秘密分散法とサーバアプリを用いた安全性と利便性を両立するパスワードマネージャの提案

福光 正幸 † 長谷川 真吾 ‡ 岩崎 淳也* 酒井 正夫 ‡ 高橋 大樹 ‡

† 北海道情報大学 情報メディア学部
069-8585 北海道江別市西野幌 59 番 2
fukumitsu@do-johodai.ac.jp

‡ 東北大学 情報科学研究科
980-8576 仙台市青葉区川内 41
{hasegawa,sakai}@cite.tohoku.ac.jp, dtakahashi@isl.is.tohoku.ac.jp

* 東北大学 大学院医学系研究科　医学教育推進センター
980-8575 仙台市青葉区星陵町 2-1
iwazaki@med.tohoku.ac.jp

あらまし 複数の ID/パスワードを一括して管理するパスワードマネージャは, その安全性をマスターパスワードやセキュリティトークンで守るのが一般的である. しかし, 前者は, ユーザが強度の低いマスターパスワードを使用することで安全性が低下し, また, 後者は, トークンの携帯が必須のため利便性が低下する問題がある. そこで, 本稿では, 近年普及するパーソナルサーバを活用し, 秘密分散法により ID/パスワードのデータを, PC, スマートフォン, サーバに分散保存し, さらに, サーバアプリにより制限的なパスワード認証の利用を可能にすることで, 高い安全性を実現しつつ, 実用上の利便性を低下させない, 新しいパスワードマネージャの方式を提案する.

A Proposal of a password manager satisfying security and usability by using the secret sharing and a personal server.

Masayuki Fukumitsu† Shingo Hasegawa‡ Junya Iwazaki* Masao Sakai‡ Daiki Takahashi‡

†Faculty of Information Media, Hokkido Information University.
Nishi Nopporo 59-2 Ebetsu, Hokkaido 069-8585, JAPAN
fukumitsu@do-johodai.ac.jp

‡Graduate School of Information Sciences, Tohoku University
41 Kawauchi, Aoba-ku, Sendai, Miyagi, 980-8576, JAPAN
{hasegawa,sakai}@cite.tohoku.ac.jp, dtakahashi@isl.is.tohoku.ac.jp

* Graduate School of Medicine, Office of Medical Education, Tohoku University
2-1 Seiryo-machi, Aoba-ku, Sendai, Miyagi, 980-8575, JAPAN
iwazaki@med.tohoku.ac.jp

Abstract Password managers protect their passwords using a master password or a security token. The security of it using the former is weakened if users use weak master passwords. The usability of it using the latter is low since users always need the token to log in. In this paper, we propose a new framework of password managers which is secure and has the high usability by employing the secret sharing and the personal servers.

1 はじめに

現在, Web サービスでは, ID/パスワードによるユーザ認証が幅広く使用されているが, その使用には様々な問題が指摘されている. 例えば, ユーザが脆弱なパスワードを設定する問題（以後, パスワード脆弱性問題）が挙げられる. ユーザは，自分が覚えやすいパスワードを設定する傾向があるため, 安全性が不十分なパスワードになってしまうことが少なくない [1, 6]. また, 現代人は, 多様な Web サービスの利用を求められつつあるが, それに伴い, ユーザが複数の Web サービスにおいて同じ ID/パスワードを使い回す問題（以後, ID/パスワード使い回し問題）[3, 6] も指摘されている. ユーザが ID/パスワードの使い回しをした場合, どこか 1 つの Web サービスにおいて ID/パスワードが破られたり, または ID/パスワードを含む大規模なデータ漏洩が発生すると, 同じ ID/パスワードを使い回している全ての Web サービスのユーザアカウントが同時に危険にさらされることになる [6].

このような ID/パスワード認証における問題に対して, Web サービスを運営する一部の Web サイトでは, 使用可能なパスワードに厳格なポリシーを設けたり, 一定時間当たりのユーザ認証の試行回数を制限するなどの対策をとっている. しかし, このような対策はパスワード脆弱性問題には有効であっても, ID/パスワード使い回し問題には効果が限定的である. ID/パスワード使い回し問題への対策としては, ユーザ認証方式自体を, 画像認証, 生体認証などの ID/パスワード認証以外の多様なものに変更することが有効と考えられる. しかし, その変更には, Web サービス側での対応だけでなく, ユーザ側での認証用ハードウェアの導入や, 新しいユーザ認証方式に対するユーザーの学習コストといった負担が発生する. また, 使い勝手の良いユーザ認証方式には人気が集まるため, 多様なユーザ認証方式が幅広く採用されるというのは現実的ではない. さらに, ユーザが新しい多様なユーザ認証方式に習熟するまでに発生するセキュリティリスクも無視することができない.

