

パスワード非開示によるパスワード強度測定方式

菅井 琢¹ 大東 俊博² 柿崎 淑郎³ 金岡 晃¹

概要：利用者が設定するパスワードの強度向上を促す仕組みとして、パスワードの強度測定とそのフィードバック機構が存在する。代表的なものとしては強度を画面上に表示するパスワード強度メーターがあり、これらのフィードバック機構は利用者のパスワード強度向上に効果があることがこれまでの研究で明らかになっている。パスワード強度測定の手法は多岐にわたり、リモート側にパスワードを送り測定をする手法もいくつか存在する。一方で、そういったパスワード入力途中の外部送信に関する脅威についてはこれまで十分に議論されてこなかった。そこで本研究では、まず現状の強度測定手法について代表的な Web サービスを調査し、強度測定がローカル側とリモート側で行われているケースがどれほどあるかを明らかにする。そのうえで、パスワードの入力途中の脅威についてあらためて整理をし、その対策の必要性を示す。そして対策のアプローチとして、パスワードを非開示のまま強度測定する手法の必要性について述べ、その実現アプローチを 3 つ示す。さらにその実現性を評価し、最も実現性の高い手法について試作を行い、パフォーマンス評価とユーザ実験を行った。その結果、提案手法の実現性の高さが示された。

キーワード：キーワード

1. はじめに

パスワードは長らく研究がされている分野であり、さまざまな知見がこれまでに得られてきている。研究テーマの 1 つに、利用者が意識せずに高い強度のパスワードを設定するように導く手法がある。パスワードの強度を測定し、その強度を利用者にフィードバックすることで強度向上を促すこの手法は、ユーザインタフェースとしてビジュアル効果を加えることでパスワード強度メーターとして利用されたり、あるいは文章の表示により注意喚起を促すような利用がされる。これまでの研究において、パスワード強度メーターの採用は利用者のパスワード強度向上に効果があることが明らかにされ、多くのサービスで強度測定とそのフィードバック機構が採用されることとなった。

近年では、そのフィードバック機構や強度測定手法も高度化を迎えている。多くのデータをもとに強度測定する手法や、現実のパスワード発生確率の近似値を軽量に測定することで精度の高い強度を提供する手法など、多くの研究が行われている。実際にサービスとして提供されている強度測定手法も多岐にわたり、ブラウザといったユーザの

ローカル側で強度計算されるものや、サーバに入力途中のパスワードを送信してその強度をリモートで計算するものもある。

我々は、リモートでパスワードの強度計算をするサービスに着目した。パスワードの安全性については、設定されたパスワードの送信や保管、ログイン試行時のパスワード認証の方法など、さまざまな面で安全性の担保が行われてきているが、強度計算を行うような、パスワード入力時の外部送信についてはこれまでその脅威が十分には考えられてこなかった。最近ではブラウザの拡張機能を悪用することで入力データを意図せず外部送信させられてしまう被害が起きるなど、パスワード入力途中時の外部送信についても十分な対策が必要である。

そこで本研究では、まず現状の強度測定手法について代表的な Web サービスを調査し、強度測定がローカル側とリモート側で行われているケースがどれほどあるかを明らかにする。そのうえで、パスワードの入力途中の脅威についてあらためて整理をし、その対策の必要性を示す。

そして対策のアプローチとして、パスワードを非開示のまま強度測定する手法の必要性について述べ、その実現アプローチを 3 つ示す。それぞれについて現実的な利用について調査と考察を行う、そのうち最も現実的な利用可能性が高いと考えられるものについてさらに試作を行い、実際のパスワードデータ等を用いて評価を行う。評価はパ

¹ 東邦大学
Toho University

² 東海大学
Tokai University

³ 東京電機大学
Tokyo Denki University

パフォーマンス評価とユーザ実験の2つの視点で行い、それぞれに考察を行う。

実験の結果、パフォーマンス評価は設定するパラメータ等により大きく変わるものの、ユーザ実験の結果は提案手法が低いユーザビリティでないことを示すものとなり、提案手法が十分に実用的であることが示された。

