enabling convenient login for legitimate users while preventing such attacks is a difficult problem.

A. Brute-Force Attack
In cryptography, a brute-force attack, or exhaustive key search, is a cryptanalytic attack that can, in theory, be used against any encrypted data (except for data encrypted in an information-theoretically secure manner). Such an attack might be utilized when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. It consists of systematically checking all possible keys until the correct key is found. In the worst case, this would involve traversing the entire search space. The key length used in the cipher determines the practical feasibility of performing a brute-force attack, with longer keys exponentially more difficult to crack than shorter ones. A cipher with a key length of N bits can be broken in a worst-case time proportional to 2N and an average time of half that. Brute-force attacks can be made less effective by obfuscating the data to be encoded, something that makes it more difficult for an attacker to recognize when he/she has cracked the code. One of the measures of the strength of an encryption system is how long it would theoretically take an attacker to mount a successful brute-force attack against it.

Brute-force attacks are an application of brute-force search, the general problem-solving technique of enumerating all candidates and checking each one.

Fig. 1: Modern GPUs are Well-Suited to the Repetitive Tasks Associated With Hardware-Based Password Cracking

B. Section Headings
In case of an offline attack where the attacker has access to the encrypted material, he can try key combinations at his leisure without the risk of discovery or interference. However database and directory administrators can take countermeasures against online attacks, for example by limiting the number of attempts that a password can be tried, by introducing time delays between successive attempts, increasing the answer’s complexity (e.g. requiring a CAPTCHA answer or verification code sent via cell phone), and/or locking accounts out after unsuccessful logon attempts. Website administrators may prevent a particular IP address from trying more than a predetermined number of password attempts against any account on the site.

1. Dictionary Attack
In cryptanalysis and computer security, a dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by trying likely possibilities, such as words in a dictionary.

C. Technique
A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values). In contrast with a brute force attack, where a large proportion key space is searched systematically, a dictionary attack tries only those possibilities which are most likely to succeed, typically derived from a list of words for example a dictionary (hence the phrase dictionary attack). Generally,
D. Password Cracking
In cryptanalysis and computer security and password cracking is the process of recovering passwords from data that has been stored in or transmitted by a computer system. A common approach is to repeatedly try guesses for the password. Another common approach is to say that you have “forgotten” the password and then change it.

The purpose of password cracking might be to help a user recover a forgotten password (though installing an entirely new password is less of a security risk, but involves system administration privileges), to gain unauthorized access to a system, or as a preventive measure by system administrators to check for easily crackable passwords. On a file-by-file basis, password cracking is utilized to gain access to digital evidence for which a judge has allowed access but the particular file’s access is restricted.

E. Captcha
It is an acronym based on the word “capture” and standing for “Completely Automated Public Turing test to tell Computers and Humans Apart”. Carnegie Mellon University attempted to trademark the term, but the trademark application was abandoned on 21 April 2008.

A CAPTCHA is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a human being. The process usually involves a computer asking a user to complete a simple test which the computer is able to grade. These tests are designed to be easy for a computer to generate but difficult for a computer to solve. If a correct solution is received, it can be presumed to have been entered by a human. A common type of CAPTCHA requires the user to type letters and/or digits from a distorted image that appears on the screen. Such tests are commonly used to prevent unwanted internet bots from accessing websites.

Some CAPTCHAs try to utilize the ability of the people to see three-dimensional objects. On this image, symbols are drawn with lines of different thickness to make an effect of extrusion.

A modern CAPTCHA, rather than attempting to create a distorted background and high levels of warping on the text, might focus on making segmentation difficult by adding an angled line.

Another way to make segmentation difficult is to crowd symbols together, as in Yahoo’s current CAPTCHA format.