1.1 パスワードマネージャ

ID/パスワード認証における問題を, ユーザ自身が解決する手段の 1 つとして，パスワードマネージャの使用が考えられる. パスワードマネージャとは，複数の ID/パスワードを一括して管理するソフトウェアであり，近年では，Web ブラウザとの連携使用を前提として，Web ブラウザのプラグイン (拡張機能) や, または Web アプリケーションとして開発されている. 商用製品・サービスとしては, 1Password [7] や, LastPass [8] などが有名である. また, 最新の Web ブラウザは簡易なパスワードマネージャ機能を標準で搭載している.

Web ブラウザと連携する一般的なパスワードマネージャの動作の概要について説明する. Web ブラウザで, パスワードマネージャに未登録の Web サービスのログイン処理が実行されると, その Web サービスの URL と ID/パスワード（以後, ログイン情報）がパスワードマネージャに登録される. なお, この時に記録されるログイン情報のデータは, 端末の盗難や不正ログイン時の情報漏洩を回避するために, マスターパスワードやセキュリティトークンを用いて暗号化される. また, Web ブラウザで, パスワードマネージャに登録済み Web サービスの URL に接続した場合, パスワードマネージャがマスターパスワードやセキュリティトークンによる本人確認を行い, それに通過することで, 登録済みのログイン情報の ID/パスワードが自動入力される.

このように, パスワードマネージャを利用することで, ユーザは, 2 回目以降のログイン処理時には, その Web サービスの ID/パスワードを覚えておく必要がない. したがって, ユーザは, Web サービス毎に異なる, 強度の高いパスワードを設定することが容易となる. すなわち, パスワードマネージャは, パスワード脆弱性問題と ID/パスワード使い回し問題への有効な対策と言える.

前述したように, パスワードマネージャに記録されたログイン情報の安全は, マスターパスワードやトークンを用いて守られる. マスターパスワードを用いる場合には, 安易なマスター

パスワードを用いると, 端末盗難時のオフライン攻撃により, 登録済みの全てのログイン情報が攻撃者に知られてしまう恐れがある. したがって, 強度の高いマスターパスワードの設定が重要となる. しかし, 全てのユーザにそれを求めることは容易ではない.

一方, トークンを用いた場合には, オフライン攻撃のリスクを低減できる. しかし, ユーザは, パスワードマネージャの利用時にトークンを所持している必要がある. また, 端末とトークンが一緒に盗難される危険を回避するために, 端末とトークンを別々に管理する必要がある. しかし, 実際には, ユーザは利便性のために, 端末とトークンを同じ場所に保管する可能性が高い.

1.2 スマートフォンを活用したMBCCO方式 [4]

前節で述べたパスワードマネージャの実用上の問題への対策として, McCarbey らは, スマートフォンをトークン兼補助端末として利用するパスワードマネージャ(以後, MBCCO 方式) を提案している [4].

MBCCO 方式では, パスワードマネージャに登録された Web サービスのログイン情報の暗号化データを, 補助端末として紐付けられたスマートフォンに保存し, パスワードマネージャを利用する PC 端末（以降, PC）側には, 登録された Web サービスのログイン情報の暗号化データを残さず, それを復号する復号鍵 (共通鍵暗号のため暗号化鍵と同一) のみを保存するのが特徴である. ユーザが, PC 側で登録済みの Web サービスにログインする場合, スマートフォン側で保存済みリストからその Web サービスを選択することで, 保存されている該当の暗号データが PC に送信され, PC でその復号処理がなされ ID/パスワードが自動入力される.

MBCCO 方式では, PC が盗まれても，PC にはログイン情報のデータが保存されていないため, 攻撃者はそれだけではログイン情報を絶対に手に入れることが出来ない. また, スマートフォンが盗まれた場合も, 復号鍵は PC 側にのみ保存されているため，ログイン情報の暗号化データを復号するためには, 共通鍵暗号を破る必要がある.