2. 関連研究

2.1 パスワードの脅威に関する研究

パスワードの脅威については多くの研究がされてきた。2009年にRockYou.comの情報漏えいにより3000万を超えるアカウントのパスワードが平文で閲覧可能になり、その後RockYou.comのデータを用いた大規模分析によりパスワードの研究が高度化を迎えた。Weirらはパスワードの強度や推測攻撃へと応用し、その精度を高めた[1]。またパスワードから他の手法に移行を考えた時の難しさを網羅的に議論し、配置容易性についてのパスワードの利便性の高さを指摘する調査もあった[2]。

その後、高度化された推測攻撃に対するべく、さまざまなアプローチがさらに発展を遂げた。パスワード設定時の構成ポリシーの効果についてのKellyらの研究は、その効果測定に推測攻撃を用いた[8]。

同じく、パスワード強度メーターについても複数の視点で研究がされてきた。メーターの外観の違いによる効果の差がUrらにより調査され、インターフェースの重要性を指摘した[9]。また強度の測定手法についてもエントロピーをもとにした近似値計算[6]や、軽量に実行可能な手法[11]をはじめとして多くの研究がされてきた[5], [7], [10]

パスワード研究についての概観はFlorncioらの調査が有用である[4]。

2017年になり、Ruotiらは次世代のパスワードを考えるべく、あらためてその脅威を整理して議論を行った[12]。そこでは脅威の存在と攻撃の可能性を整理して議論がされていたが、これらはパスワード強度メーターについても適用可能な部分があった。Ruotiらが整理した情報をもとにしたパスワード強度メーターの脅威については3章で議論をおこなう。

2.2 秘密パターンマッチングの関連研究

本論文の主目的は、リモートでのパスワード強度測定において測定者に開示することなく強度を測定する手法を実現することにある。その目的に対して直接の適用ではないが、関連する研究が存在する。

パスワード強度メーターでは、エンドユーザが入力したパスワードの文字列をあるパターンと比較することで強度を測定する手法がある。文字列のパターンマッチングをパターンを秘匿にしたまま行うことができる手法により、非開示によるパスワード強度メーターの構成は考えられる。

文字列の秘匿マッチングとしては、Hazayらが提案した手法がある[13]。Hazayらの手法では暗号化されたパターンのマッチングを内積を用いることで実現している。

また特殊なマッチングな例としては、DNAのパターンを秘匿したまま行う手法も知られている[14]。この手法ではDNAの構成が4種類の塩基から成立していることと、それらの連続性がランダムではなく一定の特徴を持つことを応用している。

3. リモートでのパスワード強度測定における脅威

3.1 パスワード認証全体に関する脅威

Ruotiらは、次世代のパスワード認証の在り方を議論するにあたり、パスワードについての脅威を整理した[12]。そこでは、パスワードに関わる行為をまず分類し、それらの行為に関わるステークホルダーの洗い出しと、それら行為に対する責任の所在と攻撃を受ける可能性とその手法について整理を行った。パスワードに関わる行為としてはパスワード生成や記憶、管理、入力、送信、認証、保管などがある。またステークホルダーとしては、ユーザー、OS、アプリケーション（Webサイトを含む）、通信路、パスワード検証サービスが挙げられている。

通信路の脅威については、行為として送信（Transmit）が挙げられており、その他の行為の脅威と責任の所在については議論がされていない。パスワード検証サービスについては、行為として受信、検証、管理と保管が挙げられている。

3.2 パスワード強度測定における脅威

Ruotiらの分類では、パスワードの生成についてはユーザーとアプリケーション側で行為が行われ、その責任はその2者に存在するとされているが、パスワードの強度測定としてのメーター機能がサービスに存在する場合、異なる脅威と責任の所在が発生する。

