NIST SPECIAL PUBLICATION 1800-13B

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Volume B:
Approach, Architecture, and Security Characteristics

Paul Grassi
Applied Cybersecurity Division
Information Technology Laboratory

Bill Fisher
National Cybersecurity Center of Excellence
Information Technology Laboratory

Spike E. Dog
Santos Jha
William Kim
Taylor McCorkill
Joseph Portner
Mark Russell
Sudhi Umarji
The MITRE Corporation
McLean, VA

William C. Barker
Dakota Consulting
Silver Spring, MD

April 2018

DRAFT
This publication is available free of charge from:
https://www.nccoe.nist.gov/projects/use-cases/mobile-sso

National Institute of Standards and Technology
U.S. Department of Commerce

NCCoE
NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST or NCCoE, nor is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-13B, Natl. Inst. Stand. Technol. Spec. Publ. 1800-13B, 73 pages, (April 2018), CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback. As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: psfr-nccoe@nist.gov.

Public comment period: April 16, 2018 through June 18, 2018

All comments are subject to release under the Freedom of Information Act (FOIA).

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in IT security—the NCCoE applies standards and best practices to develop modular, easily adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cyber Security Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Md.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication Series 1800) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align more easily with relevant standards and best practices and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

ABSTRACT

On-demand access to public safety data is critical to ensuring that public safety and first responder (PSFR) personnel can deliver the proper care and support during an emergency. This requirement necessitates heavy reliance on mobile platforms while in the field, which may be used to access sensitive information, such as personally identifiable information (PII), law enforcement sensitive (LES) information, or protected health information (PHI). However, complex authentication requirements can hinder the process of providing emergency services, and any delay—even seconds—can become a matter of life or death.

In collaboration with NIST'S Public Safety Communications Research lab (PSCR) and industry stakeholders, the NCCoE aims to help PSFR personnel to efficiently and securely gain access to mission data via mobile devices and applications (apps). This practice guide describes a reference design for multifactor authentication (MFA) and mobile single sign-on (MSSO) for native and web apps, while improving interoperability between mobile platforms, apps, and identity providers, irrespective of the app development platform used in their construction. This NCCoE practice guide details a collaborative

effort between the NCCoE and technology providers to demonstrate a standards-based approach using commercially available and open-source products.

This guide discusses potential security risks facing organizations, benefits that may result from the implementation of an MFA/MSSO system, and the approach that the NCCoE took in developing a reference architecture and build. This guide includes a discussion of major architecture design considerations, an explanation of the security characteristics achieved by the reference design, and a mapping of the security characteristics to applicable standards and security control families.

For parties interested in adopting all or part of the NCCoE reference architecture, this guide includes a detailed description of the installation, configuration, and integration of all components.

KEYWORDS

access control; authentication; authorization; identity; identity management; identity provider; single sign-on; relying party

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Donna Dodson	NIST NCCoE
Tim McBride	NIST NCCoE
Jeff Vettraino	FirstNet
FNU Rajan	FirstNet
John Beltz	NIST Public Safety Communications Research Lab
Chris Leggett	Ping Identity
Paul Madsen	Ping Identity
John Bradley	Yubico
Adam Migus	Yubico
Derek Hanson	Yubico
Adam Lewis	Motorola Solutions
Mike Korus	Motorola Solutions
Dan Griesmann	Motorola Solutions

Name	Organization
Arshad Noor	StrongAuth
Pushkar Marathe	StrongAuth
Max Smyth	StrongAuth
Scott Wong	StrongAuth
Akhilesh Sah	Nok Nok Labs
Avinash Umap	Nok Nok Labs

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

Technology Partner/Collaborator	Build Involvement
Ping Identity	Federation Server
Motorola Solutions	Mobile Apps
Yubico	External Authenticators
Nok Nok Labs	Fast Identity Online (FIDO) Universal Authentication Framework (UAF) Server
StrongAuth	FIDO Universal Second Factor (U2F) Server

Contents

List of Figures

List of Tables

1 Summary

The National Cybersecurity Center of Excellence (NCCoE), with the National Institute of Standards and Technology's (NIST's) Public Safety Communications Research (PSCR) lab, is helping the public safety and first responder (PSFR) community address the challenge of securing sensitive information accessed on mobile applications (apps). The Mobile Application Single Sign-On (SSO) Project is a collaborative effort with industry and the information technology (IT) community, including vendors of cybersecurity solutions.

This project aims to help PSFR personnel efficiently and securely gain access to mission-critical data via mobile devices and applications through mobile SSO, identity federation, and multifactor authentication (MFA) solutions for native and web applications by using standards-based commercially available and open-source products.

The reference design herein:

- provides a detailed example solution and capabilities that address risk and security controls
- demonstrates standards-based MFA, identity federation, and mobile SSO for native and web applications
- supports multiple authentication methods, considering unique environmental constraints faced by first responders in emergency medical services, law enforcement, and fire services

1.1 Challenge

On-demand access to public safety data is critical to ensuring that PSFR personnel can protect life and property during an emergency. Mobile platforms offer a significant operational advantage to public safety stakeholders by providing access to mission-critical information and services while deployed in the field, during training and exercises, or when participating in the day-to-day business and preparing for emergencies during non-emergency periods. These advantages can be limited if complex authentication requirements hinder PSFR personnel, especially when a delay—even seconds—is a matter of containing or exacerbating an emergency situation. PSFR communities are challenged with implementing efficient and secure authentication mechanisms to protect access to this sensitive information, while meeting the demands of their operational environment.

Many public safety organizations (PSOs) are in the process of transitioning from traditional land-based mobile communications to high-speed, regional or nationwide, wireless broadband networks (e.g., First Responder Network Authority [FirstNet]). These emerging 5G systems employ internet protocol (IP)-based communications to provide secure and interoperable public safety communications to support initiatives, such as Criminal Justice Information Services (CJIS); Regional Information Sharing Systems (RISS); and international justice and public safety services, such as those provided by Nlets, the International Justice and Public Safety Network. This transition will foster critically needed interoperability within and among jurisdictions, but will create a significant increase in the number of mobile Android and iPhone operating system (iOS) devices that PSOs will need to manage.

Current PSO authentication services may not be sustainable in the face of this growth. There are needs to improve security assurance, limit authentication requirements that are imposed on users (e.g., avoid the number of passwords that are required), improve the usability and efficiency of user account management, and share identities across jurisdictional boundaries. Currently, there is no single management or administrative hierarchy spanning the PSFR population. PSFR organizations operate in a variety of environments with different authentication requirements. Standards-based solutions are needed to support technical interoperability and this diverse set of PSO environments.

1.1.1 Easing User Authentication Requirements

Many devices that digitally access public safety information employ different software applications to access different information sources. Single-factor authentication processes, usually passwords, are most commonly required to access each of these applications. Users often need different passwords or personal identification numbers (PINs) for each application used to access critical information. Authentication prompts, such as entering complex passwords on a small touchscreen for each application, can hinder PSFRs. There is an operational need for the mobile systems on which they rely to support a single authentication process that can be used to access multiple applications. This is referred to as single sign-on, or SSO.

1.1.2 Improving Authentication Assurance

Single-factor password authentication mechanisms for mobile native and web applications may not provide sufficient protection for control of access to law enforcement–sensitive (LES), protected health information (PHI), or personally identifiable information (PII). Replacement of passwords by multifactor technology (e.g., a PIN, plus some physical token or biometric) is widely recognized as necessary for access to sensitive information. Technology for these capabilities exists, but budgetary, contractual, and operational considerations have impeded the implementation and use of these technologies. PSOs need a solution that supports differing authenticator requirements across the community (e.g., law enforcement, fire response, emergency medical services) and a “future proof” solution allowing for the adoption of evolving technologies that may better support PSFRs in the line of duty.

1.1.3 Federating Identities and User Account Management

PSFRs need access to a variety of applications and databases to support routine activities and emergency situations. These resources may be accessed by portable mobile devices or mobile data terminals in vehicles. It is not uncommon for these resources to reside within neighboring jurisdictions at the federal, state, county, or local level. Even when the information is within the same jurisdiction, it may reside in a third-party vendor’s cloud service. This environment results in the issuance of many user accounts to each PSFR that are managed and updated by those neighboring jurisdictions or cloud service providers. When a PSFR leaves or changes job functions, the home organization must ensure that accounts are deactivated, avoiding any orphaned accounts managed by third parties. PSOs need a solution that reduces the number of accounts managed and allows user account and credentials issued by a PSFR's home organization to access information across jurisdictions and with cloud services. The ability of one organization to accept the identity and credentials from another organization, in the form of an identity assertion, is called identity federation. Current commercially available standards support this functionality.

1.2 Solution

This NIST Cybersecurity Practice Guide demonstrates how commercially available technologies, standards, and best practices implementing SSO, identity federation, and MFA can meet the needs of PSFR communities when accessing services from mobile devices.

In our lab at the NCCoE, we built an environment that simulates common identity providers (IdPs) and software applications found in PSFR infrastructure. In this guide, we show how a PSFR entity can leverage this infrastructure to implement SSO, identity federation, and MFA for native and web applications on mobile platforms. SSO, federation, and MFA capabilities can be implemented independently, but implementing them together would achieve maximum improvement with respect to usability, interoperability, and security.

At its core, the architecture described in Section 4 implements the Internet Engineering Task Force's (IETF's) Best Current Practice (BCP) guidance found in Request for Comments (RFC) 8252, *OAuth 2.0 for Native Apps* [1]. Leveraging technology newly available in modern mobile operating systems (OSs), RFC 8252 defines a specific flow allowing for authentication to mobile native applications without exposing user credentials to the client application. This authentication can be leveraged by additional mobile native and web applications to provide an SSO experience, avoiding the need for the user to manage credentials independently for each application. Using the Fast Identity Online (FIDO) universal authentication framework (UAF) [2] and universal second factor (U2F) [3] protocols, this solution supports MFA on mobile platforms that use a diverse set of authenticators. The use of security assertion markup language (SAML) 2.0 [4] and OpenID Connect (OIDC) 1.0 [5] federation protocols allows PSOs to share identity assertions between applications and across PSO jurisdictions. Using this architecture allows PSFR personnel to authenticate once—say, at the beginning of their shift—and then leverage that single authentication to gain access to many other mobile native and web applications while on duty, reducing the time needed for authentication.

The PSFR community comprises tens of thousands of different organizations across the United States, many of which may operate their own IdPs. Today, most IdPs use SAML 2.0, but OIDC is rapidly gaining market share as an alternative for identity federation. As this build architecture demonstrates, an Open Authorization (OAuth) Authorization Server (AS) can integrate with both OIDC and SAML IdPs.

The guide provides:

- a detailed example solution and capabilities that may be implemented independently or in combination to address risk and security controls
- a demonstration of the approach using multiple, commercially available products
- how-to instructions for implementers and security engineers on integrating and configuring the example solution into their organization's enterprise in a manner that achieves security goals with minimum impact on operational efficiency and expense

Commercial, standards-based products, such as the ones that we used, are readily available and interoperable with existing IT infrastructure and investments.

This guide lists all of the necessary components and provides installation, configuration, and integration information so that a PSFR entity can replicate what we have built. The NCCoE does not particularly endorse the suite of commercial products used in our reference design. These products were used after an open call in the Federal Register to participate. Each organization's security experts should identify the standards-based products that will best integrate with its existing tools and IT system infrastructure. Organizations can adopt this solution or a different one that adheres to these guidelines in whole, or an organization can use this guide as a starting point for tailoring and implementing parts of a solution.