F. Accessibility
Because CAPTCHAs rely on visual perception, users unable to view a CAPTCHA due to a disability will be unable to perform the task protected by a CAPTCHA. The groups who commonly struggle with visual CAPTCHAs are:

- People who are blind, colour blind or have other issues with vision
- Individuals with dyslexia
- People of advanced age
- People with intellectual or developmental disabilities

Sites implementing CAPTCHAs may provide an audio version of the CAPTCHA in addition to the visual method. The official CAPTCHA site recommends providing an audio CAPTCHA for accessibility reasons, but it is still not usable for deaf, blind people or for users of some text-based web browsers.

Due to the sound distortion present in audio CAPTCHAs and visual distortion present in visual CAPTCHAs, offering one as an alternative to the other does not help people with impairments in both areas. While deaf blind is a small group, having some degree of impairment in both areas is actually common, and very common amongst older people.

III. RSA
RSA is an algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard
Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was classified until 1997. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message. Whether breaking RSA encryption is as hard as factoring is an open question known as the RSA problem.

Operation
The RSA algorithm involves three steps: key generation, encryption and decryption.

1. Key Generation
RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

Choose Two Distinct Prime Numbers p and q.
For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

1. Compute n = pq.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
2. Compute φ(n) = (p – 1)(q – 1), where φ is Euler’s totient function.
3. Choose an integer e such that 1 < e < φ(n) and greatest common divisor gcd(e, φ(n)) = 1; i.e., e and φ(n) are coprime.
   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.
4. Determine d as d ≡ e−1 (mod φ(n)), i.e., d is the multiplicative inverse of e (modulo φ(n)).
   - This is more clearly stated as solve for d given de ≡ 1 (mod φ(n))
   - This is often computed using the extended Euclidean algorithm.
   - d is kept as the private key exponent.

   By construction, d e ≡ 1 (mod φ(n)). The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.
   - An alternative, used by PKCS#1, is to choose d matching de ≡ 1 (mod λ) with λ = lcm(p – 1, q – 1), where lcm is the least common multiple. Using λ instead of φ(n) allows more choices for d. λ can also be defined using the Carmichael function, λ(n).
   - The ANSI X9.31 standard prescribes, IEEE 1363 describes, and PKCS#1 allows, that p and q match additional requirements: being strong primes, and being different enough that Fermat factorization fails.

Encryption
Alice transmits her public key (n, e) to Bob and keeps the private key secret. He first turns M into an integer m, such that 0 ≤ m < n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to c ≡ m^e (mod n).
This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Decryption
Alice can recover m from c by using her private key exponent d via computing m ≡ c^d (mod n).
Given m, she can recover the original message M by reversing the padding scheme.
By using RSA in Password Resistant Protocol give better security and protect the system in a consistent way.
In this section, we present the PGRP protocol, including the goals and design choices.

A. Goals, Operational Assumptions and Overview

1. Protocol Goals
Our objectives for PGRP include the following:

- The login protocol should make brute force and dictionary attacks ineffective even for adversaries with access to large botnets (i.e., capable of launching the attack from many remote hosts).
- The protocol should not have any significant impact on usability (user convenience). For example: for legitimate users, any additional steps besides entering login credentials should be minimal. Increasing the security of the protocol must have minimal effect in decreasing the login usability.
- The protocol should be easy to deploy and scalable, requiring minimum computational resources in terms of memory, processing time, and disk space.

2. Overview
The general idea behind PGRP (see fig. 1) is that except for the following two cases, all remote hosts must correctly answer an ATT challenge prior to being informed whether access is granted or the login attempt is unsuccessful:

- when the number of failed login attempts for a given username is very small; and
- when the remote host has successfully logged in using the same username in the past

In contrast to previous protocols, PGRP uses either IP addresses, cookies, or both to identify machines from which users have been successfully authenticated. The decision to require an ATT challenge upon receiving incorrect credentials is based on the received cookie (if any) and/or the remote host’s IP address.