パスワードの強度測定は、JavaScriptなどを用いてブラウザや端末のアプリケーション側で測定を行うローカル測定と、ブラックリストなどを併用してサーバ側で測定を行うリモート測定に大別される。

リモート測定においては、ユーザが入力しているパスワードがローカル側で秘匿化の処理を受けずにそのまま測定を行うサーバ側に送られる。通信路はTLSにより暗号化などがされていることが多いため第3者からの盗聴には耐性があると考えられるが、サーバ側にはユーザの入力が平文のまま渡されることになる。最終的に決定されたパスワードは安全に送られて安全に保管される可能性があるものの、一方で強度測定時にパスワードが開示されてしまう脅威は存在しており、Ruotiらの議論はパスワード認証単体での議論としては成立しているが、パスワードの強度測

定を使ったパスワードメーターや構成ポリシーの適用などのフィードバック機構についてはさらなる検討が必要であろう。

実際に、ブラウザの拡張機能を悪用した認証情報やクレジットカード情報の詐取事件が起こっており、本来ならローカルで強度測定されるはずのデータが知らずに第三者のリモートサーバに送られてしまい、利用者側はそれに気づかないことや悪意の有無についての判断が付かない可能性が考えられる。

3.3 リモートでのパスワード強度測定状況

脅威がどれほど存在するかを明確にするため、実施にパスワード強度の測定がどのように行われているかを調査した。調査は2016年5月8日時点でのAlexa社が提供する情報の上位100サイトに対して、6月から7月にかけてWebブラウザを通じたアクセスで行った。

上位100サイトの中には、同一の企業によるサービスが各国や地域で提供されているものがそれぞれ入っているものも存在する。そういったサービスの中には、認証や登録の情報は一元化されているものもある。Google社のサービスは100サイトのうち18を占めていたが、ユーザ登録は1つのサイトに統一されていた。また100サイトのうち、いくつかのサイトについてはユーザ登録の前処理として電話番号によるSMS確認が必要なものがあり、それらについては調査が行えなかったために対象外とした

調査の結果、13のサービスにおいてパスワード強度測定が行われていることが確認された。13のサービスのうち、ローカルで強度測定を行うサービスが9、リモートで強度測定を行うサービスが4あった。いずれも世界的に広くユーザが存在するサービスであり、リモートでの強度測定が一般的であることがわかり、リモートでの強度測定の脅威を分析する必要性が示されたと言える。

4. リモートパスワード強度測定への対策アプローチ

4.1 アプローチの基礎検討

パスワードの強度測定をどのサービスでもすべてローカル側で行い、サーバ側にはパスワードの平文情報を送らないということが可能ならば、対策となりうるが、現実的ではないと考えられる。

利用者が設定するパスワードを制限することは、サービス提供者が考える認証レベルなどのポリシーが反映されているものであり、そのポリシーの中にはブラックリストを利用した制限などを加えることも十分に考えられる。ブラックリストの管理やデータサイズなどを考えると、リアルタイムでの更新や大量ブラックリストデータのダウンロードなどが発生するため、ローカル側で完全に実施することの難しさがあり、リモート側で実施する動機がサービ

ス提供側に存在する。

このことから、リモートでのパスワード強度測定の排除は難しいと考えられる。よって、アプローチの基礎検討としてはリモートでのパスワード強度測定を安全に行うことを方針とする。

その方針をもとに、本研究では3つのアプローチを検討した。

- 秘匿パターンマッチング手法の利用（1）文字列マッチング利用
- 秘匿パターンマッチング手法の利用（2）DNA マッチングの応用
- ハッシュ関数を用いた非開示マッチング

それぞれについて、アプローチの概説とその実現性について議論する。その概要と、可能性について議論します。

4.2 秘密パターンマッチング手法の利用（1）：文字列マッチング利用

2章で紹介したHazayらの秘匿文字列マッチング手法は、エンドユーザ側がパスワードの候補を暗号化してサーバ側に送信し、サーバ側でマッチングを行うことでスコア計算に応用できる可能性がある。サーバ側ではブラックリストとなる単語や文字の構成、同一文字の連続回数、長さなどをリモートの強度測定に利用される各種指標をHazayらの手法で実現可能である。