1.3 Benefits

The NCCoE, in collaboration with our stakeholders in the PSFR community, identified the need for a mobile SSO and MFA solution for native and web applications. This NCCoE practice guide, *Mobile Application Single Sign-On*, can help PSOs:

- define requirements for mobile application SSO and MFA implementation
- improve interoperability between mobile platforms, applications, and IdPs, regardless of the application development platform used in their construction
- enhance the efficiency of PSFRs by reducing the number of authentication steps, the time needed to get access to critical data, and the number of credentials that need to be managed
- support a diverse set of credentials, enabling PSOs to choose an authentication solution that best meets their individual needs
- enable cross-jurisdictional information sharing by identity federation

2 How to Use This Guide

This NIST Cybersecurity Practice Guide demonstrates a standards-based reference design and provides users with the information they need to replicate an MFA and mobile SSO solution for mobile native and web applications. This reference design is modular and can be deployed in whole or in parts.

This guide contains three volumes:

- NIST Special Publication (SP) 1800-13A: *Executive Summary*
- NIST SP 1800-13B: *Approach, Architecture, and Security Characteristics*–what we built and why **(you are here)**
- NIST SP 1800-13C: *How-To Guides*–instructions for building the example solution

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary (NIST SP 1800-13A)*, which describes the:

- challenges that enterprises face in MFA and mobile SSO for native and web applications
- example solution built at the NCCoE
- benefits of adopting the example solution

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in this part of the guide, *NIST SP 1800-13B,* which describes what we did and why. The following sections will be of particular interest:

- Section 3.5, Risk Assessment, provides a description of the risk analysis we performed
- Appendix A, Mapping to Cybersecurity Framework Core, maps the security characteristics of this example solution to cybersecurity standards and best practices

You might share the *Executive Summary, NIST SP 1800-13A,* with your leadership team members to help them understand the importance of adopting a standards-based MFA and mobile SSO solution for native and web applications.

IT professionals who want to implement an approach like this will find the whole practice guide useful. You can use the How-To portion of the guide, *NIST SP 1800-13C*, to replicate all or parts of the build created in our lab. The How-To guide provides specific product installation, configuration, and integration instructions for implementing the example solution. We do not recreate the product manufacturer's documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create an example solution.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, this guide does not endorse these particular products. Your organization can adopt this solution or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing SSO or MFA separately. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope you will seek products that are congruent with applicable standards and best practices. Section 3.7 lists the products we used and maps them to the cybersecurity controls provided by this reference solution.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but a possible solution. This is a draft guide. We seek feedback on its contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to psfr-nccoe@nist.gov.

2.1 Typographical Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	filenames and pathnames references to documents that are not hyperlinks, new terms, and placeholders	For detailed definitions of terms, see the *NCCoE Glossary.*
Bold	names of menus, options, command buttons and fields	Choose **File > Edit**.
`Monospace`	command-line input, on-screen computer output, sample code examples, status codes	`mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's National Cybersecurity Center of Excellence are available at http://nccoe.nist.gov

3 Approach

In conjunction with the PSFR community, the NCCoE developed a project description identifying MFA and SSO for mobile native and web applications as a critical need for PSFR organizations. The NCCoE then engaged subject matter experts from industry organizations, technology vendors, and standards bodies to develop an architecture and reference design leveraging new capabilities in modern mobile OSs and best current practices in SSO and MFA.

3.1 Audience

This guide is intended for individuals or entities who are interested in understanding the mobile native and web application SSO and MFA reference designs that the NCCoE has implemented to allow PSFR personnel to securely and efficiently gain access to mission-critical data by using mobile devices. Though the NCCoE developed this reference design with the PSFR community, any party interested in SSO and MFA for native mobile and web applications can leverage the architecture and design principles implemented in this guide.

The overall build architecture addresses three different audiences with somewhat separate concerns:

- IdPs – PSFR organizations that issue and maintain user accounts for their users. Larger PSFR organizations may operate their own IdP infrastructures and may federate using SAML or OIDC services, while others may seek to use an IdP service provider. IdPs are responsible for identity proofing, account creation, account and attribute management, and credential management.
- Relying parties (RPs) – organizations providing application services to multiple PSFR organizations. RPs may be software-as-a-service (SaaS) providers or PSFR organizations providing shared services consumed by other organizations. The RP operates an OAuth 2.0 AS, which integrates with users' IdPs and issues access tokens to enable mobile apps to make requests to the back-end application servers.
- App developers – mobile application developers. Today, mobile client apps are typically developed by the same software provider as the back-end RP applications. However, the OAuth framework enables interoperability between RP applications and third-party client apps. In any case, mobile application development is a specialized skill with unique considerations and requirements. Mobile application developers should consider implementing the AppAuth library for IETF RFC 8252 to enable standards-based SSO.

3.2 Scope

The focus of this project is to address the need for secure and efficient mobile native and web application SSO. The NCCoE drafted a use case that identified numerous desired solution characteristics. After an open call in the Federal Register for vendors to help develop a solution, we chose participating technology collaborators on a first-come, first-served basis. We scoped the project to produce the following high-level desired outcomes:

- provide a standards-based solution architecture that selects an effective and secure approach to implementing mobile SSO, leveraging native capabilities of the mobile OS
- ensure that mobile applications do not have access to user credentials

- support MFA and multiple authentication protocols
- support multiple authenticators, considering unique environmental constraints faced by first responders in emergency medical services, law enforcement, and fire services
- support cross-jurisdictional information sharing through the use of identity federation

To maintain the project's focus on core SSO and MFA requirements, the following subjects are out of scope. These technologies and practices are critical to a successful implementation, but they do not directly affect the core design decisions.

- Identity proofing – The solution will create synthetic digital identities that represent the identities and attributes of public safety personnel to test authentication assertions. This includes the usage of a lab-configured identity repository—not a genuine repository and schema provided by any PSO. This guide will not demonstrate an identity proofing process.
- Access control – This solution will support the creation and federation of attributes, but will not discuss or demonstrate access control policies that an RP might implement to govern access to specific resources.
- Credential storage – This solution will be agnostic to where credentials are stored on the mobile device. For example, this use case is not affected by storing a certificate in software versus hardware, such as a trusted platform module (TPM).
- Enterprise Mobility Management (EMM) – The solution will assume that all applications involved in the SSO experience are allowable via an EMM. This implementation may be supported by using an EMM (for example, to automatically provision required mobile apps to the device), but it does not strictly depend on using an EMM.
- Fallback authentication mechanisms – This solution involves the use of multifactor authenticators, which may consist of physical authentication devices or cryptographic keys stored directly on mobile devices. Situations may arise where a user's authenticator or device has been lost or stolen. This practice guide recommends registering multiple authenticators for each user as a partial mitigation, but, in some cases, it may be necessary to either enable users to fall back to single-factor authentication or provide other alternatives. Such fallback mechanisms must be evaluated considering the organization's security and availability requirements.

3.3 Assumptions

Before implementing the capabilities described in this practice guide, organizations should review the assumptions underlying the NCCoE build. These assumptions are detailed in Appendix B. Though not in scope for this effort, implementers should consider whether the same assumptions can be made based on current policy, process, and IT infrastructure. As detailed in Appendix B, applicable and appropriate guidance is provided to assist this process for the following functions:

- identity proofing

- mobile device security
- mobile application security
- EMM
- FIDO enrollment process

3.4 Business Case

Any decision to implement IT systems within an organization must begin with a solid business case. This business case could be an independent initiative or a component of the organization's strategic planning cycle. Individual business units or functional areas typically derive functional or business unit strategies from the overall organization's strategic plan. The business drivers for any IT project must originate in these strategic plans, and the decision to determine if an organization will invest in mobile SSO, identity federation, or MFA by implementing the solution in this practice guide will be based on the organization's decision-making process for initiating new projects.

An important set of inputs to the business case are the risks to the organization from mobile authentication and identity management, as outlined in Section 3.5. Apart from addressing cybersecurity risks, SSO also improves the user experience and alleviates the overhead associated with maintaining and using passwords for multiple applications. This provides a degree of convenience to all types of users, but reducing the authentication overhead for PSFR users, and reducing barriers to getting the information and applications that they need, could have a tremendous effect. First responder organizations and application providers also benefit by using interoperable standards that provide easy integration across disparate technology platforms. In addition, the burden of account management is reduced by using a single user account managed by the organization to access multiple applications and services.

3.5 Risk Assessment

NIST SP 800-30 [6], *Guide for Conducting Risk Assessments,* states, "Risk is the net negative impact of the exercise of a vulnerability, considering both the probability and the impact of occurrence. Risk management is the process of identifying risk, assessing risk, and taking steps to reduce risk to an acceptable level." The NCCoE recommends that any discussion of risk management, particularly at the enterprise level, begins with a comprehensive review of NIST 800-37, *Guide for Applying the Risk Management Framework to Federal Information Systems* [7], material that is available to the public. The risk management framework guidance as a whole proved invaluable in giving us a baseline to assess risks, from which we developed the project, the security characteristics of the build, and this guide.

3.5.1 PSFR Risks

As PSFR communities adopt mobile platforms and applications, organizations should consider potential risks that these new devices and ecosystems introduce that may negatively affect PSFR organizations and the ability of PSFR personnel to operate. These risks include, but are not limited to, the following risks:

- The reliance on passwords alone by many PSFR entities has the effect of expanding the scope of a single application/database compromise when users fall back to reusing a small set of easily remembered passwords across multiple applications.
- Complex passwords are harder to remember and input into IT systems. Mobile devices exacerbate this issue with small screens, touchscreens that may not work with gloves or other PSFR equipment, and three separate keyboards among which the user must switch. In an emergency response, any delay in accessing information may prove critical to containing a situation.
- Social engineering, man-in-the-middle attacks, replay attacks, and phishing all present real threats to password-based authentication systems.
- Deterministic, cryptographic authentication mechanisms have security benefits, yet come with the challenge of cryptographic key management. Loss or misuse of cryptographic keys could undermine an authentication system, leading to unauthorized access or data leakage.
- Biometric authentication mechanisms may be optimal for some PSFR personnel, yet organizations need to ensure that PII, such as fingerprint templates, is protected.
- Credentials exposed to mobile apps could be stolen by malicious apps or misused by non-malicious apps. Previously, it was common for native apps to use embedded user agents (commonly implemented with web views) for OAuth requests. That approach has many drawbacks, including the host app being able to copy user credentials and cookies, as well as the user needing to authenticate again in each app.

3.5.2 Mobile Ecosystem Threats

Any discussion of risks and vulnerabilities is incomplete without considering the threats that are involved. NIST SP 800-150, *Guide to Cyber Threat Information Sharing* [8], states:

> *A cyber threat is "any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service."*

To simplify this concept, a *threat* is anything that can exploit a vulnerability to damage an asset. Finding the intersection of these three will yield a *risk*. Understanding the applicable threats to a system is the first step to determining its risks.

However, identifying and delving into mobile threats is not the primary goal of this practice guide. Instead, we rely on prior work from NIST's Mobile Threat Catalogue (MTC), along with its associated NIST Interagency Report (NISTIR) 8144, *Assessing Threats to Mobile Devices & Infrastructure* [9]. Each entry in the MTC contains several pieces of information: an identifier, a category, a high-level description, details on its origin, exploit examples, examples of common vulnerabilities and exposures (CVE), possible countermeasures, and academic references. For the purposes of this practice guide, we are primarily interested in threat identifiers, categories, descriptions, and countermeasures.

In broad strokes, the MTC covers 32 threat categories that are grouped into 12 distinct classes, as shown in Table 3-1. Of these categories, three in particular, highlighted in green in the table, are covered by the guidance in this practice guide. If implemented correctly, this guidance will help mitigate those threats.