B. Data Structure and Function Description

1. Data Structures
PGRP maintains three data structures:

- W. A list of {source IP address, username} pairs such that for each pair, a successful login from the source IP address has been initiated for the username previously.
- FT. Each entry in this table represents the number of failed logins per user. A maximum of k2 failed login attempts are recorded. Accessing a nonexistent index returns 0.
- FS. Each entry in this table represents the number of failed login attempts indexed by (srcIP, username) for hosts in W or hosts with valid cookies.

```plaintext
begin
    1. ReadCredential(un, pw, cookie) // login prompt to enter username/password pair
    2. if LoginCorrect(un, pw) then // username/password pair is correct
        3. if (!ValidCookie(un, h1, h2)) \ (|firstIP| \in W) \ (FS[,un] < h1) \ (FT[un] < h2) then
            4. FS[|firstIP|, un] = 0
            5. Add |firstIP| to W
            6. GrantAccess(un, cookie)
        else if (ATTChallenge() = Pass) then
            7. Message(The answer to the ATT challenge is incorrect)
        else if (!ValidUsername(un) \ (FT[un] < k2)) then
            8. FS[|firstIP|, un] = FS[|firstIP|, un] + 1
            9. Message(The username or password is incorrect)
        else if (ATTChallenge() = Fail) then
            10. Message(The username or password is incorrect)
        else if (ATTChallenge() = Pass) then
            11. Message(The answer to the ATT challenge is incorrect)
        else
            12. Message(The answer to the ATT challenge is incorrect)
end
```

Fig. 1: PGRP: Password Guessing Resistant Protocol
Here, srcIP is the IP address for a host in W or a host with a valid cookie, and UN is a valid username from srcIP. A maximum of k1 failed login attempts are recorded; crossing this threshold may mandate passing an ATT (e.g., depending on FT[un]). An entry is set to 0 after a successful login attempt. Accessing a nonexistent index returns 0.

2. Functions
PGRP uses the following functions (IN denotes input and OUT denotes output):
- ReadCredential (OUT: un,pw,cookie). Shows a login prompt to the user and returns the entered username and password, and the cookie received from the user’s browser (if any).
- LoginCorrect (IN: un,pw; OUT: true/false). If the provided username-password pair is valid, the function returns true; otherwise, it returns false.
- GrantAccess (IN: un,cookie). The function sends
  the cookie to the user’s browser and then enables access to the specified user account.
- ATTChallenge (OUT: Pass/Fail). Challenges the user with an ATT and returns “Pass” if the answer is correct; otherwise, it returns “Fail.”
- Valid Username (IN: un; OUT: true/false). If the provided username exists in the login system,
  the function returns true; otherwise, it returns false.
- Valid (IN: cookie,un,k1,state; OUT: cookie, true/false).
  First, the function checks the validity of the cookie (if any) where it is considered invalid in the following cases:
  - the login username does not match the cookie username;
  - the cookie is expired; or
  - the cookie counter is equal to or greater than k1.
  The function returns true only when a valid cookie is received. If state = true (i.e., the entered user credentials are correct, as in line 4 of Fig. 1), a new cookie is created (if cookies are supported in the login system) including the following information: username, expiry date, and a counter of the number of failed login attempts (since the last successful login; initialized to 0). Notice that if state = true, the bfunction does not send the created cookie to the user’s browser. Rather, the cookie is sent later by the GrantAccess() function. If state = false (i.e., the entered user credentials are incorrect, as in line 16 of Fig. 1) and a valid cookie is received, the cookie counter is incremented by one and the cookie is sent back to the user’s browser. No action is performed for all the other cases.

C. Cookies versus Source IP Addresses
Similar to the previous protocols, PGRP keeps track of user machines from which successful logins have been initiated previously. Browser cookies seem a good choice for this purpose if the login server offers a web-based interface. Typically, if no cookie is sent by the user browser to the login server, the server sends a cookie to the browser after a successful login to identify the user on the next login attempt. However, if the user uses multiple browsers or more than one OS on the same machine, the login server will be unable to identify the user in all cases. Cookies may also be deleted by users, or automatically as enabled by the private browsing mode of most modern browsers. Moreover, cookie theft (e.g., through session hijacking) might enable an adversary to impersonate a user who has successfully authenticated in the past. In addition, using cookies requires a browser interface (which, e.g., is not applicable to SSH).