Hazayらの手法では、乱数とあわせてElGamal暗号とDH鍵交換を組み合わせることで文字毎の暗号化を行い、暗号化された文字列同士を内積計算をすることでマッチングの判定をする。Hazayらの手法ではクライアント側の演算とサーバ側の演算の両方が必要となるため、必要となる演算をクライアント側とサーバ側を想定した環境で用意し、そのパフォーマンスを評価した。

クライアント側環境としてはブラウザ上でJavaScriptによる実行、サーバ環境としてJavaによる実行を行った。クライアント側、サーバ側で必要となる演算の種類と1回の演算に必要な処理時間、また1回のマッチングに必要なそれぞれの演算回数を表1、2に示す。ここで、パスワードとして送られる情報は160ビットとし、サーバ側でマッチング用に用意された情報は40ビットであるとした。またいずれもOSとしてWindows 10、CPUはIntel Core i3-3320 (3.3GHz)、RAMは6GB、利用ブラウザはGoogle Chrome (55.0.2883.87m)の環境で実施した。計測はそれぞれ100回繰り返し、その平均処理時間を求めた。

これらの処理時間に加え、Hazayらの手法では2者間での通信が発生する。今回の環境では、通信が163回発生する。クライアントがPOSTアクセスによりサーバにアクセスし、サーバがレスポンスを返すまでの時間を測定したところ、平均で1.06msecとなった。このことから通信時間としては少なくとも172.78msecが必要になると考えられ

表 1 Hazay らの手法におけるクライアント側の処理回数と処理時間

演算種類	処理時間 (ms)	回数	処理合計 (ms)
乗算	0.01111	5485	60.9386
べき乗	0.01031	5082	52.3954
ビット列変換	1.20988	200	241.976
Π_{keygen}	0.02633	1	0.02633
乱数生成	0.02481	160	3.9696
Elgamal 暗号	0.48908	160	78.2528
逆元	0.26403	121	31.94763
Π_{isbit}	0.01753	40	0.701182
合計			470.2073129

表 2 Hazay らの手法におけるサーバ側の処理回数と処理時間

演算種類	処理時間 (ms)	回数	処理合計 (ms)
乗算	1.8344E-06	5485	0.010061684
べき乗	7.8846E-05	5082	0.400694864
ビット列への変換	0.023939	200	4.7878
Π_{keygen}	0.000402	1	0.00040222
乱数生成	0.0006423	40	0.025694552
Elgamal 暗号	4.51405	40	180.562
逆元	0.002798	121	0.338558
Π_{isbit}	0.3712632	160	59.4021201
合計			245.5273

る。サーバ側の処理時間の合計とクライアント側の処理時間の合計、そしてこの通信時間を合わせると、1 回のマッチングに 888msec がかかることとなる。

これは 1 回のマッチングについて必要な時間であり、パスワード強度測定として用いる場合は、ブラックリストとのマッチングや特定文字列（同一文字の連続）のマッチングなど多くのマッチングが必要となる。また強度測定は一般的に利用者がキーボードより入力するたびに実施されることも考えると、Hazay らの手法を直接適用することは現実的ではないと判断できる。

4.3 秘匿パターンマッチング手法の利用 (2) : DNA マッチングの応用

DNA は 4 種類の塩基から構成される。また塩基の構成は順序まったくのランダムではなく、特性がある。そこに注目して Florncio らは効率のよい秘匿マッチングを提案した [3]。

パスワードも、文字を L、数字を D、記号を S としたときのパスワードの構成で特性を測ることがある。このパスワードの構成が DNA の塩基と同様に何らかの特性があり、さらにその構成による強度の分布が構成ごとに有意に差があると、Florncio らの手法を応用した効率のよい秘匿マッチングが実現できる可能性がある。