Table 3-1 Threat Classes and Categories

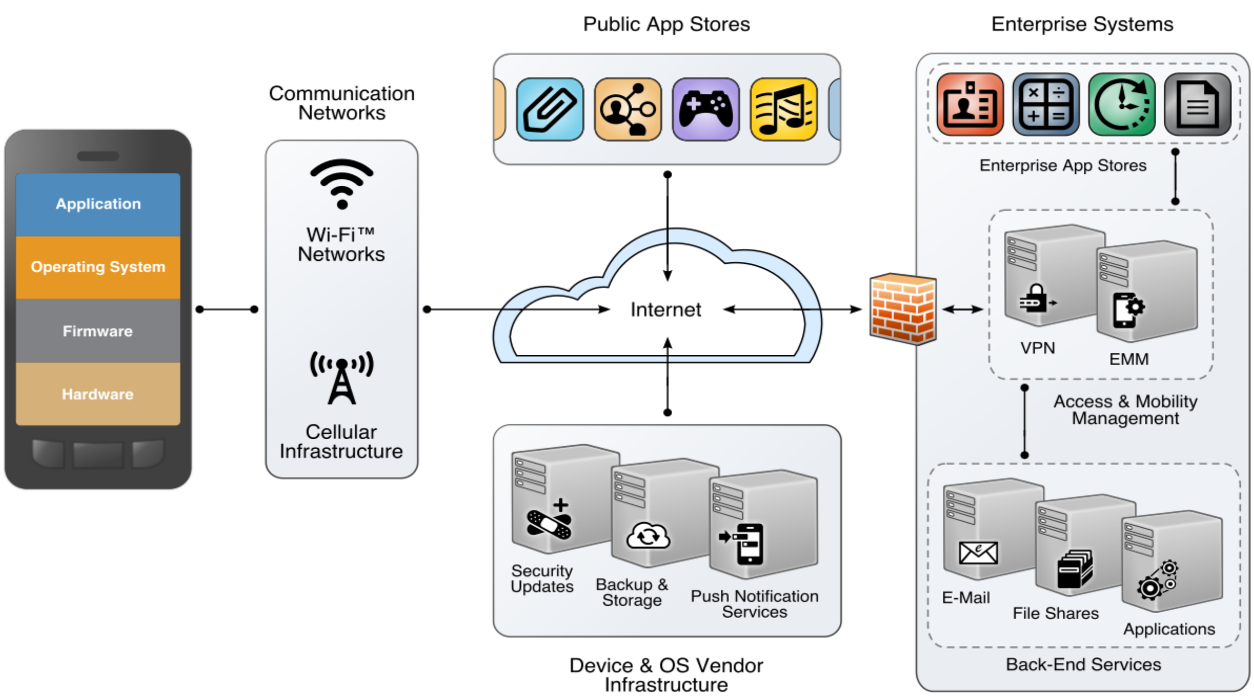
Threat Class	Threat Category
Application	Malicious or Privacy-Invasive Application
	Vulnerable Applications
Authentication	Authentication: User or Device to Network
	Authentication: User or Device to Remote Service
	Authentication: User to Device
Cellular	Carrier Infrastructure
	Carrier Interoperability
	Cellular Air Interface
	Consumer-Grade Femtocell
	SMS / MMS / RCS
	USSD
	VoLTE
Ecosystem	Mobile Application Store
	Mobile OS & Vendor Infrastructure

Threat Class	Threat Category
Local Area Network (LAN) and Personal Area Network (PAN)	Network Threats: Bluetooth
	Network Threats: Near Field Communication (NFC)
	Network Threats: Wi-Fi
Payment	Application-Based
	In-App Purchases
	NFC-Based
Physical Access	Physical Access
Privacy	Behavior Tracking
Supply Chain	Supply Chain
Stack	Baseband Subsystem
	Boot Firmware
	Device Drivers
	Isolated Execution Environments
	Mobile OS

Threat Class	Threat Category	Threat Class	Threat Category
EMM	Enterprise Mobility Management		SD Card
Global Positioning System (GPS)	GPS		USIM / SIM / UICC Security

The other categories, while still important elements of the mobile ecosystem and critical to the health of an overall mobility architecture, are out of scope for this document. The entire mobile ecosystem should be considered when analyzing threats to the architecture; this ecosystem is depicted in Figure 3-1, taken from NISTIR 8144. Each player in the ecosystem—the mobile device user, the enterprise, the network operator, the app developer, and the original equipment manufacturer (OEM)—can find suggestions to deter other threats by reviewing the MTC and NISTIR 8144. Many of these share common solutions, such as using EMM software to monitor device health, and installing apps only from authorized sources.

Figure 3-1 The Mobile Ecosystem



3.5.3 Authentication and Federation Threats

The MTC is a useful reference from the perspective of mobile devices, applications, and networks. In the context of mobile SSO, specific threats to authentication and federation systems must also be considered. Table 8-1 in NIST SP 800-63B [10] lists several categories of threats against authenticators:

- theft–stealing a physical authenticator, such as a smart card or U2F device
- duplication–unauthorized copying of an authenticator, such as a password or private key
- eavesdropping–interception of an authenticator secret when in use
- offline cracking–attacks on authenticators that do not require interactive authentication attempts, such as brute-force attacks on passwords used to protect cryptographic keys
- side channel attack–exposure of an authentication secret through observation of the authenticator’s physical characteristics
- phishing or pharming–capturing authenticator output through impersonation of the RP or IdP
- social engineering–using a pretext to convince the user to subvert the authentication process
- online guessing–attempting to guess passwords through repeated online authentication attempts with the RP or IdP
- endpoint compromise–malicious code on the user’s device, which is stealing authenticator secrets, redirecting authentication attempts to unintended RPs, or otherwise subverting the authentication process
- unauthorized binding–binding an attacker-controlled authenticator with the user’s account by intercepting the authenticator during provisioning or impersonating the user in the enrollment process

These threats undermine the basic assumption that use of an authenticator in an authentication protocol demonstrates that the user initiating the protocol is the individual referenced by the claimed user identifier. Mitigating these threats is the primary design goal of MFA, and the FIDO specifications address many of these threats.

An additional set of threats concerns federation protocols. Authentication threats affect the process of direct authentication of the user to the RP or IdP, whereas federation threats affect the assurance that the IdP can deliver assertions that are genuine and unaltered, only to the intended RP. Table 8-1 in NIST SP 800-63C [11] lists the following federation threats:

- assertion manufacture or modification–generation of a false assertion or unauthorized modification of a valid assertion
- assertion disclosure–disclosure of sensitive information contained in an assertion to an unauthorized third party
- assertion repudiation by the IdP–IdP denies having authenticated a user after the fact

- assertion repudiation by the subscriber–subscriber denies having authenticated and performed actions on the system
- assertion redirect–subversion of the federation protocol flow to enable an attacker to obtain the assertion or to redirect it to an unintended RP
- assertion reuse–attacker obtains a previously used assertion to establish his own session with the RP
- assertion substitution–attacker substitutes an assertion for a different user in the federation flow, leading to session hijacking or fixation

Federation protocols are complex and require interaction among multiple systems, typically under different management. Implementers should carefully apply best security practices relevant to the federation protocols in use. Most federation protocols can incorporate security measures to address these threats, but this may require specific configuration and enabling optional features.

3.6 Systems Engineering

Some organizations use a systems engineering–based approach to plan and implement their IT projects. Organizations wishing to implement IT systems should conduct robust requirements development, taking into consideration the operational needs of each system stakeholder. Standards such as International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) 15288:2015, *Systems and software engineering–System life cycle processes* [12], and NIST SP 800-160, *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems* [13], provide guidance for applying security in systems development. With both standards, organizations can choose to adopt only those sections of the standard that are relevant to their development approach, environment, and business context. NIST SP 800-160 recommends a thorough analysis of alternative solution classes accounting for security objectives, considerations, concerns, limitations, and constraints. This advice applies to both new system developments and integration of components into existing systems, the focus of this practice guide. Section 4.1, General Architecture Considerations, may assist organizations with this analysis.

3.7 Technologies

Table 3-2 lists all technologies used in this project, and provides a mapping among the generic application term, the specific product used, and the NIST Cybersecurity Framework (CSF) subcategory that the product provides. For a mapping of CSF subcategories to security controls, please refer to Appendix A, Mapping to Cybersecurity Framework Core. Refer to Table A-1 for an explanation of the CSF category and subcategory codes.

Table 3-2 Products and Technologies

Component	Specific Product Used	How the Component Functions in the Build	Applicable CSF Subcategories
Federation Server	Ping Federate 8.2	OAuth 2.0 AS OIDC provider SAML 2 IdP	PR.AC-3: Remote access is managed
FIDO U2F Server	StrongAuth StrongKey Crypto Engine (SKCE) 2.0	FIDO U2F server	PR.AC-1: Identities and credentials are managed for authorized devices and users
External Authenticator	YubiKey Neo	FIDO U2F token supporting authentication over NFC	PR-AC-1: Identities and credentials are managed for authorized devices and users
FIDO UAF Server	Nok Nok Labs FIDO UAF Server	UAF authenticator enrollment, authentication, and transaction confirmation	PR.AC-1: Identities and credentials are managed for authorized devices and users
Mobile Applications (including SaaS back end)	Motorola Solutions Public Safety Experience (PSX) Cockpit, PSX Messenger, and PSX Mapping 5.2	Provide application programming interfaces (APIs) for mobile client apps to access cloud-hosted services and data; consume OAuth tokens	PR.AC-3: Remote access is managed
SSO Implementing Best Current Practice	AppAuth Software Development Kit (SDK)	Library used by mobile apps, providing an IETF RFC 8252-compliant OAuth 2.0 client implementation; implements authorization requests, Proof Key for Code Exchange (PKCE), and token refresh	PR.AC-3: Remote access is managed

4 Architecture

The NCCoE worked with industry subject matter experts to develop an open, standards-based, commercially available architecture demonstrating three main capabilities:

- SSO to RP applications using OAuth 2.0 implemented in accordance with RFC 8252 (the *OAuth 2.0 for Native Apps* BCP)
- Identity federation to RP applications using both SAML 2.0 and OIDC 1.0
- MFA to mobile native and web applications using FIDO UAF and U2F

Though these capabilities are implemented as an integrated solution in this guide, organizational requirements may dictate that only a subset of these capabilities be implemented. The modular approach of this architecture is designed to support such use cases.

Additionally, the authors of this document recognize that PSFR organizations will have diverse IT infrastructures, which may include previously purchased authentication, federation, or SSO capabilities, and legacy technology. For this reason, Section 4.1 and Appendix C outline general considerations that any organization may apply when designing an architecture tailored to organizational needs. Section 4.2 follows with considerations for implementing the architecture specifically developed by the NCCoE for this project.

Organizations are encouraged to read Section 3.2, Section 3.3, Section 3.5, and Appendix B to provide context for this architecture design.

4.1 General Architectural Considerations

The PSFR community is large and diverse, comprising numerous state, local, tribal, and federal organizations with individual missions and jurisdictions. PSFR personnel include police, firefighters, emergency medical technicians, public health officials, and other skilled support personnel. There is no single management or administrative hierarchy spanning the PSFR population. PSFR organizations operate in a variety of environments with different technology requirements and wide variations in IT staffing and budgets.

Cooperation and communication among PSFR organizations at multiple levels is crucial to addressing emergencies that span organizational boundaries. Examples include coordination among multiple services within a city (e.g., fire and police services), among different state law enforcement agencies to address interstate crime, and among federal agencies like the Department of Homeland Security (DHS) and its state and local counterparts. This coordination is generally achieved through peer-to-peer interaction and agreement or through federation structures, such as the National Identity Exchange Federation (NIEF). Where interoperability is achieved, it is the result of the cooperation of willing partners, rather than adherence to central mandates.

Enabling interoperability across the heterogeneous, decentralized PSFR user base requires a standards-based solution; a proprietary solution might not be uniformly adopted and could not be mandated. The solution must also support identity federation and federated authentication, as user accounts and authenticators are managed by several different organizations. The solution must also accommodate organizations of different sizes, levels of technical capabilities, and budgets. Compatibility with the existing capabilities of fielded identity systems can reduce the barrier to entry for smaller organizations.