MBCCO 方式のパスワードマネージャの利用時には, スマートフォンが必須となるため，スマートフォンを物理トークンとみなすことができる. 前述したように, トークンはパスワードマネージャの利用時に必須であり, また, 一緒の盗難を避けるために普段は端末とトークンを別々に管理する必要がある. 常時携帯されるスマートフォンをトークンとして用いる MBCCO 方式は, この点でも適している.

1.3 本研究の貢献

前節で紹介した MBCCO 方式は, スマートフォン側からパスワードマネージャの機能を利用できず, またスマートフォンを紛失した場合には登録済みの全てのログイン情報の復元が不可能になる問題が, その開発者ら自身により指摘されている [4].

そこで, 本研究では, MBCCO 方式のフレームワークを拡張し, この 2 つの問題を解決する新しいパスワードマネージャの提案を行う. 提案方式では, 近年普及するパーソナルサーバ（以後, サーバ）を活用し, $(2,3)$-しきい値秘密分散法によりログイン情報のデータを, PC, スマートフォン, サーバに分散保存する. ここで, (k,n)-しきい値秘密分散法 [5] とは, 秘密情報を n 人で安全に共有する手法であり, k 人以上が結託しない限り, 元の秘密情報を一切復元できないことが情報理論的に保証されている. したがって, 提案方式の $(2,3)$-しきい値秘密分散法の場合, 3 個の分散データ（以後, シェア）の内, 任意の 2 個以上を用いることで元のログイン情報が復元できるが，シェアが 1 個の場合では復元できない. この特性を活かすことで, PC とスマートフォンの両方からパスワードマネージャの機能が利用可能となり, また, スマートフォンまたは PC の一方が失われた場合にも登録済みのログイン情報を復元できるようになる. また, PC, スマートフォン, サーバの何れか 1 つのデータが盗まれだけでは, 攻撃者が元のログイン情報を復元することは非常に困難である.

提案方式では, スマートフォン（または, PC）からパスワードマネージャ機能を利用する際は, PC（または, スマートフォン）だけでなくサーバも補助端末として利用できる. スマートフォンから利用する場合は, ユーザの手元に PC がないケースも十分想定されるため, このことは可用性を高める効果がある.

ところで, 提案方式では, サーバを補助端末として利用する際のユーザ認証に, パスワード認証を用いる. これは, マスターパスワードを撤廃した MBCCO 方式に比べて欠点とも考えられる. しかし, 提案方式では, サーバのパスワード認証が破られた場合でも, シェアの 1 個が奪われるだけであり, 攻撃者はそれだけでは元のログイン情報を復元できない. もちろん, スマートフォンまたは PC の一方が盗難されて, シェアが既に 1 個奪われている段階ならば, サーバのパスワード認証の危険性は深刻となる. それでも, 攻撃者はサーバに対するオンライン攻撃を行わなければならないため, サーバアプリケーション側で, パスワード認証の試行回数の制限や, 攻撃検知・通知, またシェアへの制限的アクセスなどの工夫の余地があり, 単純なパスワード認証以上の安全性が期待できる.

2 MBCCO 方式のプロトコル [4]

MBCCO 方式のパスワードマネージャの機能は, 初期設定, ログイン情報登録, ログイン情報復元の 3 つに大別される. 本節では, 図 1 に示す各機能のプロトコルについて順に説明する.

初期設定プロトコル　初期設定プロトコルは最初の準備作業であり, PC の Web ブラウザにプラグインを, またスマートフォンに専用のスマートフォンアプリ（以後, スマフォアプリ）をインストールする. また, 共通鍵方式の鍵 k を作成し PC に保存する.

さらに, インターネット上に PC/スマートフォン間の安全な通信路を構築するための前処理を行う. インターネット上の 2 点間で安全な通信路を構築するためには, どちらかが相手の IP アドレス情報を知り, また相互に相手の正しい公開鍵を共有する必要がある. しかし, 一般的なユーザが利用する PC とスマートフォンは, いずれも IP アドレス情報が固定ではなく, また公的機関の証明書付きの公開鍵も所持していないため, PC/スマートフォン間に安全な通信路を構築することは容易ではない. MBCCO 方式では, 安全な TLS 通信路構築 [2] に必要な情報を, QR コードで PC からスマートフォンに直接的に送信することで, なりすましやデータ改竄のリスクを回避している.