そこで、オープンソースとしても提供されている zxcvbn の手法と RockYou のパスワードデータセットを用いて、それぞれのパスワードのスコアを計算し、さらにそれぞれのパスワードをパスワード構成ごとに分類し、それぞれの構

成における強度スコアの分布を調査した。zxcvbn ではスコアだけではなく、アタックタイムの推測値を算出することもできるため、スコアの平均と分散、アタックタイムの平均と分散をそれぞれの構成ごとに計算して比較をした。結果の分量が多くなるため、省略するが、それぞれの分布に統計的な有意性が現れることなく、パスワードの構成自体は強い特性を持つものではないことが示されたため、Florncio らの手法を応用することは難しいと判断した。

4.4 ハッシュ関数を用いた非開示マッチング

暗号学的ハッシュ関数を利用することで、パスワード情報を非開示のままマッチングする手法を提案する。

基本のアイデアとしては次の通りとなる。クライアント側は、強度測定を行いたいパスワードをハッシュ関数によりハッシュ化し、サーバ側へ送る。サーバ側は、受け取ったハッシュ値と一致するデータを DB より探し、一致するデータが見つかった場合はそのデータに付随するスコアを返す。これによりサーバ側にパスワードを非開示したままスコアの取得が可能になる。

暗号学的ハッシュ関数は軽量に計算が可能であるため、クライアント側での依頼データの作成やサーバ側でのマッチング速度も現実的な時間で行えることが考えられる。次章ではこの手法を詳細に検討する。

5. ハッシュ関数を用いた非開示マッチングの提案

5.1 基本的な機能と前提

ハッシュ関数によりハッシュ化されたパスワードをサーバ側へ渡し、サーバ側はハッシュ値をもとにデータ探索を行い、一致したデータに付随するスコア情報を返すことで強度スコアの提供を行う。

前提として、パスワードのリストとそれぞれのスコア情報を持っている信頼できる第 3 者機関がいることを想定する。第 3 者機関はパスワードとそのスコアの情報を、パスワードをハッシュ化したデータベースにしてサーバ側へ送ることで、初期の設定とする。

サーバ側はパスワードのハッシュ値とそのパスワードのスコアを持つ。このとき、パスワードの種類や数はサービスの内容により選択可能であるとする。たとえば、ブラックリストとして利用することも可能であり、あるいはある特定のパスワード群にのみスコア提供をしたい場合など、多岐にわたり利用可能になる。またスコアの算出方法はこの手法とは独立して設定可能であるため、強度の測定手法は柔軟に選択可能である。

5.2 匿名化への対応

スコアの算出手法によっては、あるパスワードが特殊なスコアを持つなど、データの秘匿性が保たれないケースが

考えられる。RockYou データセットを zxcvbn でスコア算出した場合、14,344,391 件のパスワードが 0-4 のスコアに分類される。最も少ないスコアが 0 で 58,349 件となるため、この場合は十分な秘匿化ができていと考えられる。一方で、Ur らが提案した手法 [10] によるスコア算出では、0-100 までのスコア分類に対し、RockYou データセットからランダムに選んだ 240,023 件に対し、スコアを 10 刻みにした分類としても 90-99 のスコアを出すパスワードが 1 件、80-89 のスコアを出すパスワードが 2 件と偏りが生じ、秘匿化が十分ではないと判断することもできる。匿名化については、スコアの算出手法に強く依存するものではあるが、k-匿名化の手法などを応用することで秘匿性の強化は可能である。

5.3 リプレイ攻撃への対応

パスワードをハッシュ関数に通して得られるハッシュ値は、パスワードの値が同じときは同じハッシュ値が得られる。このハッシュ値をそのままサーバ側におくと、サーバ側は少なくとも以前と同じパスワードのデータが送られてきたことがわかる。また第 3 者が通信をのぞき見している場合、リプレイ攻撃が可能になる。