Emergency response and other specialized work performed by PSFR personnel often require that they wear personal protective equipment, such as gloves, masks, respirators, and helmets. This equipment renders some authentication methods impractical or unusable. Fingerprint scanners cannot be used with gloves, authentication using a mobile device camera to analyze the user's face or iris may be hampered by masks or goggles, and entering complex passwords on small virtual keyboards is also impractical for gloved users. In addition, PSFR work often involves urgent and hazardous situations requiring the ability to quickly perform mission activities like driving, firefighting, and administering urgent medical aid. Therefore, the solution must support a variety of authenticators in an interoperable way so that individual user groups can select authenticators suited to their operational constraints.

In considering these requirements, the NCCoE implemented a standards-based architecture and reference design. Section 4.1.1 through Section 4.1.3 detail the primary standards used, while Appendix C goes into great depth on architectural consideration when implementing these standards.

4.1.1 SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries

SSO enables a user to authenticate once and to subsequently access different applications without having to authenticate again. SSO on mobile devices is complicated by the sandboxed architecture, which makes it difficult to share the session state with back-end systems between individual apps. EMM vendors have provided solutions through proprietary SDKs, but this approach requires integrating the SDK with each individual app and does not scale to a large and diverse population, such as the PSFR user community.

OAuth 2.0 is an IETF standard that has been widely adopted to provide delegated authorization of clients accessing representational state transfer (REST) interfaces, including mobile applications. OAuth 2.0, when implemented in accordance with RFC 8252 (the *OAuth 2.0 for Native Apps* BCP), provides a standards-based SSO pattern for mobile apps. The OpenID Foundation's AppAuth libraries [14] can facilitate building mobile apps in full compliance with IETF RFC 8252, but any mobile app that follows RFC 8252's core recommendation of using a shared external user-agent for the OAuth authorization flow will have the benefit of SSO. OAuth considerations and recommendations are detailed in Section C.1 of Appendix C.

4.1.2 Identity Federation

SAML 2.0 [4] and OIDC 1.0 [5] are two standards that enable an application to redirect users to an IdP for authentication and to receive an assertion of the user's identity and other optional attributes. Federation is important in a distributed environment like the PSFR community, where user management occurs in numerous local organizations. Federated authentication relieves users of having to create accounts in each application that they need to access, and frees application owners from managing user accounts and credentials. OIDC is a more recent protocol, but many organizations have existing SAML deployments. The architecture supports both standards to facilitate adoption without requiring upgrades or modifications to existing SAML IdPs. Federation considerations and recommendations are detailed in Section C.2 of Appendix C.

4.1.3 FIDO and Authenticator Types

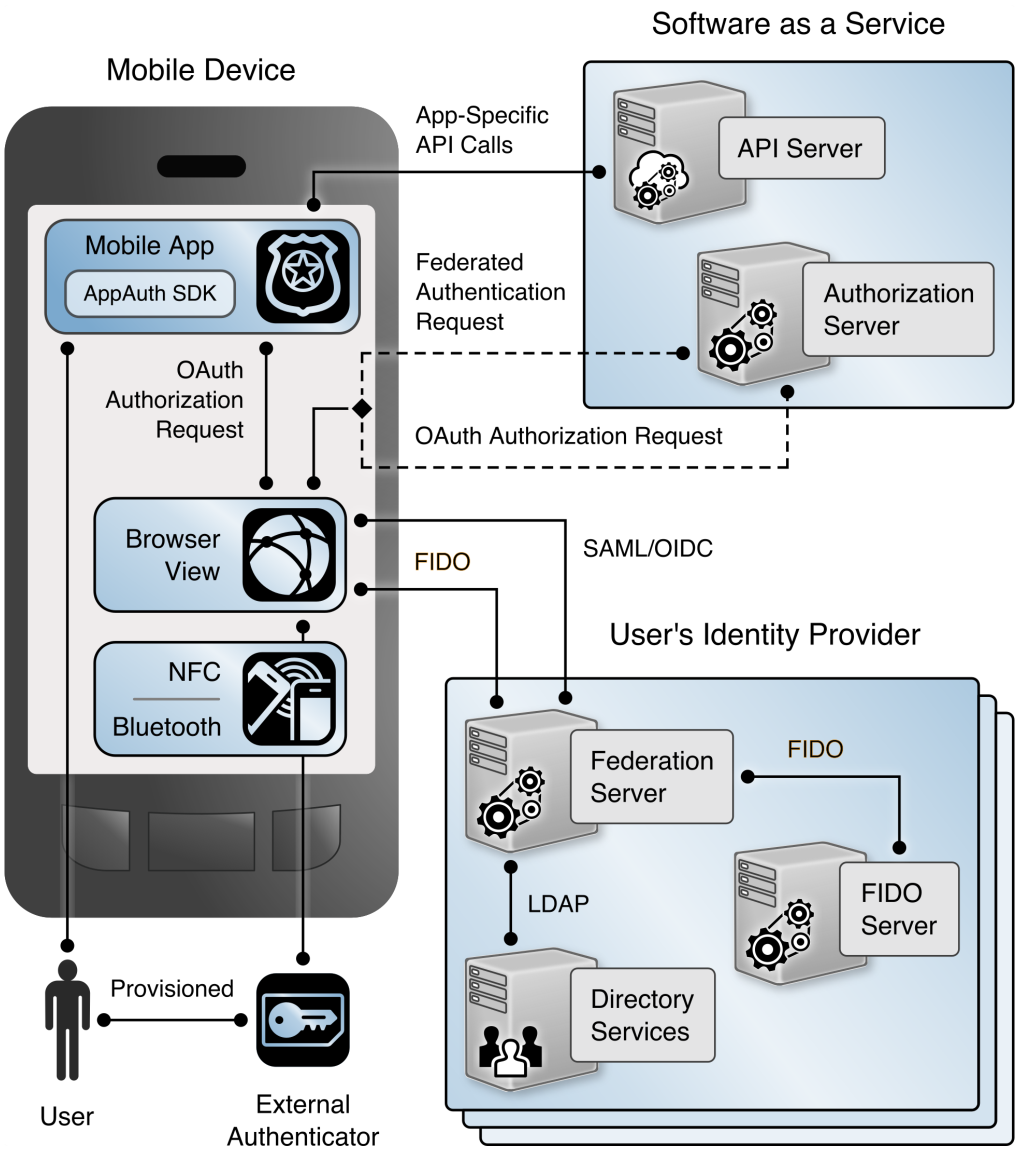
When considering MFA implementations, PSFR organizations should carefully consider organizationally defined authenticator requirements. These requirements are detailed in Section C.3 of Appendix C.

FIDO provides a standard framework within which vendors have produced a wide range of interoperable biometric, hardware, and software authenticators. This will enable PSFR organizations to choose authenticators suitable to their operational constraints. The FIDO Alliance has published specifications for two types of authenticators based on UAF and U2F. These protocols operate agnostic of the FIDO authenticator, allowing PSOs to choose any FIDO-certified authenticator that meets operational requirements and to implement it with this solution. The protocols, FIDO key registration, FIDO authenticator attestation, and FIDO deployment considerations are also detailed in Section C.3 of Appendix C.

4.2 High-Level Architecture

The NCCoE implemented both FIDO UAF and U2F for this project. The high-level architecture varies somewhat between the two implementations. Figure 4-1 depicts the interactions between the key elements of the build architecture with the U2F implementation.

568 **Figure 4-1 High-Level U2F Architecture**

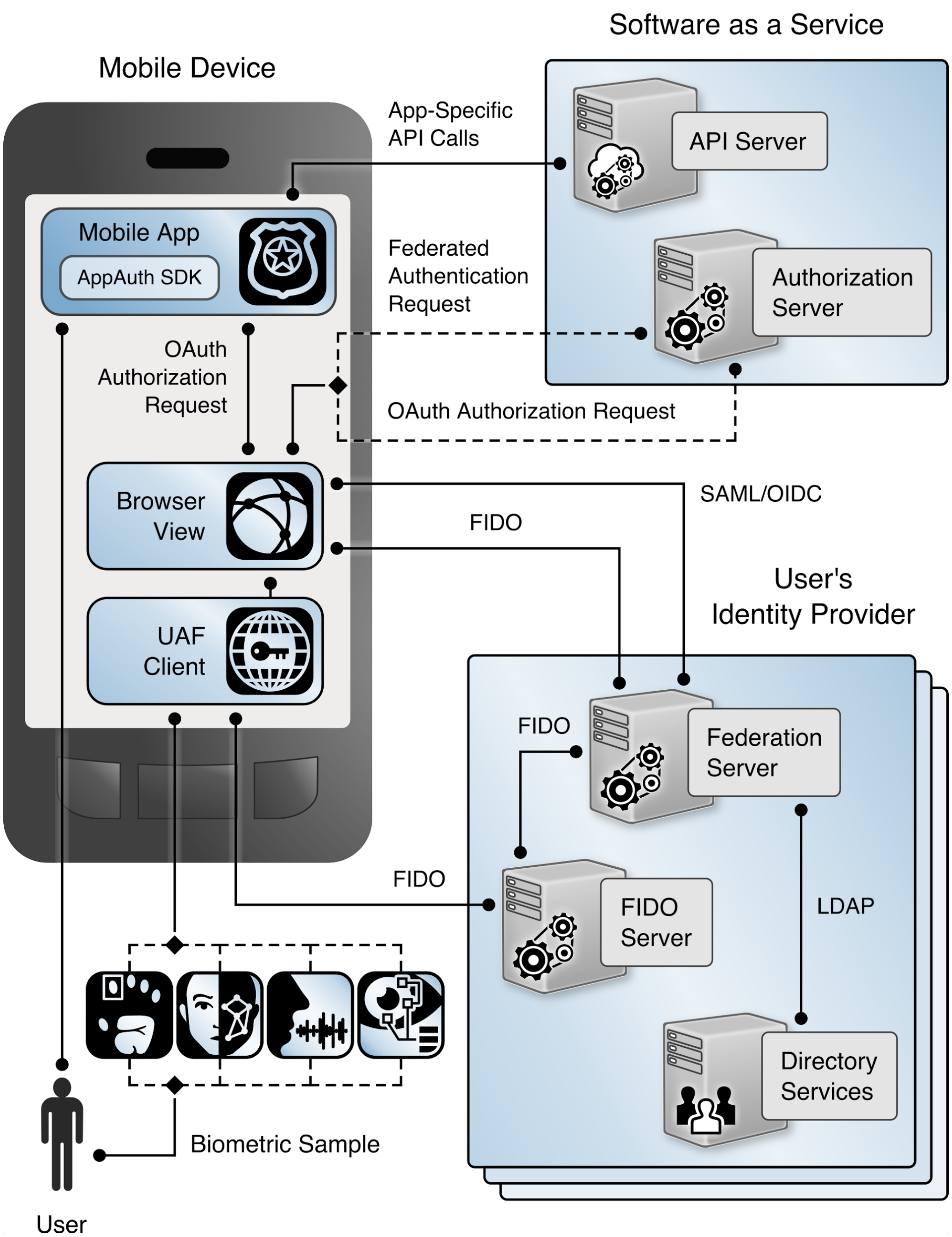


569

570 On the mobile device, the mobile app includes the OpenID Foundation's AppAuth library, which
571 streamlines implementation of the OAuth client functionality in accordance with the IETF RFC 8252,
572 *OAuth 2.0 for Native Apps*, guidance. AppAuth orchestrates the authorization request flow by using the
573 device's native browser capabilities, including the use of in-app browser tabs on devices that support
574 them. The mobile device also supports the two FIDO authentication schemes, UAF and U2F. UAF
575 typically involves an internal (on-device) authenticator that authenticates the user directly to the device
576 by using biometrics, other hardware capabilities, or a software client. U2F typically involves an external
577 hardware authenticator token, which communicates with the device over NFC or Bluetooth.