ログイン情報登録プロトコル　ログイン情報登録プロトコルは, PC の Web ブラウザ上でログイン処理が行われ, ユーザがそのログイン情報 $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ の登録を許可した場合に実行される.

図 1 の（Add-2）では, site の暗号化データ c_{site} を, PC からスマートフォンに送信する. 具体的には, PC がスマートフォンにプッシュ通知を送り, スマートフォン側でユーザがその通知をタップすることでスマフォアプリが起動し, 安全な TLS 通信路を構築して c_{site} を送信する.

図 1 の（Add-3）では, スマフォアプリは, タグ t_{site} を, c_{site} のラベルとしてユーザに入力させる. ここで, ユーザが, 第三者に理解できないように t_{site} を決めることで, スマートフォンから $(t_{\mathsf{site}}, c_{\mathsf{site}})$ のリストが漏洩した場合でも, ユーザが登録済み Web サービスを秘匿する効果が期待できる.

ログイン情報復元プロトコル　ユーザが, Web ブラウザ上にログイン情報を登録済みの Web サービスのログイン画面を表示し, さらにスマートフォンでスマフォアプリを起動して, 登録済み Web サービスのタグリストからログイン情報を復元させたいタグ t_{site} をタップすることで, スマフォアプリは, スマートフォンから PC に, t_{site} に対応する c_{site} を, 安全な TLS 通信路を利用して送信する. ブラウザプラグインは, 鍵 k で c_{site} から $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を復号し, Web ブラウザの表示ページの URL が URL_{site} と一致することを確認し, 一致すれば $(ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を入力する.

初期設定プロトコル

(Ini-1) PC にはブラウザプラグインを, またスマートフォンにはスマフォアプリをインストールする.

(Ini-2) プラウザプラグインは, 共通鍵暗号方式の鍵 k を作成し, PC に保存する.

(Ini-3) ブラウザプラグインは, TLS 通信認証情報（公開鍵/秘密鍵ペアとその公開鍵の X.509 証明書）を, PC 用 $(pk_{\mathsf{PC}}, sk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ とスマートフォン用 $(pk_{\mathsf{smp}}, sk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ にそれぞれ生成する. また, $(pk_{\mathsf{smp}}, sk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ と $(pk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ を QR コードでスマートフォンに送信し, スマートフォンはこれらを保存する. さらに, ブラウザプラグインは, $(pk_{\mathsf{PC}}, sk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ と $(pk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ を PC に保存し, sk_{smp} を削除する.

ログイン情報登録プロトコル

(Add-1) ブラウザプラグインは, 登録する Web サービスのログイン情報 $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を鍵 k で暗号化し, 暗号化データ c_{site} を作成する.

(Add-2) ブラウザプラグインは, c_{site} をスマートフォンへ送信し, site と c_{site} を削除する.

(Add-3) スマフォアプリは, タグ（登録する Web サービス名等の任意の識別子）t_{site} をユーザに入力させ, $(t_{\mathsf{site}}, c_{\mathsf{site}})$ としてスマートフォンに保存する.

ログイン情報復元プロトコル

(Ret-1) スマフォアプリは, ログイン情報を復元させたい Web サービスの t_{site} をユーザに選択させ, 対応する c_{site} を PC に送信する.

(Ret-2) ブラウザプラグインは, 鍵 k を用いて, c_{site} から $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を復元する.

(Ret-3) ブラウザプラグインは, Web ブラウザの表示ページの URL が URL_{site} と一致することを確認し, 一致すれば $(ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を入力する.

図 1: MBCCO 方式のプロトコル

3 提案方式のプロトコル

本稿では, MBCCO 方式のフレームワークを拡張した新しい方式を提案する. 提案方式のパスワードマネージャの機能は, 初期設定, ログイン情報登録, ログイン情報復元, 端末復旧の 4 つに大別される. 本節では, 図 2 に示す各機能のプロトコルについて順に説明する. なお, サーバの公開鍵 pk_{svr} (対応する秘密鍵を sk_{svr} とする) には既に X.509 証明書 $X.509_{\mathsf{svr}}$ が発行済みであり, PC またはスマートフォンとサーバ間の通信には安全な TLS 通信が利用できると仮定する.

初期設定プロトコル はじめに, PC の Web ブラウザにプラグインを, スマートフォンにスマフォアプリを, またサーバにサーバアプリをインストールする.