Alternatively, a user machine can be identified by the source IP address. Relying on source IP addresses to trace users may result in inaccurate identification for various reasons, including: 1) the same machine might be assigned different IP addresses over time (e.g., through the network DHCP server and dial-up Internet); and 2) a group of machines might be represented by a smaller number or even a single Internet-addressable IP address if a NAT mechanism is in place. However, most NATs serve few hosts and DHCPs usually rotate IP addresses on the order of several days. Drawbacks of identifying a user by means of either a browser cookie or a source IP address include:
- failing to identify a machine from which the user has authenticated successfully in the past; and
- wrongfully identifying a machine the user has not authenticated before.

D. Decision Function for Requesting ATTs
Below we discuss issues related to ATT challenges as provided by the login server in Fig. 1. The decision to challenge the user with an ATT depends on two factors:
- whether the user has authenticated successfully from the same machine previously; and
- The total number of failed login attempts for a specific user account. For definitions of W, FT, and FS.

1. Username-Password Pair is Valid
As in the condition in line 4, upon entering a correct username-password pair, the user will not be asked to answer an ATT challenge in the following cases:
- A valid cookie is received from the user machine (i.e., the function Valid returns true) and the number of failed login attempts from the user machine’s IP address for that username, FS(srcIP; un), is less than k1 over a time period determined by t3;
- The user machine’s IP address is in the whitelisted W and the number of failed login attempts from this IP address for that username, FS(srcIP; un), is less than k1 over a time period determined by t3;
- The number of failed login attempts from any machine for that username, FT(un), is below a threshold k2 over a time period determined by t2. The last case enables a user who tries to login from a new machine/IP address for the first time before k2 is reached to proceed without an ATT. However, if the number of failed login attempts for the username exceeds the threshold k2 (default 3), this might indicate a guessing attack and hence the user must pass an ATT challenge.

2. Username-Password Pair is Invalid
Upon entering an incorrect username-password pair, the user will not be asked to answer an ATT challenge in the following cases:
- A valid cookie is received from the user machine (i.e., the function Valid returns true) and the number of failed login attempts from the user machine’s IP address for that username, FS(srcIP; un), is less than k1 (line 16) over a time period determined by t3;
- The user machine’s IP address is in the white list W and the number of failed login attempts from this IP address for that username, FS(srcIP; un), is less than k1 (line 16) over a time period determined by t3;
The username is valid and the number of failed login attempts (from any machine) for that username, FT(un), is below a threshold k2 (line 19) over a time period determined by t2. A failed login attempt from a user with a valid cookie or in the white list W will not increase the total number of failed login attempts in the FT table since it is expected that legitimate users may potentially forget or mistype their password (line 16-18). Nevertheless, if the user machine is identified by a cookie, a corresponding counter of the failed login attempts in the cookie will be updated. In addition, the FS entry indexed by the {source IP address, username} pair will also be incremented (line 17). Once the cookie counter or the corresponding FS entry hits or exceeds the threshold k1 (default value 30), the user must correctly answer an ATT challenge.

3. Output Messages
PGRP shows different messages in case of incorrect {username, password} pair (lines 21 and 24) and incorrect answer to the given ATT challenge (lines 14 and 26). While showing a human that the entered {username, password} pair is incorrect, an automated program unwilling to answer the ATT challenge cannot confirm whether it is the pair or the ATT that was incorrect. However, while this is more convenient for legitimate users, it gives more information to the attacker about the answered ATTs. PGRP can be modified to display only one message in lines 14, 21, 24, and 26 (e.g., “login fails” as in the PS and VS protocols) to prevent such information leakage.