Figure 4-2 shows the corresponding architecture view with the FIDO UAF components.

Figure 4-2 High-Level UAF Architecture



The SaaS provider hosts application servers that provide APIs consumed by mobile apps, as well as an OAuth AS. The browser on the mobile device connects to the AS to initiate the OAuth authorization code flow. The AS redirects the browser to the user's organization's IdP to authenticate the user. Once the user has authenticated, the AS will issue an access token, which is returned to the mobile app through a browser redirect and can be used to authorize requests to the application servers.

The user's IdP includes a federation server that implements SAML or OIDC, directory services containing user accounts and attributes, and a FIDO authentication service that can issue authentication challenges and validate the responses that are returned from FIDO authenticators. The FIDO authentication service may be built into the IdP, but is more commonly provided by a separate server.

A SaaS provider may provide multiple apps, which may be protected by the same AS. For example, Motorola Solutions provides both the PSX Mapping and PSX Messaging applications, which are protected by a shared AS. Users may also use services from different SaaS providers, which would have separate ASs. This build architecture can provide SSO between apps hosted by a single SaaS provider, as well as across apps provided by multiple SaaS vendors.

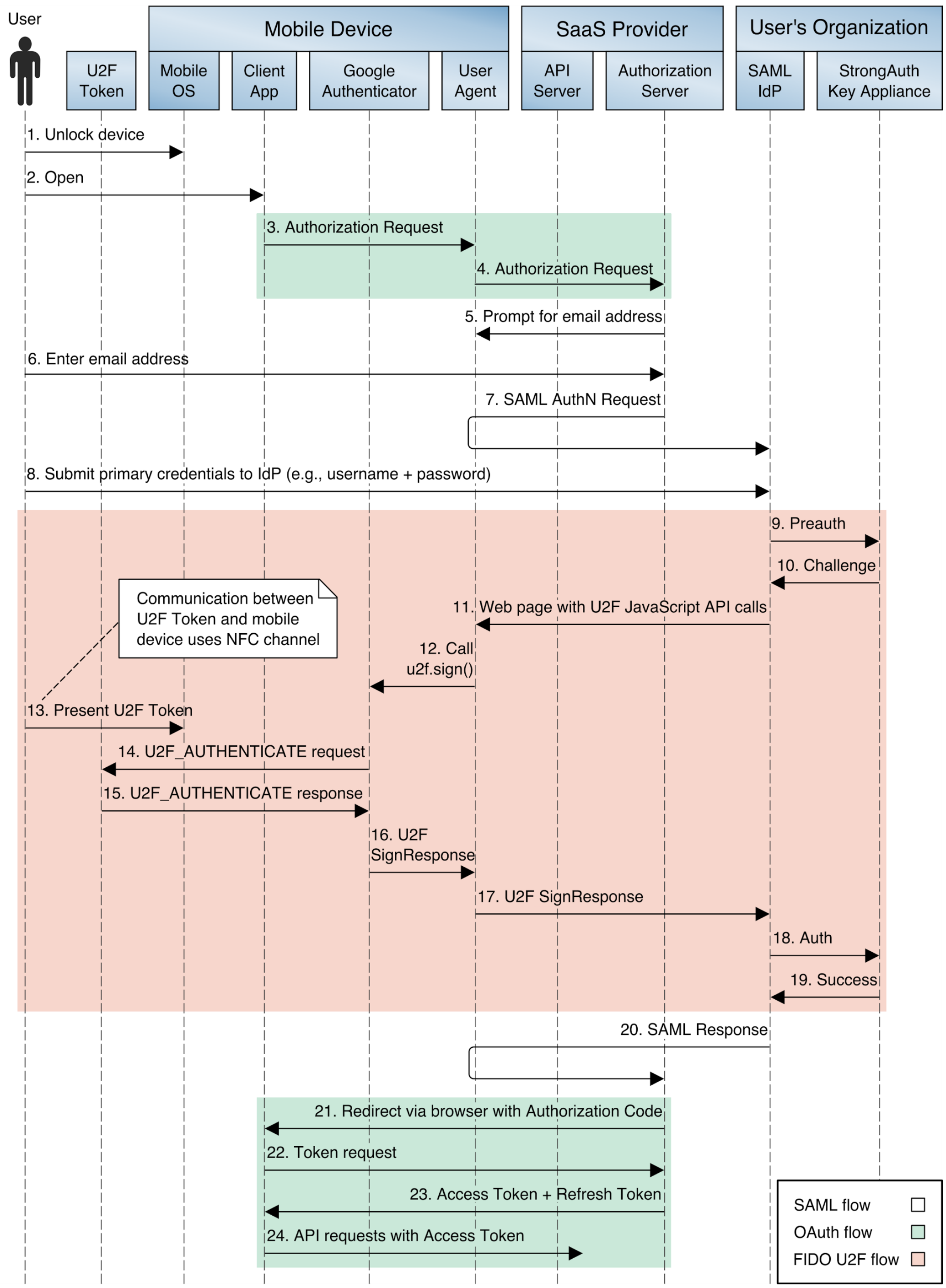
4.3 Detailed Architecture Flow

The mobile SSO lab implementation demonstrates two authentication flows: one in which the user authenticates to a SAML IdP with a YubiKey Neo U2F token and a PIN, and one in which the user authenticates to an OIDC IdP by using UAF with a fingerprint. These pairings of federation and authentication protocols are purely arbitrary; U2F could just as easily be used with OIDC, for example.

4.3.1 SAML and U2F Authentication Flow

The authentication flow using SAML and U2F is depicted in Figure 4-3. This figure depicts the message flows among different components on the mobile device or hosted by the SaaS provider or user organization. In the figure, colored backgrounds differentiate the SAML, OAuth, and FIDO U2F protocol flows. Prior to this authentication flow, the user must have registered a FIDO U2F token with the IdP, and the AS and IdP must have exchanged metadata and established an RP trust.

Figure 4-3 SAML and U2F Sequence Diagram



The detailed steps are as follows:

1. The user unlocks the mobile device. Any form of lock-screen authentication can be used; it is not directly tied to the subsequent authentication or authorization.
2. The user opens a mobile app that connects to the SaaS provider's back-end services. The mobile app determines that an OAuth token is needed. This may occur because the app has no access or refresh tokens cached, it has an existing token known to be expired based on token metadata, or it may submit a request to the API server with a cached bearer token and receive an HTTP 401 status code in the response.
3. The mobile app initiates an OAuth authorization request using the authorization code flow by invoking an in-app browser tab with the Uniform Resource Locator (URL) of the SaaS provider AS's authorization endpoint.
4. The in-app browser tab submits the request to the AS over an Hypertext Transfer Protocol Secure (HTTPS) connection. This begins the OAuth 2 authorization flow.
5. The AS returns a page that prompts for the user's email address.
6. The user submits the email address. The AS uses the domain of the email address for IdP discovery. The user needs to specify the email address only one time; the address is stored in a cookie in the device browser and will be used to automatically determine the user's IdP on subsequent visits to the AS.
7. The AS redirects the device browser to the user's IdP with a SAML authentication request. This begins the SAML authentication flow.
8. The IdP returns a login page. The user submits a username and PIN. The IdP validates these credentials against the directory service. If the credentials are invalid, the IdP redirects back to the login page with an error message and prompts the user to authenticate again. If the credentials are valid, the IdP continues to Step 9.
9. The IdP submits a "preauth" API request to the StrongAuth SKCE server. The preauth request includes the authenticated username obtained in Step 8. This begins the FIDO U2F authentication process.
10. The SKCE responds with a U2F challenge that must be signed by the user's registered key in the U2F token to complete authentication. If the user has multiple keys registered, the SKCE returns a challenge for each key so that the user can authenticate with any registered authenticator.

11. The IdP returns a page to the user's browser that includes Google's JavaScript U2F API and the challenge obtained from the SKCE in Step 10. The user taps a button on the page to initiate U2F authentication, which triggers a call to the u2f.sign JavaScript function.
12. The u2f.sign function invokes the Google Authenticator app, passing it the challenge, the appId (typically the domain name of the IdP), and an array of the user's registered key.
13. Google Authenticator prompts the user to hold the U2F token against the NFC radio of the mobile device, which the user does.
14. Google Authenticator connects to the U2F token over the NFC channel and sends an applet selection command to activate the U2F applet on the token. Google Authenticator then submits a U2F_AUTHENTICATE message to the token.
15. Provided that the token has one of the keys registered at the IdP, it signs the challenge and returns the signature in an authentication success response over the NFC channel.
16. Google Authenticator returns the signature to the browser in a SignResponse object.
17. The callback script on the authentication web page returns the SignResponse object to the IdP.
18. The IdP calls the "authenticate" API on the SKCE, passing the SignResponse as a parameter.
19. The SKCE validates the signature of the challenge by using the registered public key, and verifies that the appId matches the IdP's and that the response was received within the configured timeout. The API returns a response to the IdP, indicating success or failure, and any error messages. This concludes the U2F authentication process; the user has now authenticated to the IdP, which sets a session cookie.
20. The IdP returns a SAML response indicating the authentication success or failure to the AS through a browser redirect. If authentication has succeeded, the response will include the user's identifier and, optionally, additional attribute assertions. This concludes the SAML authentication flow. The user is now authenticated to the AS, which sets a session cookie. Optionally, the AS could prompt the user to approve the authorization request, displaying the scopes of access being requested at this step.
21. The AS sends a redirect to the browser with the authorization code. The target of the redirect is the mobile app's redirect_uri, a link that opens in the mobile app through a mechanism provided by the mobile OS (e.g., custom request scheme or Android AppLink).
22. The mobile app extracts the authorization code from the URL and submits it to the AS's token endpoint.

23. The AS responds with an access token, and, optionally, a refresh token that can be used to obtain an additional access token when the original token expires. This concludes the OAuth authorization flow.

24. The mobile app can now submit API requests to the SaaS provider's back-end services by using the access token in accordance with the bearer token authorization scheme defined in RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage* [15].

4.3.2 OpenID Connect and UAF Authentication Flow

The authentication flow involving OIDC and UAF is depicted in Figure 4-4.