これらの脅威には、毎回乱数を発生させてパスワードとともにハッシュ関数に通すことで回避が可能になる。サーバ側での対応が必要になるが、乱数値を渡すことでサーバ側にも情報が漏えいする恐れがあるため、乱数は渡さないほうが望ましい。そこで、以下の手法により対応する。

- クライアント側
 - 乱数 r を用意する
 - パスワード w のハッシュ値 $H(w)$ を求める
 - 乱数 r とパスワードのハッシュ値 $H(w)$ から再度ハッシュ値 $H(H(w), r)$ を求める
 - $H(H(w), r)$ をサーバに送信する
- クライアント側
 - $H(H(w), r)$ を受信する
 - データベースからパスワード候補 $H(w')$ とそのスコア $S(w')$ を選択する。
 - 乱数候補 r' を選択する
 - パスワード候補 $H(w')$ と乱数候補 r' から $H(H(w'), r')$ を計算し、受信データと一致を確認する
 - 一致した場合 $S(w')$ を返す

乱数の値域とデータベースのサイズに依存してパフォーマンスが落ちるが、脅威とデータベースの検討によりバランスが取れると考えられる。

5.4 データベースに存在しないパスワードへの対応

信頼できる第 3 者から受け取ったデータに含まれないパスワードを受け取った場合、サーバ側は適切な強度スコア

の回答ができない。

そういった場合の回答の必要については、この手法をサービスとしてどう利用するかに依存する。たとえば、存在しないことで十分な情報が提供可能とする場合は、スコアとして最高値をつけて回答することや、別途特殊なデータを送信することも可能になるだろう。一方で、新たにデータを秘匿したまま追加する方法として Hazay らの手法などを用いてスコアを算出することも考えられる。登録に時間はかかるものの、1 日に 1 回行うバッチ処理で対応するなど、リアルタイムとしての回答とは別の処理を行うことで、サーバ側のデータベースをある程度の動的性を持たせることも可能となる。

6. 試作システムによる評価

提案手法を試作し評価を行った。試作では、ブラウザ側で乱数生成とハッシュ値計算を行い送信するプログラムを JavaScript で作成し、Web サイトコンテンツとして埋め込んだ。サーバ側では、受信したデータに対して、あらかじめ定められた乱数の値域に対して総当たりでマッチングを行う。サーバ側は Linux OS に MySQL を Database として持ち、ハッシュ化したパスワードデータとそのスコアをテーブルとして持つ。

6.1 基礎性能評価

RockYou データセットの頻出上位 100,000 件のパスワードから得たハッシュ値データとスコアを計算して MySQL のテーブルに入力し、その検索時間を測った。検索クエリは 100,000 件からランダムに選んだ 1,000 件で行った。ここでは乱数を導入せず、単純にハッシュ値のマッチングがどれほどの時間で行われるかを測る。

実行の結果、1 件あたりの実行時間が 38.918msec となった。続いて、同じ 1,000 件で再度実行をすると、1 件あたりの実行時間が 0.042msec となり大幅に改善された。これは MySQL のサーバがクエリのキャッシュをしているためである。キャッシュの性能についてはチューニングにより変更がされ、統一的な評価が難しいため、この 2 つの値を併記した。

6.2 ユーザ実験

提案手法のユーザビリティを評価するため、ユーザ実験を行った。

6.2.1 利用したサービスと期間、報酬

2018 年 8 月 18 日 (土) ~ 8 月 20 日 (月)、ランサーズにて被験者を募集し、アンケートに回答してもらうとともに、研究室で準備したサーバ上での作業をしてもらうことで実験を行った。タスク終了の目安時間は 5 分で報酬は 100 円である。目安時間の 5 分については、実際に研究室で他の学生に模擬実験を行ってもらい、計測した作業時間

から妥当であると考えた値である。報酬については、東京都の1時間あたりの最低賃金を基準とした額である。被験者は493人であった。

6.2.2 実験目的

実験目的として、新たなパスワード強度メーターの利用実験である旨を伝えた。実験にあたり、パスワードの送信がされることについての同意を得た。さらにパスワード入力に当たっては普段さまざまなサービスで実際に利用しているパスワードを入力しないよう注意喚起を行った。