4. Why Not to Black-List Offending IP Addresses
We choose not to create a blacklist for IP addresses making many failed login attempts for the following reasons: 1) this list may consume considerable memory; 2) legitimate users from blacklisted IP addresses could be blocked (e.g., using compromised machines); and 3) hosts using dynamic IP addresses seem more attractive targets (compared to hosts with static IP addresses) for adversaries to launch their attacks from (e.g., spammers). If the cookie mechanism is not available for the login server, PGRP can operate by using only source IP addresses to keep track of user machines.

E. Usability Comments on ATT Challenges
Our main security goal is to restrict an attacker who is in control of a large botnet from launching online single account or multi account password dictionary attacks. In terms of usability, we want to reduce the number of ATTs sent to legitimate users as much as possible. A user receives ATTs when the total number of failed attempts exceeds threshold k2, and the login attempt is initiated from 1) an unknown machine (i.e., no valid cookies or white-listed IP addresses), or 2) a known machine from which the user has already failed k1 times. This happens for both cases of correct and incorrect username-password pairs, assuming the provided username is valid. Below we discuss different login scenarios and the extra effort as required from users by PGRP. The analysis below indicates that only limited usability impact may be expected from our proposal; the same can also be inferred from our real-world data analysis, e.g., the number of ATTs sent to legitimate users (see Section 5). However, we have not yet carried out any formal user testing. For notation and parameters as used in the following, see Fig. 1. For definitions of W, FT, and FS, see “Data structures” in Section 3.2.

1. First Time Login from an Unknown Machine
If a valid username-password pair is provided from an unknown machine (i.e., one from which no successful login has occurred within a designated period), no ATTs are required if the total fail count from unknown machines is below k2 (within a time period determined by t2). This threshold may be exceeded as follows: 1) the user may provide incorrect passwords from that machine k2 times; 2) attackers may have attempted k2 failed passwords (from unknown machines); or 3) a combination of 1) and 2). Once a user successfully logs in, the machine’s IP address is added to the known list (W).

2. Subsequent Login from a Known Machine
ATTs are sent to a known machine (i.e., one from which a successful login has occurred within a designated period) only when k1 is hit or crossed (see line 4 in Fig. 1) for that machine and the user account is possibly under attack (i.e., k2 failed attempts also occurred on the account’s username from unknown machines). By setting k1 to be relatively large (e.g., k1 = 30), legitimate users may make a reasonable number of password mistakes without experiencing any ATTs.

3. Valid Password is Provided
Users may be understandably annoyed if they provide a valid password, and yet are asked to answer an ATT. When a valid password is provided by the user, no ATT challenges are sent if the attempt comes from a known machine which has not been used for more than k1 _ 1 failed login attempts within a time period determined by t3. If the user hits or crosses the threshold k1, still no ATTs are sent if the number of failed login attempts from unknown machines remains below k2. Thus, users must pass ATT challenges only when they attempt login from unknown machines and the number of failed attempts from unknown machines has hit or crossed k2 (possibly due to an ongoing attack). We believe this is an uncommon occurrence, as was apparent from our collected data.

4. Invalid Password
This may be a common occurrence for several reasons:
- if users need multiple attempts to recall the correct password;
- if users cycle-through multiple passwords due to multi-password interference; and
- typing errors including activating the caps lock key, sometimes aggravated by onscreen masking of password characters.

From each known machine, a user is allowed up to k1 attempts, before challenged with ATTs; i.e., if the user has logged in from n unknown machines (within a time period determined by t3), then in total n (k1 _ k2) attempts are allowed without ATTs. While high values of k1 (30 by default) provide convenient login for legitimate users in common use cases, we do not recommend very high values (e.g., k1 = 10,000) as that may aid guessing attacks when a cookie is stolen or a dynamic white-listed IP address is assigned to an attacker’s machine (i.e., a bot). Note that in VS, an adversary can make a certain number of failed connection attempts (the threshold b2 in Fig. 4) for all (or as many as possible) users of system, with the result that any failed login attempt from a legitimate user will face an ATT challenge. In PGRP, user convenience is unaffected by an attacker’s actions, as long as there are not more than k1 = 1 unsuccessful login attempts from known machines.
5. Invalid Username