Figure 4-4 OIDC and UAF Sequence Diagram

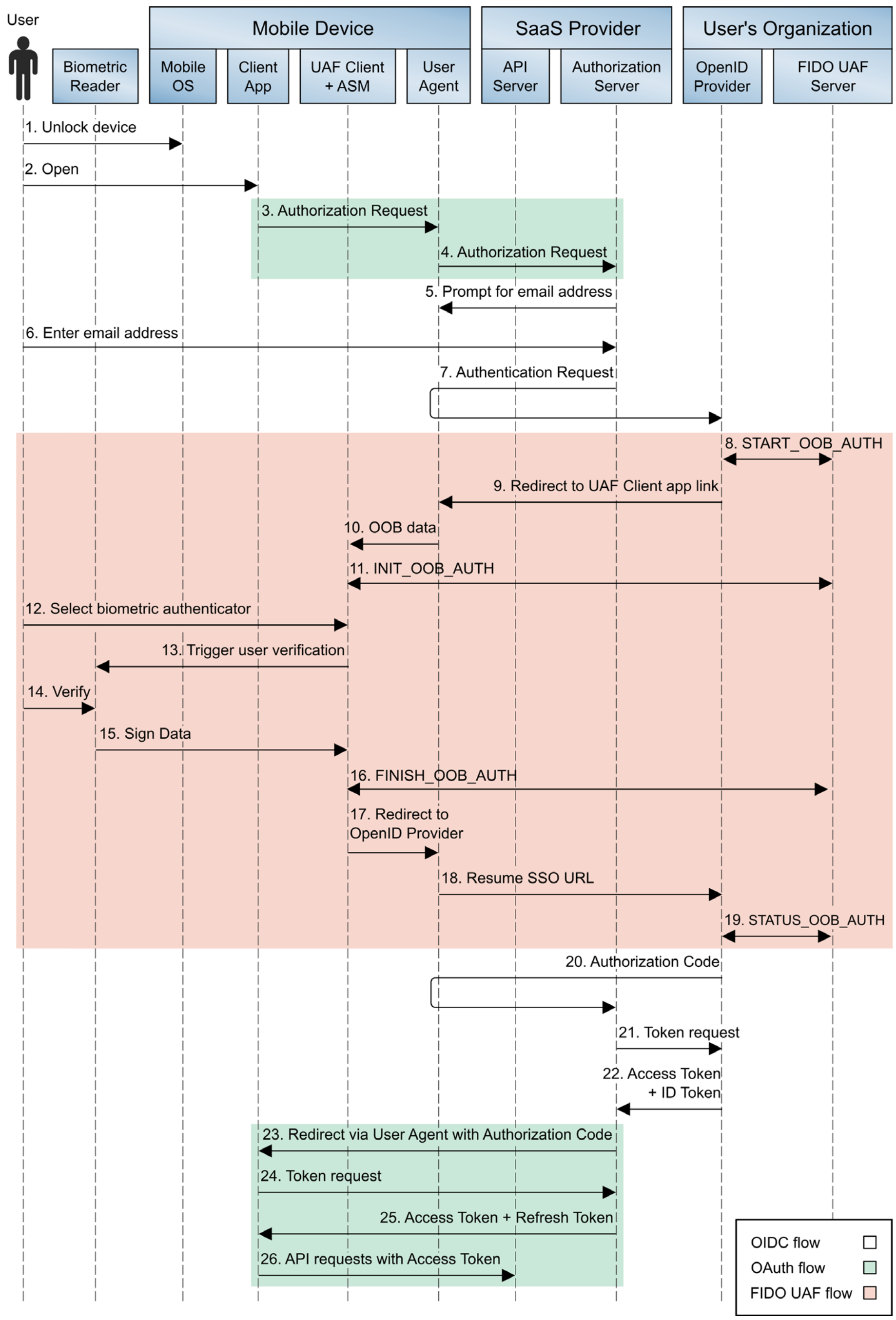


Figure 4-4 uses the same conventions and color coding as the earlier SAML/U2F diagram (Figure 4-3) to depict components on the device, at the SaaS provider and at the user's organization. Prior to this authentication flow, the user must have registered a FIDO UAF authenticator with the IdP, and the AS must be registered as an OIDC client at the IdP. The detailed steps are listed below. For ease of comparison, steps that are identical to the corresponding step in Figure 4-3 are shown in italics.

1. *The user unlocks the mobile device. Any form of lock-screen authentication can be used; it is not directly tied to the subsequent authentication or authorization.*
2. *The user opens a mobile app that connects to the SaaS provider's back-end services. The mobile app determines that an OAuth token is needed. This may occur because the app has no access or refresh tokens cached, it has an existing token known to be expired based on token metadata, or it may submit a request to the API server with a cached bearer token and receive an HTTP 401 status code in the response.*
3. *The mobile app initiates an OAuth authorization request using the authorization code flow by invoking an in-app browser tab with the URL of the SaaS provider AS's authorization endpoint.*
4. *The in-app browser tab submits the request to the AS over an HTTPS connection. This begins the OAuth 2 authorization flow.*
5. *The AS returns a page that prompts for the user's email address.*
6. *The user submits the email address. The AS uses the domain of the email address for IdP discovery. The user needs to specify the email address only one time; the address is stored in a cookie in the device browser and will be used to automatically determine the user's IdP on subsequent visits to the AS.*
7. The AS redirects the device browser to the user's IdP with an OIDC authentication request. This begins the OIDC authentication flow.
8. The IdP submits a START_OOB_AUTH request to the UAF authentication server. The server responds with a data structure containing the necessary information for a UAF client to initiate an out-of-band (OOB) authentication, including a transaction identifier linked to the user's session at the IdP.
9. The IdP returns an HTTP redirect to the in-app browser tab. The redirect target URL is an app link that will pass the OOB data to the Nok Nok Labs Passport application on the device.
10. The Nok Nok Passport app opens and extracts the OOB data from the app link URL.
11. Passport sends an INIT_OOB_AUTH request to the UAF authentication server, including the OOB data and a list of authenticators available on the device that the user has registered for use at the IdP. The server responds with a set of UAF challenges for the registered authenticators.

12. If the user has multiple registered authenticators (e.g., fingerprint and voice authentication), Passport prompts the user to select which authenticator to use.

13. Passport activates the authenticator, which prompts the user to perform the required steps for verification. For example, if the selected authenticator is the Android Fingerprint authenticator, the standard Android fingerprint user interface (UI) overlay will pop over the browser and prompt the user to scan an enrolled fingerprint. The authenticator UI may be presented by Passport (for example, the PIN authenticator), or it may be provided by an OS component.

14. The user completes the biometric scan or other user verification activity. Verification occurs locally on the device; biometrics and secrets are not transmitted to the server.

15. The authenticator signs the UAF challenge by using the private key that was created during initial UAF enrollment with the IdP. The authenticator returns control to the Passport application through an app link with the signed UAF challenge.

16. The Passport app sends a FINISH_OOB_AUTH API request to the UAF authentication server. The server extracts the username and registered public key and validates the signed response. The server can also validate the authenticator's attestation signature and check that the security properties of the authenticator satisfy the IdP's security policy. The server caches the authentication result.

17. The Passport app closes, returning control to the in-app browser tab, which is redirected to the "resume SSO" URL at the IdP. This URL is defined on the Ping server to enable multistep authentication flows and allow the browser to be redirected back to the IdP after completing required authentication steps with another application.

18. The in-app browser tab requests the Resume SSO URL at the IdP.

19. The IdP sends a STATUS_OOB_AUTH API request to the UAF authentication server. The UAF server responds with the success/failure status of the out-of-band authentication, and any associated error messages. (Note: The IdP begins sending STATUS_OOB_AUTH requests periodically, following Step 9 in the flow, and continues to do so until a final status is returned or the transaction times out.) This concludes the UAF authentication process; the user has now authenticated to the IdP, which sets a session cookie.

20. The IdP returns an authorization code to the AS through a browser redirect.

21. The AS submits a token request to the IdP's token endpoint, authenticating with its credentials and including the authorization code.

22. The IdP responds with an identification (ID) token and an access token. The ID token includes the user's identifier and, optionally, additional attribute assertions. The access token can option-

ally be used to request additional user claims at the IdP's user information endpoint. This concludes the OIDC authentication flow. The user is now authenticated to the AS, which sets a session cookie. Optionally, the AS could prompt for the user to approve the authorization request, displaying the scopes of access being requested at this step.

23. *The AS sends a redirect to the browser with the authorization code. The target of the redirect is the mobile app's redirect_uri, a link that opens in the mobile app through a mechanism provided by the mobile OS (e.g., custom request scheme or Android AppLink).*

24. *The mobile app extracts the authorization code from the URL and submits it to the AS's token endpoint.*

25. *The AS responds with an access token, and, optionally, a refresh token that can be used to obtain an additional access token when the original token expires. This concludes the OAuth authorization flow.*

26. *The mobile app can now submit API requests to the SaaS provider's back-end services by using the access token in accordance with the bearer token authorization scheme.*

Both authentication flows end with a single app obtaining an access token to access back-end resources. At this point, traditional OAuth token life cycle management would begin. Access tokens have an expiration time. Depending on the application's security policy, refresh tokens may be issued along with the access token and used to obtain a new access token when the initial token expires. Refresh tokens and access tokens can continue to be issued in this manner for as long as the security policy allows. When the current access token has expired and no additional refresh tokens are available, the mobile app would submit a new authorization request to the AS.

Apart from obtaining an access token, the user has established sessions with the AS and IdP that can be used for SSO.

4.4 Single Sign-On with the OAuth Authorization Flow

When multiple apps invoke a common user agent to perform the OAuth authorization flow, the user agent maintains the session state with the AS and IdP. In the build architecture, this can enable SSO in two scenarios.

In the first case, assume that a user has launched a mobile application, has been redirected to an IdP to authenticate, and has completed the OAuth flow to obtain an access token. Later, the user launches a second app that connects to the same AS used by the first app. The app will initiate an authorization request, using the same user-agent as the first app. Provided that the user has not logged out at the AS, this request will be sent with the session cookie that was established when the user authenticated in the previous authorization flow. The AS will recognize the user's active session and issue an access token to the second app, without requiring the user to authenticate again.

In the second case, again assume that the user has completed an OAuth flow, including authentication to an IdP, while launching the first app. Later, the user launches a second app that connects to a different AS from the first app. Again, the second app initiates an authorization request, using the same user-agent as the first app. The user has no active session with the second AS, so the user-agent is redirected to the IdP to obtain an authentication assertion. Provided that the user has not logged out at the IdP, the authentication request will include the previously established session cookie, and the user will not be required to authenticate again at the IdP. The IdP will return an assertion to the AS, which will then issue an access token to the second app.

This architecture can also provide SSO across native and web applications. If the web app is an RP to the same SAML or OIDC IdP used in the authentication flow described above, the app will redirect the browser to the IdP and resume the user's existing session, without the need to reauthenticate, provided that the browser used to access the web app is the same one used in the authorization flow described above. For example, if a Google Chrome Custom Tab is used in the native app OAuth flow, then accessing the web app in Chrome will provide a shared cookie store and SSO. If the web app uses the OAuth 2.0 implicit grant, then SSO could follow either of the above workflows, depending on whether the user is already authenticated at the AS used by the app.

When apps use embedded web views, instead of the system browser or in-app tabs for the OAuth authorization flow, each individual app's web view has its own cookie store, so there is no continuity of the session state as the user transitions from one app to another, and the user must authenticate each time.

4.5 App Developer Perspective of the Build

The following paragraphs provide takeaways from an application developer's perspective regarding the experience of the build team, inclusive of FIDO, the AppAuth library, PKCE, and Chrome Custom Tabs.

AppAuth was integrated as described in Section C.1 of Appendix C. From an application developer perspective, the primary emphasis in the build was integrating AppAuth. The authentication technology was basically transparent to the developer. In fact, the native application developers for this project had no visibility to the FIDO U2F or UAF integration. This transparency was achieved through the AppAuth pattern of delegating the authentication process to the in-app browser tab capability of the OS. Other application developer effects are listed below:

- There are several pieces of information that must be supplied by an application in the OAuth Authorization Request, such as the scope and the client ID, which an OAuth AS might use to apply appropriate authentication policy. These details are obtained during the OAuth client registration process with the AS.
- The ability to support multiple IdPs, without requiring any hard-coding of IdP URLs in the app itself, was achieved by using Hypertext Markup Language (HTML) forms hosted by the IdP to

collect information from end users (e.g., domain) during login, which was used to perform IdP discovery.

4.6 Identity Provider Perspective of the Build

The IdP is responsible for account and attribute creation and maintenance, as well as credential provisioning, management, and de-provisioning. Some IdP concerns for this architecture are listed below:

- Enrollment/registration of authenticators. IdPs should consider the enrollment process and life cycle management for MFA. For this NCCoE project, FIDO UAF enrollment was launched by the user via tapping a native enrollment application (Nok Nok Labs' Passport app). During user authentication, the same application (Passport) was invoked programmatically (via AppLink) to perform FIDO authentication. In a production implementation, the IdP would need to put processes in place to enroll, retire, or replace authenticators when needed. A process for responding when authenticators are lost or stolen is particularly important to prevent unauthorized access.
- For UAF: A FIDO UAF client must be installed (e.g., we installed Nok Nok Labs' NNL Passport). When utilizing AppLink, a script must be written in the IdP adapter to request user permission to follow the AppLink (invoke FIDO UAF client).
- For U2F: Download and install Google Authenticator (or equivalent) because mobile browsers do not support FIDO U2F 1.1 natively (as do some desktop browsers).