つぎに, ユーザは, サーバアプリにログイン用パスワード P_{svr} を設定し, それを記憶する. また, ブラウザプラグインとスマフォアプリが, サーバアプリに安全な接続が可能になるために, サーバの IP アドレス情報をブラウザプラグインとスマフォアプリに設定し, さらに, $(pk_{\mathsf{svr}}, X.509_{\mathsf{svr}})$ を PC とスマートフォンに保存する.

最後に, MBCCO 方式と同様に, PC/スマートフォン間に安全な通信路を構築するために必要な情報を作成し, QR コードで PC からスマートフォンに直接的に送信する. この結果, 全ての端末の公開鍵とその X.509 証明書 $(pk_{\mathsf{D}}, X.509_{\mathsf{D}})$, $\mathsf{D} \in \{\mathsf{PC}, \mathsf{smp}, \mathsf{svr}\}$ を, ブラウザプラグインとスマフォアプリは使用できる.

ログイン情報登録プロトコル ログイン情報登録プロトコルは, ブラウザプラグイン（またはスマフォアプリ）で, ユーザが, ある Web サービス（URL_{site}）のログイン情報 $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ の登録を許可した場合に実行される.

ブラウザプラグイン（またはスマフォアプリ）は, site を $(2,3)$-秘密分散法により分散し, PC, スマートフォン, サーバ用の 3 つシェア $s^{(\mathsf{D})}_{\mathsf{site}}, \mathsf{D} \in \{\mathsf{PC}, \mathsf{smp}, \mathsf{svr}\}$ を作成する. また, 各シェア $s^{(\mathsf{D})}_{\mathsf{site}}$ を, それぞれの公開鍵 pk_{D} で暗号化し, $c^{(\mathsf{D})}_{\mathsf{site}}$ を作成する. さらに, ユーザが入力したタグ（登録する Web サービス名等の任意の識別子）t_{site} と, URL_{site} のハッシュ値 h_{site} を用いて, ログイン情報の分散暗号化データ

$$Z_{\mathsf{site}} = \left(h_{\mathsf{site}}, t_{\mathsf{site}}, c^{(\mathsf{PC})}_{\mathsf{site}}, c^{(\mathsf{smp})}_{\mathsf{site}}, c^{(\mathsf{svr})}_{\mathsf{site}}\right)$$

を作成し, PC とスマートフォンとサーバで共有する．そして，各端末 D は，$c_{\mathsf{site}}^{(\mathsf{D})}$ を自身の秘密鍵 sk_{D} で復元して $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s_{\mathsf{site}}^{(\mathsf{D})}\right)$ を保存し，Z_{site} を削除する．ただし，PC やスマートフォンは停止しているなどの理由で，リアルタイムで $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s_{\mathsf{site}}^{(\mathsf{D})}\right)$ が保存されない場合がある．そこで，サーバだけは，全ての端末が $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s_{\mathsf{site}}^{(\mathsf{D})}\right)$ を保存するまで，Z_{site} の削除を保留する．

ログイン情報復元プロトコル　ブラウザプラグインから，ログイン情報復元プロトコルを実行する場合，ユーザは，用いる補助端末をスマートフォンとサーバから選択する．

スマートフォンを選択した場合，MBCCO 方式と同様に，ユーザがスマートフォンでスマフォアプリを起動して，登録済み Web サービスのタグリストからログイン情報を復元させたいタグ t_{site} をタップすることで，スマートフォンから PC に，t_{site} に対応する $s_{\mathsf{site}}^{(\mathsf{smp})}$ が安全な TLS 通信路を利用して送信される．

一方，サーバを選択した場合，ブラウザプラグインは，サーバへのログイン用パスワード p をユーザに入力させ，Web ブラウザの表示中ページの URL のハッシュ値 h を計算し，サーバに接続リクエスト (p, h) を送信する．そして，サーバアプリは，パスワード認証が通過（$p = P_{\mathsf{svr}}$）すれば，$h = h_{\mathsf{site}}$ に対応するシェア $s_{\mathsf{site}}^{(\mathsf{svr})}$ を PC に送信する．また，サーバが削除を保留中の $\{Z_{\mathsf{site}}\}$ を確認し，未保存の $\left\{\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s_{\mathsf{site}}^{(\mathsf{D})}\right)\right\}$ があれば PC に保存する．

