Workgroup: Tigress
Internet-Draft:
draft-tigress-sample-implementation-01
Published: 9 November 2022
Intended Status: Informational
Expires: 13 May 2023
Authors: D. Vinokurov A. Bulgakov J. L. Giraud
Apple Inc Apple Inc Apple Inc
C. Astiz A. Pelletier J. Hansen
Apple Inc Apple Inc Apple Inc

Transfer Digital Credentials Securely: sample implementation and threat model

Abstract

This document describes a sample implementation of Tigress internet draft [Tigress-00] and a threat model of this implementation.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/dimmyvi/tigress-sample-implementation. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tigress-sample-implementation/.

Source for this draft and an issue tracker can be found at https://github.com/dimmyvi/tigress-sample-implementation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 May 2023.

Copyright Notice

Table of Contents

1. Introduction

This document provides a sample implementation and threat model for Tigress draft [Tigress-00].

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

*DCK - Digital Car Key

*AP - Application Processor

*TTL - Time To Live

*AAA - Apple Anonymous Attestation - a subtype of WebAuthn [WebAuthn-2]

*PKI - public Key Infrastructure

*UUID - a unique identifier defined in [RFC4122]

*SMS - Short Message Service

*OS - Operating System

*URI - Uniform Resource Identifier

*URL - Universal Resource Locator

*RNG - Random Number Generator

3. Sample Implementation - Digital Car Key sharing example.

*An Owner's device (Sender) starts sharing flow with selection of credential entitlements for the key shared - e.g. access entitlements (allow open the car, allow start the engine, allow to drive the car), time of sharing - e.g. from 09/01/2022 to 09/03/2022, then generates a KeyCreationRequest (per [CCC-Digital-Key-30]).

*The Owner's device generates a new symmetric encryption key (Secret) and builds an encrypted KeyCreationRequest. Then it generates an attestation data, that follows a WebAuthn API [WebAuthn-2], specific to Apple - AAA, which covers the encrypted content. Owner device makes a call to Relay server (Intermediary) - createMailbox, passing over the encrypted content, device attestation, mailbox configuration (mailbox time-to-live, access rights - Read/Write/Delete), preview (display information) details, its push notification token and a unique deviceClaim.

*Relay server verifies device attestation using WebAuthn verification rules specific to attestation data used, including verifying device PKI certificate in attestation blob. Relay server creates a mailbox, using mailboxConfiguration received in the request and stores encrypted content in it.

*The mailbox has a time-to-live which defines when it is to expire and be deleted by the Relay server. This time is limited by the value that can be considered both sufficient to complete the transfer and secure against brute force attacks on the encrypted content - e.g. 48 hours.

*Relay server generates a unique mailboxIdentifier value, that is hard to predict - i.e. using UUID - and builds a full URL (shareURL) referencing the mailbox - e.g. "https://www.example.com/v1/m/2bba630e-519b-11ec-bf63-0242ac130002", which it returns to the Owner device.

*Owner's device locally stores the shareURL and the Secret and sends the shareURL with optional vertical in URL parameter and mandatory secret in Fragment part (e.g. "https://www.example.com/v1/m/2bba630e-519b-11ec-bf63-0242ac130002?v=c#hXlr6aRC7KgJpOLTNZaLsw==") to the Friend's device (Receiver) over SMS.

*Friend's device receives the shareURL in SMS, messaging application makes an automatic GET call to shareURL (excluding Fragment part - Secret) - and fetches a preview (Display Information) html page with OpenGraph tags in the head:

```
<html prefix="og: https://ogp.me/ns#">
<head>
 <title>Shared Key</title>
 <meta content="Shared Key" property="og:title"/>
 <meta content="You've been invited to add a shared digital car key to y
 <meta content="https://example.com/displayInfo/general.png" property="o
 <meta content="https://example.com/displayInfo/general.png" property="o
 <meta content="200" property="og:image:width"/>
 <meta content="100" property="og:image:height"/>
</head>
</html>
```

Figure 1: OpenGraph preview of a credential

*Messaging application shows a preview of the DCK on the Friend's device that Owner wants to share with them. User accepts the shareURL by clicking on the preview in the messaging application. Messaging application redirects the user to wallet (credential manager application) using a deep link mechanism embedded into the OS.

*Wallet receives the shareURL with the Secret in the Fragment. Friend's device checks if the Relay server is in allow-list of accepted Relay servers.

*Wallet reads secure content from the mailbox using shareURL (without the Fragment part) with ReadSecureContentFromMailbox method, passing a unique deviceClaim with the request. Thus, relay server binds the mailbox (identified by mailboxIdentifier) with the Owner's device (with Owner's device deviceClaim at the mailbox creation time) and the Friend's device (with Friend's device deviceClaim at the first time Friend's device calls ReadSecureContentFromMailbox for the mailbox). Now only these 2 devices are allowed to read and write secure content to this particular mailbox. This secures the message exchange and

prevents other devices from altering the exchange between Owner and Friend.