When a user tries login with a nonexistent username (e.g., typing errors), an ATT challenge is given. Irrespective of the password or ATT answer, the login fails. This feature restricts attackers from learning valid usernames, and improves protocol performance in terms of memory usage (i.e., no entries in protocol data structures W, FT, or FS). However, from a usability point of view, this is not ideal. We expect that this type of error would be limited in practice (in part because usernames, in contrast to passwords, are echoed on a display).

F. Analysis of Results

Test results are analyzed from different perspectives below.

1. The number of successful login attempts.

The larger the ratio of successful login attempts without answering ATTs to total successful login attempts, the more convenient the login experience for the user.

2. The number of unique usernames in successful logins.

For PGRP, default parameters, the number of unique usernames in successful logins that involved answering ATTs (in the Ssh SSH data set) is three. Thus, the majority of valid users were not challenged with any ATT.

3. The number of failed login attempts with valid usernames.

Failed login attempts with valid usernames could be from either malicious or benign sources.

4. The number of unique valid usernames in failed login attempts.

In both data sets, setting $k_2 = 1$ in PGRP causes a significant decrease in the number of unique valid usernames that face ATT challenges in failed login attempts.

5. The number of failed login attempts with invalid usernames.

Any login attempt with invalid username triggers an ATT in PGRP (i.e., no failed login attempt with invalid usernames avoids an ATT). Indeed, all attempts with invalid usernames trigger ATTs in both data sets.

V. Conclusion

In this paper, we addressed the problem of online dictionary attacks and presented an authentication protocol to counter the same. In the protocol, the client is required to compute the response to the presented challenge. Online password guessing attacks on password-only systems have been observed for decades. Computing this response is deliberately designed to be a time taking operation thus ensuring that the client is not able to launch a large number of login requests in a small amount of time. The protocol is designed in a fashion such that the computation of this response does not poses any problems for a legitimate user since she may reuse the last computation, but is time consuming and costly for an adversary trying to launch thousands of login requests per second.

Table 1: Comparative Security Analysis for Multiaccount Attacks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PGRF</td>
<td>0</td>
<td>$e^N$</td>
<td>$\leq m_2^N$</td>
<td>m_1^N</td>
<td>$m_2^N$</td>
</tr>
<tr>
<td>ATT</td>
<td>0</td>
<td>$\sigma^N$</td>
<td>$\leq m_3^N$</td>
<td>$m_3^N$</td>
<td>$m_3^N$</td>
</tr>
</tbody>
</table>

Consider $k_2 = 3$, $p = 0.05$, $b_1 = 5$, and $b_2 = 5$, for concreteness.

In previous ATT-based login protocols, there exists a security usability trade-off with respect to the number of free failed login attempts (i.e., with no ATTs) versus user login convenience (e.g., less ATTs and other requirements). In contrast, PGRP is more restrictive against brute force and dictionary attacks while safely allowing a large number of free failed attempts for legitimate users. Our empirical experiments on two data sets (of one-year duration) gathered from operational network environments show that while PGRP is apparently more effective in preventing password guessing attacks (without answering ATT challenges), it also offers more convenient login experience, e.g., fewer ATT challenges for legitimate users even if no cookies are available. By encrypting the keys by using RSA the security become good. However, we reiterate that no user testing of PGRP has been conducted so far. PGRP appears suitable for organizations of both small and large number of user accounts.

References


Srinivasa Rao. His Masters Degree in M.Sc Computer Science From 2007 To 2009 At Gayatri Vidya Parishad At Visakhapatnam From Andhra University. The M.TECH. Degree in CSE from Eluru College of Engineering and Technology, Eluru in 2013. At present, He is engaged in “Protection against Significant Online Attacks on Login Password”.