6.2.3 パスワード強度メーターのデザイン

パスワード強度メーターのデザインは、Urらの研究[9]で用いられたbaselineメーターと同等なものを用意した。パスワードのスコアを8段階に分け、段階に応じてスコアを表すバーが伸びるようにし、さらに色も赤からオレンジ、緑と変わるようにした。

本来の「特徴分析のためのパスワード収集」と言う目的を伝えと、被験者がパスワード作成に影響を及ぼしてしまう恐れがあったため、なるべく自然にパスワードを作成してもらえよう、別の目的を伝え実験を行った。別の目的とは、ユーザがウェブサイトやアプリなど複数のサービスにアカウントを持つ現代において、どれだけ複数のアカウントを管理できているかを調査したいと言う、「利用者の記憶の調査」である。そのため被験者には最初、「利用者による記憶を測るために、今回10種類の仮ウェブサイトにてユーザ登録してアカウントを作ってもらおう。日程をあけてもう1度実験を行い、どれだけ前回登録したパスワードを覚えていられるか調査する。」と言う説明をし、パスワードを作成してもらった。そして最後に真の目的を伝え、同意を得た人のデータのみサーバに取得した。

6.2.4 比較対象

提案システムとの比較のために、単純にDB内のパスワードとマッチングをしてスコアを返すサービスを用意した。ブラウザのインターフェースは同じものとし、被験者はランダムに提案サービスと比較対象サービスに割り振られることとした。

6.2.5 パスワードスコアの計算

パスワードスコアの計算は、Weirらが提案したPCFG (Probabilistic Context-free Grammar) をもとにしたパスワードの出現確率計算手法により確率を求め、その確率から情報量を求めてスコアとして用いた。乱数の設定は値域を0から9までとした。

6.2.6 実験の詳細

実験の流れは以下の通りである。

- (1) ランサーズのページから今回実験のために作成したウェブサイトへアクセスする
- (2) 目的を提示し、同意を得る
- (3) 実験のために作成したウェブサイトの画面に遷移する
- (4) パスワード入力を行う

(5) アンケートに回答する

以上の流れにより、1人1個のパスワードを入力してもらう。

データは、ランサーズ上にて被験者の性別、年齢、利用端末(パソコン、スマートフォン、その他)、職業、最終学歴を取得し、研究室のサーバに送信元IPアドレス、利用ブラウザ情報、送信日時、入力パスワード、アンケート回答を取得した。そしてランサーズ上で取得した情報とサーバで取得した情報のデータ照合を行いデータのマージをした。

6.2.7 アンケート内容

アンケート内容はSUS (System Usability Scale) の質問項目に則り設定した。SUSの質問項目を設定することにより、ユーザがそのシステムに対してどうユーザビリティを感じているかがスコア化される。

6.2.8 生命倫理審査委員会の承認

本実験は学内の生命倫理審査委員会に承認を得て行った。

6.3 ユーザ実験結果

実験の結果、アンケートの回答不備のデータを除いた479件のデータが取得できた。そのうち、提案システムを利用した被験者は253人であり、比較対象サービスを利用した被験者は226人であった。またそのSUSスコアは、提案システム利用者では64.16であり、比較対象サービスの利用者では64.48とほぼ同様の結果となった。

7. 考察

ユーザ実験の結果は良好なものであった。比較対象サービスでは、1回の検索につき1個のハッシュ値データを10万件のデータから探すだけであり、その探索にDBの探索関数を用いることができるために高速に検索が可能となっていた。一方で、提案手法では、乱数の値域が10であり、それぞれの乱数候補に対して10万件のハッシュ値を求めなおしてマッチングを取ることから、1件あたりのマッチングは高速であるものの全体としては大きなパフォーマンスの差がでる実装となった。しかしユーザ実験の結果を見ると、被験者はそのユーザビリティの低下を感じている結果とはならず、良好なユーザビリティを提供できていることが示された。