*Friend's device decrypts secure content using Secret and extracts KeyCreationRequest (ref to [CCC-Digital-Key-30] specification).

*Friend's device generates a KeySigningRequest (ref to [CCC-Digital-Key-30] specification), encrypts it with Secret and uploads to the mailbox with UpdateMailbox call to Relay server, providing its unique deviceClaim and push notification token.

*Relay server sends a push notification to Owner's device via Push Notification Server.

*Owner device, having received a push notification message, reads secure content from the mailbox using shareURL with ReadSecureContentFromMailbox method, passing its unique deviceClaim with the request. Owner's device decrypts secure content using Secret and extracts KeySigningRequest (ref to [CCC-Digital-Key-30] specification).

*Owner's device signs the Friend's device public key with Owner's private key and creates a KeyImportRequest (ref to [CCC-Digital-Key-30] specification). Owner's device encrypts it with the Secret and uploads to the mailbox with UpdateMailbox call to Relay server, providing its unique deviceClaim.

*Relay server sends a push notification to Friend's device via Push Notification Server.

*Friend's device, having received a push notification message, reads secure content from the mailbox using shareURL with ReadSecureContentFromMailbox method, passing its unique deviceClaim with the request. Friend device decrypts secure content using Secret and extracts KeyImportRequest (ref to [CCC-Digital-Key-30] specification). Friend's device provisions the new credential to the wallet and deletes the mailbox with DeleteMailbox call to the Relay server. As an additional security measure, Friend device asks for a verification code (PIN code) generated by Owner's device and communicated to Friend out-of-band.

4. Threat Model

Threat model for the sample implementation is provided at the following URL: [threat_model]: https://github.com/dimmyvi/tigress-sample-implementation/blob/main/threat_model.png "Threat model for Tigress sample implementation"

Item	Asset	Threat	Impact	Mitigation	Comment
1	Owner's DCK	Kicking-off arbitrary key sharing by spoofing user identity	DCK becomes shared with arbitrary user/ adversary allowing them access to the Owner's car	1) User auth (face/ touch ID), 2) Secure Intent	
2	Content on Intermediary server	Content recovery by brute forcing secret	Exposure of encrypted content and key redemption	1) Strong source of randomness for salt, 2) At least 128 bit key length, 3) Limitted TTL of the mailbox	
3	Content on Intermediary server	Content recovery by intercepting secret	Ability to decrypt content on Intermediary server	1) Physical separation between content and secret, e.g. secret sent as URI fragment to recipient, 2) Optional second factor(e.g. Device PIN, Activation Options - please refer to CCC Technical Specification) can be proposed to the user via user notification based on security options of selected primary sharing channel (used to share URL with secret)	
4	Content on Intermediary server	Access to content by multiple arbitrary users/devices	1) Adversary can go to partner and redeem the shared key, 2) Adversary can send push notifications	1) Mailboxes identified by version 4 UUID defined in [RFC4122] (hard to guess/ bruteforce), 2) Mailboxes 'tied' to sender and recipient (trust on first use via deviceClaim), 3) TTL limit for mailboxes, 4) Mailboxes deleted after pass redemption	
5	Content on Intermediary server	Compromised Intermediary server	1) Adversary can redeem the sharedKey, 2) Adversary can	1) Separation between content and secret, e.g. secret sent as URI fragment to recipient,	

Item	Asset	Threat	Impact	Mitigation	Comment
			send push notifications	2) TTL limit for mailboxes	
6	Content on Intermediary server and Push Tokens	Unauthenticated access to mailbox on Intermediary server	1) Adversary can redeem the sharedKey, 2) Adversary can send push notifications	1) Mailboxes identified by version 4 UUID defined in [RFC4122] (hard to guess/ bruteforce), 2) Mailboxes 'tied' to sender and recipient (trust on first use via deviceClaim), 3) TTL limit for mailboxes, 4) Mailboxes deleted after pass redemption	
7	Content on Intermediary server	User stores non-credential information in mailbox (e.g. "cat pictures")	Service abuse, Adversary can use Intermediary server as cloud storage	1) Mailboxes have size limit, 2) Mailboxes have TTL	
8	Device PIN	Receiver device compromised (redemption before friend)	Device PIN can exposure and forwarding to an advarsary	Activation Options as defined in [CCC-Digital-Key-30], Section 11.2 Sharing Principles, subsection 11.2.1.3. Activation Options	
9	Device PIN	Weak PIN can be easily guessed	Anyone with share URL in their possession can guess the PIN and redeem the key	1) Use of strong RNG as a source to generate Device PIN, 2) Long enough PIN (e.g. 6 digits) as per [NIST-800-63B] recommendations, 3) Limit the number of retries (e.g. Device PIN retry counter + limit) as per [NIST-800-63B] recommendations	[NIST-800-63B] section 5.1.1. Memorized Secret Authenticators
10	Device PIN	Eavesdropping on weak msg channels/app	PIN exposure would allow one with possession of share URL and	In person, out of band PIN transfer, e.g. voice channel	