最後に，ブラウザプラグインは，PC のシェア $s_{\mathsf{site}}^{(\mathsf{PC})}$ と，補助端末から受信したシェア $s \in \left\{s_{\mathsf{site}}^{(\mathsf{smp})}, s_{\mathsf{site}}^{(\mathsf{svr})}\right\}$ より，秘密分散法の復元プロトコルによりログイン情報 site を復元し，MBCCO 方式と同様に，$(ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を入力する．

ところで，スマフォアプリからログイン情報復元プロトコルを実行し，補助端末に PC を選択した場合でも，前述の処理のスマートフォンを PC に読み替えて同様に実行可能である．ただし，スマフォアプリから実行する場合は，手元に PC が無い場合が多いと推定されるため，あまり実用的ではない．

端末復旧プロトコル　端末復旧プロトコルは，PC, スマートフォン，サーバの何れか一つの端末が失われた場合に，新しい代替の端末を用いて元の状態に復旧するためのプロトコルである．残った 2 つの端末を用いて全ての登録済みログイン情報を復元し，新端末を含む 3 つの端末を用いて，全てのログイン情報を再登録するのが処理の概要である．図 2 では，PC が失われた場合の復旧プロトコルのみを示しているが，スマートフォンが失われた場合の復旧プロトコルも，図 2 の処理の PC をスマートフォンに読み替えて同様に実行可能である．また，サーバが失われた場合の復旧プロトコルも，新端末の初期設定やシェアの送受信の手順が異なるだけで，ほぼ同様であり難しくはない．

4　セキュリティ評価

本節では，文献 [4] の手法を参考にして，提案方式の安全性を評価する．なお，攻撃者は，過去全ての通信を盗聴できる一方で，QR コードによる通信は盗聴できないと仮定する．

以下，PC, スマートフォン，サーバの何れか 1 つが攻撃者に盗まれた場合について述べる．

PC かスマートフォンの何れか 1 つが盗まれた場合　PC とスマートフォンのどちかが盗まれた場合も，同様に議論できるため，以下では PC が盗まれた場合についてのみ述べる．

PC が盗まれると，PC に保存されているシェアのリスト $\left\{s_{\mathsf{site}}^{(\mathsf{PC})}\right\}$ が攻撃者に知られてしまうことになる．したがって，攻撃者がスマートフォンかサーバのどちらかにあるシェアを手に入れると，ログイン情報を復元できてしまう．この場合，攻撃者は以下の方法でシェアを取得しようとすることが考えられる．(A) 通信路で送受信された情報を盗聴する．(B) ログイン情報復元プロトコルや端末復旧プロトコルを実行する．

これら攻撃への耐性について以下評価する．

(A) について　(Ini-3) で QR コードを用いて PC からスマートフォンに情報を送信する所以外は，全て安全な TLS 通信を使用している．したがって，通信路で送受信された情報から，シェアを取

得するには公開鍵暗号か共通鍵暗号を破る必要があり, 攻撃者がシェアを取得するのは困難と考えられる.

(B) について　攻撃者が, 盗んだ PC を操作し, ログイン情報復元プロトコルや端末復旧プロトコルを実行し, スマートフォン/サーバからシェアを取得しようと試みる場合を考える.

まず, スマートフォンからシェアを取得する試みについて考える. 図 2 のプロトコル (Ret-2) と (Rvr-3) において, スマートフォンからシェアを取得するためには, スマートフォンに表示された通知をタップするなどのスマートフォン本体への操作が必須である. したがって, 攻撃者にスマートフォン本体を直接操作されない限り, スマートフォンから PC にシェアが送信されることはない. すなわち, 攻撃者がスマートフォンからシェアを取得することは困難である.

つぎに, サーバからシェアを取得する試みについて考える. 攻撃者がサーバからシェアを取得するためには, 図 2 のプロトコル (Ret-3) と (Rvr-5) において, パスワード認証を通過する必要があるが, 攻撃者はパスワード P_{svr} を知らないため, サーバに対してオンライン攻撃を実施することになる. しかし, サーバアプリ側では, パスワード認証の試行回数の制限や, 攻撃検知 · 通知, またはシェアへの制限的アクセスなどの工夫により, 攻撃を困難にすることが可能でる. したがって, 攻撃者がサーバからシェアを取得することも困難である.