4.7 Token and Session Management

The RP application owners have two separate areas of concern when it comes to token and session management. They have the authorization tokens to manage on the client side, and the identity tokens/sessions to receive and manage from the IdP side. Each of these functions has its own separate concerns and requirements.

When dealing with the native app's access to the RP application data, the RP operators need to make sure that appropriate authorization is in place. The architecture in Section 4.2 uses OAuth 2.0 and authorization tokens for this purpose, following the guidance from IETF RFC 8252. Native app clients present a special challenge, as mentioned earlier, especially when it comes to protecting the authorization code being returned to the client. To mitigate a code interception threat, RFC 8252 requires that both clients and servers use PKCE for public native-app clients. ASs should reject authorization requests from native apps that do not use PKCE. The lifetime of the authorization tokens depends on the use case, but the general recommendation from the OAuth working group is to use short-lived access tokens and long-lived refresh tokens. The reauthentication requirements in NIST SP 800-63B [10] can be used as guidance for maximum refresh token lifetimes at each authenticator assurance level (AAL). All security considerations from RFC 8252 apply here as well, such as making sure that attackers cannot easily guess any of the token values or credentials.

The RP may directly authenticate the user, in which case all of the current best practices for web session security and protecting the channel with Transport Layer Security (TLS) apply. However, if there is delegated or federated authentication via a third-party IdP, then the RP must also consider the implications for managing the identity claims received from the IdP, whether it be an ID token from an OIDC provider or a SAML assertion from a SAML IdP. This channel is used for authentication of the user, which means that potential PII may be obtained. Care must be taken to obtain user consent prior to authorization for the release and use of this information in accordance with relevant regulations. If OIDC is used for authentication to the RP, then all of the OAuth 2.0 security applies again here. In all cases, all channels between parties must be protected with TLS encryption.

5 Security Characteristics Analysis

The purpose of the security characteristic evaluation is to understand the extent to which the project meets its objective of demonstrating MFA and mobile SSO for native and web applications. In addition, it seeks to document the security benefits and drawbacks of the example solution.

5.1 Assumptions and Limitations

This security characteristics analysis is focused on the specific design elements of the build, consisting of MFA, SSO, and federation implementation. It discusses some elements of application development, but only the aspects that directly interact with the SSO implementation. It does not focus on potential underlying vulnerabilities in OSs, application run times, hardware, or general secure coding practices. It is assumed that risks to these foundational components are managed separately (e.g., through asset and patch management). As with any implementation, all layers of the architecture must be appropriately secured, and it is assumed that implementers will adopt standard security and maintenance practices to the elements not specifically addressed here.

This project did not include a comprehensive test of all security components or “red team” penetration testing or adversarial emulation. Cybersecurity is a rapidly evolving field where new threats and vulnerabilities are continually discovered. Therefore, this security guidance cannot be guaranteed to identify every potential weakness of the build architecture. It is assumed that implementers will follow risk management procedures as outlined in the NIST Risk Management Framework.

5.2 Threat Analysis

The following subsections describe how the build architecture addresses the threats discussed in Section 3.5.

5.2.1 Mobile Ecosystem Threat Analysis

In Section 3.5.1, we introduced the MTC, described the 32 categories of mobile threats that it covers, and highlighted the three categories that this practice guide addresses: Vulnerable Applications, Authentication: User or Device to Network, and Authentication: User or Device to Remote Service.

At the time of this writing, these categories encompass 18 entries in the MTC. However, the MTC is a living catalogue, which is continually being updated. Instead of addressing each threat, we describe, in general, how these types of threats are mitigated by the architecture laid out in this practice guide:

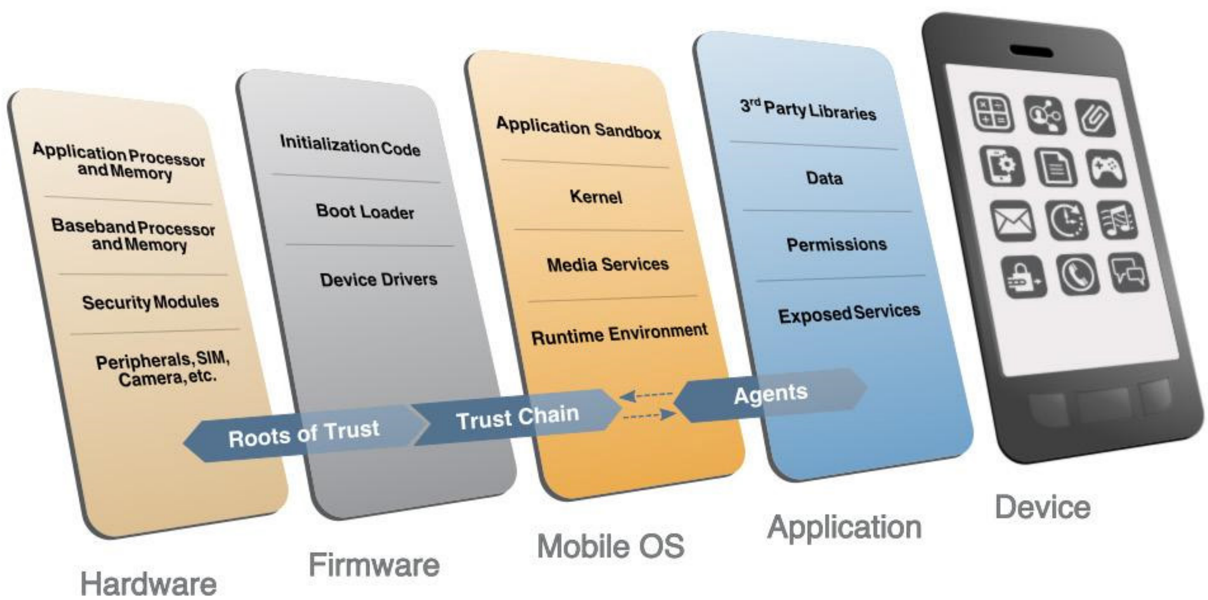
- Use encryption for data in transit: The IdP and AS enforce HTTPS encryption by default, which the app is required to use during SSO authentication.
- Use newer mobile platforms: Volume C of this guide *(NIST SP 1800-13C)* calls for using at least Android 5.0 or iOS 8.0 or newer, which mitigates weaknesses of older versions (e.g., apps can access the system log in Android 4.0 and older).
- Use built-in browser features: The AppAuth for Android library utilizes the Chrome Custom Tabs feature, which activates the device's native browser; this allows the app to leverage built-in browser features, such as identifying and avoiding known malicious web pages. Similar functionality exists on iOS devices using the SFSafariViewController and SFAuthenticationSession APIs.
- Avoid hard-coded secrets: The AppAuth guidance recommends and supports the use of PKCE; this allows developers to avoid using a hard-coded OAuth client secret.
- Avoid logging sensitive data: The AppAuth library, which handles the OAuth 2 flow, does not log any sensitive data.
- Use sound authentication practices: By using SSO, the procedures outlined in this guide allow app developers to rely on the IdP's implementation of authentication practices, such as minimum length and complexity requirements for passwords, maximum authentication attempts, and periodic reset requirements; in addition, the IdP can introduce new authenticators without any downstream effect to applications.
- Use sound token management practices: Again, this guide allows app developers to rely on the IdP's implementation of authorization tokens and good management practices, such as replay-resistance mechanisms and token expirations.
- Use two-factor authentication: Both FIDO U2F and UAF, as deployed in this build architecture, provide multifactor cryptographic user authentication. The U2F implementation requires the user to authenticate with a password or PIN and with a single-factor cryptographic token,

whereas the UAF implementation utilizes a key pair stored in the device's hardware-backed key store that is unlocked through user verification consisting of a biometric (e.g., fingerprint or voice match) or a password or PIN.

- Protect cryptographic keys: FIDO U2F and UAF authentication leverage public key cryptography. In this architecture, U2F private keys are stored external to the mobile device in a hardware-secure element on a YubiKey Neo. UAF private keys are stored on the Android device's hardware-backed key store. These private keys are never sent to external servers.
- Protect biometric templates: When using biometric authentication mechanisms, organizations should consider the storage and use of user biometric templates. This architecture relies on the native biometric mechanisms implemented by modern mobile devices and OSs, which verify biometrics templates locally and store them in protected storage.

To fully address these threats and threats in other MTC categories, additional measures should be taken by all parties involved in the mobile ecosystem: the mobile device user, the enterprise, the network operator, the app developer, and the OEM. A figure depicting this ecosystem in total is shown in Section 3.5.1. In addition, the mobile platform stack should be understood in great detail to fully assess the threats that may be applicable. An illustration of this stack, taken from NISTIR 8144 [9], is shown in Figure 5-1.

Figure 5-1 Mobile Device Technology Stack



Several tools, techniques, and best practices are available to mitigate these other threats. EMM software can allow enterprises to manage devices more fully and to gain a better understanding of device health; one example of this is detecting whether a device has been *rooted* or *jailbroken*, which compromises the security architecture of the entire platform. Application security-vetting software (commonly known as app-vetting software) can be utilized to detect vulnerabilities in first-party apps and to discover potentially malicious behavior in third-party apps. When used in conjunction with EMM software to limit which apps can be installed on a device, this can greatly lessen the attack surface of the platform. For more guidance on these threats and mitigations, refer to the MTC and NISTIR 8144 [9].

5.2.2 Authentication and Federation Threat Analysis

Section 3.5.3 discussed threats specific to authentication and federation systems, which are catalogued in NIST SP 800-63-3 [16]. MFA, provided in the build architecture by FIDO U2F and UAF, is designed to mitigate several authentication risks:

- Theft of physical authenticator – Possessing an authenticator, which could be a YubiKey (in the case of U2F) or the mobile device itself (in the case of UAF), does not, in itself, enable an attacker to impersonate the user to an RP or IdP. Additional knowledge or a biometric factor is needed to authenticate.
- Eavesdropping – Some MFA solutions, including many one-time password (OTP) implementations, are vulnerable to eavesdropping attacks. FIDO implements cryptographic authentication, which does not involve the transmission of secrets over the network.
- Social engineering – A typical social engineering exploit involves impersonating a system administrator or other authority figure under some pretext to convince users to disclose their passwords over the phone, but this comprises only a single authentication factor.
- Online guessing – Traditional password authentication schemes may be vulnerable to online guessing attacks, though lockout and throttling policies can reduce the risk. Cryptographic authentication schemes are not vulnerable to online guessing.

FIDO also incorporates protections against phishing and pharming attacks. When a FIDO authenticator is registered with an RP, a new key pair is created and associated with the RP's app ID, which is derived from the domain name in the URL where the registration transaction was initiated. During authentication, the app ID is again derived from the URL of the page that is requesting authentication, and the authenticator will sign the authentication challenge only if a key pair has been registered with the matching app ID. The FIDO facets specification enables sites to define a list of domain names that should be treated as a single app ID, to accommodate service providers that span multiple domain names, such as google.com and gmail.com.

The app ID verification effectively prevents the most common type of phishing attack, in which the attacker creates a new domain and tricks users into visiting that domain, instead of an intended RP where the user has an account. For example, an attacker might register a domain called "google-accts.com" and send emails with a pretext to get users to visit the site, such as a warning that the user's account will be disabled unless some action is taken. The attacker's site would present a login screen identical to Google's login screen, to obtain the user's password (and OTP, if enabled) credentials and to use them to impersonate the user to the real Google services. With FIDO, the authenticator would not have an existing key pair registered under the attacker’s domain, so the user would be unable to return a signed FIDO challenge to the attacker’s site. If the attacker could convince the user to register the FIDO authenticator with the malicious site and then sign an authentication challenge, the signed FIDO assertion could not be used to authenticate to Google, because the RP can also verify the app ID associated with the signed challenge, and it would not be the expected ID.