Item	Asset	Threat	Impact	Mitigation	Comment
			Secret to redeem key		
11	Device PIN	PIN recovery via timing attack	Adversary with shared URL in possession can recover PIN based on the response delay, in the case where the PIN verification is not invariant	1) Time invariant compare, 2) PIN retry counter/limit	
12	Device PIN retry counter/ limit	Device PIN brute force	Device PIN successful guess	1) Use of strong RNG as a source to generate Device PIN, 2) Long enough PIN (e.g. 6 digits) as per [NIST-800-63B] recommendations, 3) Limit the number of retries (e.g. Device PIN retry counter + limit) as per [NIST-800-63B] recommendations	[NIST-800-63B] section 5.1.1. Memorized Secret Authenticators
13	Sharing Invitation	Messaging channel eavesdropping	Share invitation forwarding and DCK redemtion by malicious party	1) Send invitation and Device PIN via different channels, e.g. Device PIN can be shared out of band (over voice), 2) Use of E2E encrypted msg apps/ chhannel	
14	Sharing Invitation	Voluntary/ Involuntary forwarding by Friend	DCK redemption before Friend	Use of messaging apps with anti-forwarding mechanisms(e.g. hide link, copy/past prevention)	
15	Sharing Invitation	Friend device compromise allow malware to forward invitation to an adversary	Share invitation forwarding and key redemption by	Activation Options as defined in [CCC-Digital-Key-30], Section 11.2 Sharing Principles, subsection	

Item	Asset	Threat	Impact	Mitigation	Comment
			malicious party	11.2.1.3. Activation Options	
16	Sharing Invitation	User mistakenly shares with the wrong person	DCK redemption by adversary/not intended user	1) Send invitation and Device PIN via different channels, e.g. Device PIN can be shared out of band (over voice), 2) DCK revocation	
17	Sharing Invitation	Owner device compromise allow malware to forward invitation to an adversary	Share invitation forwarding and key redemption by malicious party	Activation Options as defined in [CCC-Digital-Key-30], Section 11.2 Sharing Principles, subsection 11.2.1.3. Activation Options	
18	Sharing Invitation	Friend device OEM account take over	DCK provisioning on adversary's device	1) Binding to deviceClaim, 2) Device PIN shared out of band, 3) Activation Options as defined in [CCC-Digital-Key-30], Section 11.2 Sharing Principles, subsection 11.2.1.3. Activation Options	
19	User's credentials, payment card details, etc	Phishing attacks leveraging malicious Java Script in preview page	1) Preview page URL fragement contains encryption key - meaning malicious JS could use key to decrypt contents, 2) Malicious Java Script can phish for user credentials, payment card information, or other sensitive data	1) Properly vet Java Script that is embedded in the preview page, 2) Define strong content security policy	

Table 1

5. Security Considerations

TODO Security

6. IANA Considerations

This document has no IANA actions.

7. Normative References

[CCC-Digital-Key-30] Car Connectivity Consortium, "Digital Key Release 3", July 2022, <https://carconnectivity.org/download-digital-key-3-specification/>.

[NIST-800-63B] NIST, "NIST Special Publication 800-63B, Digital Identity Guidelines", November 2022, <https://pages.nist.gov/800-63-3/sp800-63b.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/rfc/rfc4122>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[Tigress-00] Vinokurov, D., Byington, M., Lerch, M., Pelletier, A., and N. Sha, "Transfer Digital Credentials Securely", November 2022, <https://datatracker.ietf.org/doc/draft-art-tigress/>.

[WebAuthn-2] W3, "Web Authentication - An API for accessing Public Key Credentials - Level 2", April 2021, <https://www.w3.org/TR/webauthn-2/>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Dmitry Vinokurov
Apple Inc

Email: dvinokurov@apple.com

Alexey Bulgakov
Apple Inc

Email: abulgakov@apple.com

Jean-Luc Giraud
Apple Inc

Email: jgiraud@apple.com

Casey Astiz
Apple Inc

Email: castiz@apple.com

Alex Pelletier
Apple Inc

Email: a_pelletier@apple.com

Jake Hansen
Apple Inc

Email: jake.hansen@apple.com