一方で、この結果はデータ数が10万件であることと乱数の値域が10であることに起因することが十分に考えられる。乱数の値域が100に増えたときには、単純に処理時間が10倍になる。その場合、被験者はストレスを抱えることになり、結果も変わってくることとなるだろう。同様に、DBが持つパスワードの件数についても同じことが言える。より条件を細分化することで、どのレベルで被験者がユーザビリティの低下を感じるかを調査し、その閾値を明らかにすることが今後の課題となるだろう。

8. おわりに

本論文では、パスワードを非開示のままリモート側でのパスワード強度の測定を実現する手法を提案し、試作による評価を行った。手法の提案は複数行ったが、そのうちに現実的な応用可能なハッシュ関数を用いた手法について実システムを試作し、パフォーマンスの評価とユーザ実験を行った。ユーザ実験の結果からは、提案手法がユーザビリティを下げる結果は出ず、良好なものとなった。一方で、試作環境でのパラメータにはより考察が必要であることも明確になった。今後は、パラメータの細分化を行い、被験者の許容範囲がどの程度にあるかを明らかにすることで、本手法の現実性をより明らかにすることが必要となる。

参考文献

- [1] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing metrics for password creation policies by attacking large sets of revealed passwords. In Proceedings of the 17th ACM conference on Computer and communications security (CCS '10), 2010
- [2] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, SP '12, pages 553-567, Washington, DC, USA, 2012. IEEE Computer Society.
- [3] D. Florncio, C. Herley, and P. C. Van Oorschot. An administrator's guide to internet password research. In Proceedings of the 28th USENIX Conference on Large Installation System Administration, LISA'14, pages 35-52, Berkeley, CA, USA, 2014. USENIX Association.
- [4] C. Castelluccia, M. Drmuth, and D. Perito. Adaptive password-strength meters from markov models. In NDSS. The Internet Society, 2012.
- [5] X. de Carn de Carnavalet and M. Mannan. From very weak to very strong: Analyzing password-strength meters. In Network and Distributed System Security Symposium (NDSS 2014). Internet Society, 2014.
- [6] M. Dell'Amico and M. Filippone. Monte carlo strength evaluation: Fast and reliable password checking. In Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, pages 158-169, New York, NY, USA, 2015. ACM.
- [7] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley. Does my password go up to eleven?: The impact of password meters on password selection. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13, pages 2379-2388, New York, NY, USA, 2013. ACM.
- [8] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 523-537, May 2012.
- [9] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor. How does your password measure up? the effect of strength meters on password creation. In Proceedings of the 21st USENIX Conference on Security Symposium, Security'12, pages 5-5, Berkeley, CA, USA, 2012. USENIX Association.
- [10] B. Ur, S. M. Segreti, L. Bauer, N. Christin, L. F. Cranor, S. Komanduri, D. Kurilova, M. L. Mazurek, W. Melicher, and R. Shay. Measuring real-world accuracies and biases in modeling password guessability. In 24th USENIX Security Symposium (USENIX Security 15), pages 463-481, Washington, D.C., Aug. 2015. USENIX Association.
- [11] Daniel Lowe Wheeler: zxcvbn: Low-Budget Password Strength Estimation, 25th USENIX Security Symposium (USENIX Security 16), 2016
- [12] Scott Ruoti and Kent Seamons. 2017. End-to-End Passwords. In Proceedings of the 2017 New Security Paradigms Workshop (NSPW 2017), 2017
- [13] Hazay C., Toft T. (2010) Computationally Secure Pattern Matching in the Presence of Malicious Adversaries. In: Abe M. (eds) Advances in Cryptology - ASIACRYPT 2010. ASIACRYPT 2010
- [14] Juan Ramn Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. 2007. Privacy preserving error resilient dna searching through oblivious automata. In Proceedings of the 14th ACM conference on Computer and communications security (CCS '07), 2007