3W-06

相手認証における鍵管理負担軽減方法の提案

木村　和紀†　　土井　洋†
情報セキュリティ大学院大学　情報セキュリティ研究科†

1. はじめに

インターネットバンキング等において「パスワード」＋「ワンタイムパスコード」のような二要素認証が利用されている．しかし，この方法はワンタイムパスコードを生成するデバイスをサービス毎に用意しなければならない等，サービス提供者及びクライアントの負担が大きい．本稿では，クライアントの負担が少ない Security Keys をベースとして，クライアントに加えサービス提供者の鍵管理負担をも軽減する方法を提案する．

2. FIDO と Security Keys

FIDO[1]は，認証における利便性向上と負担軽減を目指しており，2 つのプロトコル（UAF と U2F）がある．本稿で取り扱う U2F は，パスワードとデバイスの所持認証を組み合わせた二要素認証であり，Lang らは Security Keys[2]において，Google のアカウントサービスにおける U2F を使用した二要素認証について評価している．

Security Keys では，クライアントが所持するデバイスのセキュア・エレメントでサービス毎の鍵ペア(公開鍵と秘密鍵)を生成する．秘密鍵等をデバイス固有の鍵で暗号化したものをキーハンドルと呼ぶ．Security Keys ではキーハンドルをサービス提供者のサーバに保管する方法を採っている．これにより，クライアント側（デバイス）に，利用するサービス毎の鍵（秘密鍵）を保存しなくて良いという利点が生じる．なお，デバイスのセキュア・エレメントには，工場出荷時にアテステーション秘密鍵が格納されている．対応する公開鍵は FIDO Metadata Service からネットワーク経由で取得可能である．本研究では，Security Keys の方式を元にして，鍵の管理を極力第三者が行い，かつ安全に通信できる方法を検討した．

3. 相手認証における鍵管理負担軽減方法

3.1 秘密鍵管理とその負担について

相手認証の場合，サービス利用者(認証される側)のみが持つ情報をサービス提供者(認証する側)に示すことが基本となり，暗号技術における秘密鍵などもそれに該当する．なお，これらの情報は暗号化等を行い，盗聴されても問題ない形に変換する．

さて，秘密の情報はサービス利用者が責任を持って管理すべきものであるが，秘密の情報をセキュア・エレメント等に格納し，認証および鍵の処理などの大半をデバイス(プログラム)に行わせることが考えられる．

第 2 章で述べたように，Security Keys[2]ではクライアントがサービス毎に鍵ペアを新たに生成し，秘密鍵を暗号化した後にサービス提供者のサーバで保管する方法を採用している．しかし，サービス利用者の秘密の情報が暗号化されているとはいえ，中小規模のサービス提供者のサーバでこれらの情報を管理するのは容易ではない．

3.2 暗号化された鍵の保管方法について

クライアントおよびサービス提供者の鍵管理負担を軽減する方法として，鍵を信頼できる第三者機関（以下，「KSC(Key Storage Center)」という）が暗号化した形で保管する方法を提案する．KSC は，同一業界で共同運営を行い，それぞれのサービス毎生成された鍵を保管する場合などが想定される．同一業界で KSC を共同運営することにより，サービス提供者が各々秘密鍵を保管する負担を軽減することができ，新規にこの方式を導入する際の障壁も比較的低くなるものと思われる．

4. 提案方式

本提案における「登録」フェーズと「認証」フェーズにおけるプロトコルについて述べる．

4.1 登録

登録フェーズにおいて実現したいことは，「サービス提供者がクライアントの鍵を保管しなくて済むようにする」ことである．これを実現するために，クライアントが生成した公開鍵に対してサービス提供者が署名を行い，この署名を KSC に保管する方法を採る．以下に，登録時の手順を示す．

(1) クライアントの登録要求に対し，サービス提供者は鍵ペアの生成要求を行う(①,②)．なお，この鍵ペアは利用するサービスと紐づいて利用されるものである．
(2) 当該サービスに応じた鍵ペア(sk_{CS}, pk_{CS})を生成し，公開鍵 pk_{CS} にアテステーション秘密鍵 sk_{att} でデジタル署名 σ_{CS} を生成する(③,④)．

Proposal to reduce the burden on a key storage for authentication
† Kazunori Kimura, Hiroshi Doi
Institute of Information Security