サーバが乗っ取られた場合　この場合, サーバに保存されているシェアのリスト $\left\{s_{\mathsf{site}}^{(\mathsf{svr})}\right\}$ と未同期のログイン情報の分散暗号化データ $\{Z_{\mathsf{site}}\}$ が攻撃者に取得される. したがって, 後は, PC またはスマートフォンのいずれか一方に保存されているシェアを取得できれば, 攻撃者は登録されているログイン情報を復元できる.

ところで, 攻撃者がサーバより取得した各 Z_{site} 中の $c_{\mathsf{site}}^{(\mathsf{PC})}, c_{\mathsf{site}}^{(\mathsf{smp})}$ は, PC とスマートフォンに分配されたシェアの暗号化データなので, 攻撃者はどちらかを解読できれば良い. しかし, そのためには公開鍵暗号を解読する必要があり, 困難である.

また, 図 2 の 4 つプロトコルは, 全てシェアを暗号化してサーバに送信している. したがって, これらのプロトコルを通じて, 攻撃者がシェアを取得することも困難である.

すなわち, 攻撃者に PC, サーバ, スマートフォンのうち 1 つが盗難にあったとしても, 登録済みのログイン情報の復元は困難と考えられる.

参考文献

[1] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," Proc. IEEE Symposium on Security and Privacy, pp. 538–552, 2012.

[2] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," `http://tools.ietf.org/html/rfc5246`, August 2008.

[3] D. Florencio and C. Herley, "A large-scale study of web password habits," Proc. 16th International Conference on World Wide Web, pp. 657–666, 2007.

[4] D. McCarney, D. Barrera, J. Clark, S. Chiasson and P.C. van Oorschot, "Tapas: Design, Implementation, and Usability Evaluation of a Password Manager," Proc. ACSAC'12, pp. 89–98, 2012.

[5] A. Shamir, "How to Share a Secret," Commun. ACM vol. 22, pp. 612–613, 1979.

[6] 独立行政法人情報処理推進機構, 情報セキュリティ白書 2014, 2014.

[7] AgileBits Inc., 1Password, `https://agilebits.com/onepassword`, 2014. [Online; accessed Aug 25th, 2014].

[8] LastPass Inc., LastPass, `https://lastpass.com/`, 2014. [Online; accessed Aug 25th, 2014].

初期設定プロトコル

(Ini-1) PC にはブラウザプラグインを, スマートフォンにはスマフォアプリ, またサーバにはサーバアプリをインストールする.

(Ini-2) ユーザは, サーバアプリにログイン用パスワード P_{svr} を設定し, それを記憶する. また, サーバの IP アドレス情報とサーバの公開鍵と X.509 証明書を, ブラウザプラグインとスマフォアプリに登録する.

(Ini-3) ブラウザプラグインは, TLS 通信認証情報（公開鍵/秘密鍵ペアとその公開鍵の X.509 証明書）を, PC 用 $(pk_{\mathsf{PC}}, sk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ とスマートフォン用 $(pk_{\mathsf{smp}}, sk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ にそれぞれ生成する. また, $(pk_{\mathsf{smp}}, sk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ と $(pk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ を QR コードでスマートフォンに送信し, スマートフォンはこれらを保存する. さらに, ブラウザプラグインは, $(pk_{\mathsf{PC}}, sk_{\mathsf{PC}}, X.509_{\mathsf{PC}})$ と $(pk_{\mathsf{smp}}, X.509_{\mathsf{smp}})$ を PC に保存し, sk_{smp} を削除する.

ログイン情報登録プロトコル

(Add-1) ブラウザプラグイン（またはスマフォアプリ）は, 登録する Web サービスのタグ（登録する Web サービス名等の任意の識別子）t_{site} をユーザに入力させ，また, URL_{site} のハッシュ値 h_{site} を計算する.

(Add-2) ブラウザプラグイン（またはスマフォアプリ）は, 登録する Web サービスのログイン情報 $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を, $(2,3)$-秘密分散法により分散し, PC/スマートフォン/サーバ用の 3 つシェア $s^{(\mathsf{D})}_{\mathsf{site}}, \mathsf{D} \in \{\mathsf{PC}, \mathsf{smp}, \mathsf{svr}\}$ を作成する. また, 各シェア $s^{(\mathsf{PC})}_{\mathsf{site}}$ を, それぞれの公開鍵 pk_{D} で暗号化し, $c^{(\mathsf{D})}_{\mathsf{site}}$ を作成する.

(Add-3) ブラウザプラグイン（またはスマフォアプリ）は, PC とスマートフォンとサーバでログイン情報の分散暗号化データ $Z_{\mathsf{site}} = \left(h_{\mathsf{site}}, t_{\mathsf{site}}, c^{(\mathsf{PC})}_{\mathsf{site}}, c^{(\mathsf{smp})}_{\mathsf{site}}, c^{(\mathsf{svr})}_{\mathsf{site}}\right)$ の情報を共有し, 各端末 D は, $c^{(\mathsf{D})}_{\mathsf{site}}$ を自身の秘密鍵 sk_{D} で復元して $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{D})}_{\mathsf{site}}\right)$ を保存し, Z_{site} を削除する. ただし, サーバだけは, 全端末が $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{D})}_{\mathsf{site}}\right)$ を保存するのを待ってから, Z_{site} を削除する.

ログイン情報復元プロトコル

(Ret-1) ブラウザプラグインは, ログイン情報復元処理の補助に用いる端末を, スマートフォンとサーバから, ユーザに選択させる.

(Ret-2) ユーザがスマートフォンを選択した場合, スマフォアプリは, ログイン情報を復元させたい Web サービスの t_{site} をユーザに選択させ, 対応するシェア $s^{(\mathsf{smp})}_{\mathsf{site}}$ を PC に送信する.

(Ret-3) ユーザがサーバを選択した場合, ブラウザプラグインは, サーバへのログイン用パスワード p をユーザに入力させ, Web ブラウザの表示中ページの URL のハッシュ値 h を計算し, サーバに接続リクエスト (p, h) を送信する. サーバアプリは, パスワード認証が通過（$p = P_{\mathsf{svr}}$）すれば, 以下を行う.

(a) $h = h_{\mathsf{site}}$ に対応するシェア $s^{(\mathsf{svr})}_{\mathsf{site}}$ を PC に送信する.

(b) サーバが削除を保留する $\{Z_{\mathsf{site}}\}$ を確認し, 未保存の $\left\{\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{D})}_{\mathsf{site}}\right)\right\}$ を PC に保存する.

(Ret-4) ブラウザプラグインは, PC のシェア $s^{(\mathsf{PC})}_{\mathsf{site}}$ と, 補助端末から受信したシェア $s \in \left\{s^{(\mathsf{smp})}_{\mathsf{site}}, s^{(\mathsf{svr})}_{\mathsf{site}}\right\}$ より, ログイン情報 $\mathsf{site} = (URL_{\mathsf{site}}, ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を復元する. また, Web ブラウザの表示ページの URL が URL_{site} と一致することを確認し, 一致すれば $(ID_{\mathsf{site}}, Pass_{\mathsf{site}})$ を入力する.

端末復旧プロトコル

(Rvr-1) ユーザは, 新 PC にブラウザプラグインをインストールし, サーバの IP アドレス情報を設定する.

(Rvr-2) プロトコル (Ini-3) を実行し, PC/スマートフォン間に安全な TLS 通信路を再構築する.

(Rvr-3) スマフォアプリは, シェアのリスト $\left\{\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{smp})}_{\mathsf{site}}\right)\right\}$ を新 PC に送信し, 新 PC はそれを保存する.

(Rvr-4) ブラウザプラグインは, サーバへのログイン用パスワード p をユーザに入力させ, サーバに接続リクエスト (p, Rvr) を送信する. ここで, Rvr は端末普及リクエストコードである.

(Rvr-5) サーバアプリは, パスワード認証が通過（$p = P_{\mathsf{svr}}$）すれば, シェアのリスト $\left\{\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{svr})}_{\mathsf{site}}\right)\right\}$ を新 PC に送信し, 新 PC はそれを保存する.

(Rvr-6) ブラウザプラグインは, 各 $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{smp})}_{\mathsf{site}}, s^{(\mathsf{svr})}_{\mathsf{site}}\right)$ について, 秘密分散法の復元アルゴリズムにより site を復元し, プロトコル (Add-2), (Add-3) を順番に実行し, 全ての端末の $\left(h_{\mathsf{site}}, t_{\mathsf{site}}, s^{(\mathsf{D})}_{\mathsf{site}}\right)$ を更新する.

図 2: 提案するパスワードマネージャ