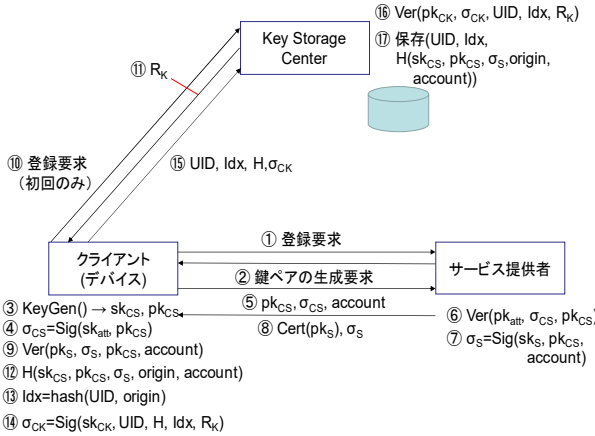

図 1 登録フェーズ

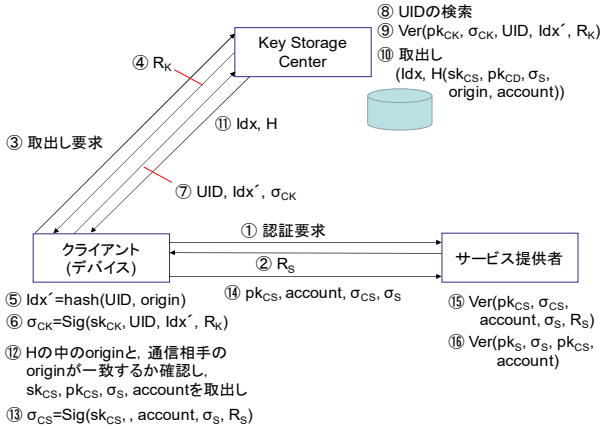

図 2 認証フェーズ

(3) ③で生成された公開鍵 pk_{CS} と④の署名 σ_{CS} を，サービス提供者に送信する(⑤)．
(4) サービス提供者は，アテステーション秘密鍵に対応する公開鍵 pk_{att} で，署名 σ_{CS} を検証したうえで，クライアントの公開鍵 pk_{CS} に対して，サービス提供者自身の秘密鍵 sk_S でデジタル署名 σ_S を生成する(⑥,⑦)．
(5) ⑦のデジタル署名 σ_S を，公開鍵証明書をつけてクライアントに送信し，クライアントは署名 σ_S を検証する(⑧,⑨)．
(6) 事前にクライアントは KSC との通信で使用する鍵ペア(sk_{CK}, pk_{CK})を生成し，公開鍵 pk_{CK} を KSC に保存しておく．
(7) クライアントの KSC に対する登録要求に対し，KSC は乱数 R_K を送信する(⑩,⑪)．
(8) クライアントはキーハンドル H，署名 σ_{CK} 等を生成し，KSC に送信する．なお，キーハンドル H は，秘密鍵 sk_{CS}，公開鍵 pk_{CS}，サービス提供者の署名 σ_S などを暗号化したうえで格納している(⑫～⑮)．
(8) KSC は署名 σ_{CK} を検証し，キーハンドル等を格納する(⑯,⑰)．

4.2 認証

認証時にサービス提供者が確認したいことは，「登録フェーズで生成した，(自身の署名がなされている)クライアントの公開鍵で検証できる署名を生成できるか」である．これにより，認証フェーズにおいても登録フェーズと同一の相手が通信相手であることを確認できる．以下に，認証手順を示す．

(1) クライアントの認証要求に対し，サービス提供者は乱数 R_S を送信する(①,②)．
(2) クライアントの KSC に対するキーハンドルの取出要求に対し，KSC は乱数 R_K を送信する(③,④)．
(3) 乱数 R_K 等に対する署名 σ_{CK} 等を生成し，ユーザ ID とともに KSC に送信する(⑤～⑦)．
(4) KSC は該当するユーザ ID を検索し，署名 σ_{CK} を検証すると，キーハンドル H などを取り出し，クライアントに送信する(⑧～⑪)．
(5) キーハンドル H を受け取ったクライアントは，キーハンドルを復号し，秘密鍵 sk_{CS} やサービス提供者の署名 σ_S などを取り出す(⑫)．
(6) サービス提供者から送信されていた乱数 R_S に，キーハンドルから取り出した秘密鍵 sk_{CS} で署名し，σ_S などとともにサービス提供者に送信する(⑬,⑭)．
(7) サービス提供者は乱数 R_S に対する署名を pk_{CS} で検証するとともに，登録フェーズで生成した pk_{CS} に対する自身の署名 σ_S を検証する(⑮,⑯)．

5. まとめ

提案方式では，サービスの登録時に，ユーザの公開鍵に対してサービス提供者自らが署名を行う．これを認証フェーズにおけるトラストアンカーとしている．これにより，サービス提供者は認証局の役割を兼ねることができる．よって，サービス提供者は認証に利用するクライアントの鍵を保管する必要がなくなり，第三者の鍵保管センターが，暗号化された秘密鍵を管理することができる．

クライアントにとっては，アテステーション秘密鍵，キーハンドルを生成する暗号化鍵，及び KSC との通信で使用する秘密鍵をセキュア・エレメントに保存しておくだけでよく，利用するサービスの数だけ鍵を管理する必要がない．

以上により，クライアント，サービス提供者双方の鍵管理負担を軽減することができるものと考える．

参考文献

[1] FIDO Alliance, “FIDO U2F Architectural Overview” (2017)
[2] Lang et al, ”Security keys: Practical cryptographic second factors for the modern web” , FC 2016, pp.422-440, Springer (2017)