A more advanced credential theft attack involves an active man-in-the-middle who can intercept the user’s requests to the legitimate RP and act as a proxy between the two. To avoid TLS server certificate validation errors, in this case, the attacker must obtain a TLS certificate for the legitimate RP site that is trusted by the user’s device. This could be accomplished by exploiting a vulnerability in a commercial certificate authority (CA); it presents a high bar for the attacker, but is not unprecedented. App ID validation is not sufficient to prevent this attacker from obtaining an authentication challenge from the RP, proxying it to the user, and using the signed assertion that it gets back from the user to authenticate to the RP. To prevent this type of attack, the FIDO specifications permit the use of token binding to protect the signed assertion that is returned to the RP by including information in the assertion about the TLS channel over which it is being delivered. If there is a man-in-the-middle (or a proxy of any kind) between the user and the RP, the RP can detect it by examining the token binding message included in the assertion and comparing it to the TLS channel over which it was received. Token binding is not universally implemented today, but, as the specification nears final publication, adoption is expected to increase.

Many of the federation threats discussed in Section 3.5.3 can be addressed by signing assertions, ensuring their integrity and authenticity. Encrypted assertions can also provide multiple protections, preventing disclosure of sensitive information contained in the assertion, and providing a strong protection against assertion redirection because only the intended RP will have the key required to decrypt the assertion. Most mitigations to federation threats require the application of protocol-specific guidance for SAML and OIDC. These considerations are not specific to the mobile SSO use case; the application of a security-focused profile of these protocols can mitigate many potential issues.

In addition to RFC 8252, application developers and RP service providers should consult the *OAuth 2.0 Threat Model and Security Considerations* documented in RFC 6819 [17] for best practices for implementing OAuth 2.0. The AppAuth library supports a secure OAuth client implementation by automatically handling details like PKCE. Key protections for OAuth and OIDC include those listed below:

- Requiring HTTPS for protocol requests and responses protects access tokens and authorization codes and authenticates the server to the client.
- Using in-app browser tabs for the authentication flow, in conformance with RFC 8252, protects user credentials from exposure to the mobile client app or the application service provider.

- OAuth tokens are associated with access scopes, which can be used to limit the authorizations granted to any given client app, which somewhat mitigates the potential for misuse of compromised access tokens.
- PKCE, as explained previously, prevents interception of the authorization code by malicious apps on the mobile device.

5.3 Scenarios and Findings

The overall test scenario involved launching the Motorola Solutions PSX Cockpit mobile app, authenticating, and then subsequently launching additional PSX apps and validating that the apps could access the back-end APIs and reflected the identity of the authenticated user. To enable testing of the two different authentication scenarios, two separate "user organization" infrastructures were created in the NCCoE lab, and both were registered as IdPs to the test PingFederate instance acting as the PSX AS. A "domain selector" was created in PingFederate to perform IdP discovery based on the domain of the user's email address, enabling the user to trigger authentication at one of the IdPs.

Prior to testing the authentication infrastructure, users had to register U2F and UAF authenticators at the respective IdPs. FIDO authenticator registration requires a process that provides high assurance that the authenticator is in the possession of the claimed account holder. In practice, this typically requires a strongly authenticated session or an in-person registration process overseen by an administrator. In the lab, a notional enrollment process was implemented with the understanding that real-world processes would be different and subject to agency security policies. Organizations should refer to NIST SP 800-63B [10] for specific considerations regarding credential enrollment. From a FIDO perspective, however, the registration data used would be the same.

Lab testing showed that the build architecture consistently provided SSO between applications. Two operational findings were uncovered during testing:

- Knowing the location of the NFC radio on the mobile device greatly improves the user experience when authenticating with an NFC token, such as the YubiKey Neo. The team found that NFC radios are in different locations on different devices; on the Nexus 6P, for example, the NFC radio is near the top of the device, near the camera, whereas, on the Galaxy S6 Edge, the NFC radio is slightly below the vertical midpoint of the device. After initial experimentation to locate the radio, team members could quickly and reliably make a good NFC connection with the YubiKey by holding it in the correct location. Device manufacturers provide NFC radio location information via device technical specifications.
- Time synchronization between servers is critical. In lab testing, intermittent authentication errors were found to be caused by clock drift between the IdP and the AS. This manifested as the AS reporting JavaScript object notation (JSON) Web Token (JWT) validation errors when attempting to validate ID tokens received from the IdP. All participants in the federation scheme should synchronize their clocks to a reliable network time protocol (NTP) source, such as the

NIST NTP pools [18]. Implementations should allow for a small amount of clock skew—on the order of a few seconds—to account for the unpredictable latency of network traffic.

6 Future Build Considerations

6.1 Single Logout

To ensure that only authorized personnel get access to application resources, users must be logged out from application sessions when access is no longer needed or when a session expires. In an SSO scenario, a user may need to be logged out from one or many applications at a given time. This scenario will demonstrate architectures for tearing down user sessions, clearly communicating to the user which application(s) have active sessions, and ensuring that active sessions are not orphaned.

6.2 Shared Devices

This scenario will focus on a situation where two or more colleagues share a single mobile device to accomplish a mission. The credentials, such as the FIDO UAF and U2F used in this guide, will be included, but may need to be registered to multiple devices. This scenario will explore situations in which multiple profiles or no profiles are installed on a device, potentially requiring the user to log out prior to giving the device to another user.

6.3 Step-Up Authentication

A user will access applications by using an acceptable, but low, assurance authenticator. Upon requesting access to an application that requires higher assurance, the user will be prompted for an additional authentication factor. Determinations on whether to step up may be based on risk-relevant data points collected by the IdP at the time of authentication, referred to as the authentication context.

Appendix A Mapping to Cybersecurity Framework Core

Table A-1 maps informative National Institute of Standards and Technology (NIST) and consensus security references to the Cybersecurity Framework (CSF) Core subcategories that are addressed by NIST Special Publication (SP) 1800-13. The references do not include protocol specifications that are implemented by the individual products that compose the demonstrated security platforms. While some of the references provide general guidance that informs implementation of referenced CSF Core Functions, the NIST SP 1800-13 references provide specific recommendations that should be considered when composing and configuring security platforms and technologies described in this practice guide.

Table A-1 CSF Categories

Category	Subcategory	Informative References
Asset Management (ID.AM): The data, personnel, devices, systems, and facilities that enable the organization to achieve business purposes are identified and managed consistent with their relative importance to business objectives and the organization's risk strategy	**ID.AM-1:** Physical devices and systems within the organization are inventoried	**CCS CSC** 1 **COBIT 5** BAI09.01, BAI09.02 **ISA 62443-2-1:2009** 4.2.3.4 **ISA 62443-3-3:2013** SR 7.8 **ISO/IEC 27001:2013** A.8.1.1, A.8.1.2 **NIST SP 800-53 Rev. 4** CM-8
Access Control (PR.AC): Access to assets and associated facilities is limited to authorized users, processes, or devices, and to authorized activities and transactions	**PR.AC-1:** Identities and credentials are managed for authorized devices and users	**CCS CSC** 16 **COBIT 5** DSS05.04, DSS06.03 **ISA 62443-2-1:2009** 4.3.3.5.1 **ISA 62443-3-3:2013** SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.7, SR 1.8, SR 1.9 **ISO/IEC 27001:2013** A.9.2.1, A.9.2.2, A.9.2.4, A.9.3.1, A.9.4.2, A.9.4.3 **NIST SP 800-53 Rev. 4** AC-2, Information Assurance (IA) Family

Category	Subcategory	Informative References
	PR.AC-3: Remote access is managed	**COBIT 5** APO13.01, DSS01.04, DSS05.03 **ISA 62443-2-1:2009** 4.3.3.6.6 **ISA 62443-3-3:2013** SR 1.13, SR 2.6 **ISO/IEC 27001:2013** A.6.2.2, A.13.1.1, A.13.2.1 **NIST SP 800-53 Rev. 4** AC-17, AC-19, AC-20
	PR.AC-4: Access permissions are managed, incorporating the principles of least privilege and separation of duties	**CCS CSC** 12, 15 **ISA 62443-2-1:2009** 4.3.3.7.3 **ISA 62443-3-3:2013** SR 2.1 **ISO/IEC 27001:2013** A.6.1.2, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4 **NIST SP 800-53 Rev. 4** AC-2, AC-3, AC-5, AC-6, AC-16
Data Security (PR.DS): Information and records (data) are managed consistent with the organization's risk strategy to protect the confidentiality, integrity, and availability of information	**PR.DS-5:** Protections against data leaks are implemented	**CCS CSC** 17 **COBIT 5** APO01.06 **ISA 62443-3-3:2013** SR 5.2 **ISO/IEC 27001:2013** A.6.1.2, A.7.1.1, A.7.1.2, A.7.3.1, A.8.2.2, A.8.2.3, A.9.1.1, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4, A.9.4.5, A.13.1.3, A.13.2.1, A.13.2.3, A.13.2.4, A.14.1.2, A.14.1.3 **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4

Category	Subcategory	Informative References
Protective Technology (PR.PT): Technical security solutions are managed to ensure the security and resilience of systems and assets, consistent with related policies, procedures, and agreements	**PR.PT-1:** Audit/log records are determined, documented, implemented, and reviewed in accordance with policy	**CCS CSC** 14 **COBIT 5** APO11.04 **ISA 62443-2-1:2009** 4.3.3.3.9, 4.3.3.5.8, 4.3.4.4.7, 4.4.2.1, 4.4.2.2, 4.4.2.4 **ISA 62443-3-3:2013** SR 2.8, SR 2.9, SR 2.10, SR 2.11, SR 2.12 **ISO/IEC 27001:2013** A.12.4.1, A.12.4.2, A.12.4.3, A.12.4.4, A.12.7.1 **NIST SP 800-53 Rev. 4** AU Family
	PR.PT-2: Removable media is protected, and its use restricted according to policy	**COBIT 5** DSS05.02, APO13.01 **ISA 62443-3-3:2013** SR 2.3 **ISO/IEC 27001:2013** A.8.2.2, A.8.2.3, A.8.3.1, A.8.3.3, A.11.2.9 **NIST SP 800-53 Rev. 4** MP-2, MP-4, MP-5, MP-7
	PR.PT-3: Access to systems and assets is controlled, incorporating the principle of least functionality	**COBIT 5** DSS05.02 **ISA 62443-2-1:2009** 4.3.3.5.1, 4.3.3.5.2, 4.3.3.5.3, 4.3.3.5.4, 4.3.3.5.5, 4.3.3.5.6, 4.3.3.5.7, 4.3.3.5.8, 4.3.3.6.1, 4.3.3.6.2, 4.3.3.6.3, 4.3.3.6.4, 4.3.3.6.5, 4.3.3.6.6, 4.3.3.6.7, 4.3.3.6.8, 4.3.3.6.9, 4.3.3.7.1, 4.3.3.7.2, 4.3.3.7.3, 4.3.3.7.4 **ISA 62443-3-3:2013** SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.6, SR 1.7, SR 1.8, SR 1.9, SR 1.10, SR 1.11, SR 1.12, SR 1.13, SR 2.1, SR 2.2, SR 2.3, SR 2.4, SR 2.5, SR 2.6, SR 2.7 **ISO/IEC 27001:2013** A.9.1.2 **NIST SP 800-53 Rev. 4** AC-3, CM-7

Category	Subcategory	Informative References
	PR.PT-4: Communications and control networks are protected	**CCS CSC** 7 **COBIT 5** DSS05.02, APO13.01 **ISA 62443-3-3:2013** SR 3.1, SR 3.5, SR 3.8, SR 4.1, SR 4.3, SR 5.1, SR 5.2, SR 5.3, SR 7.1, SR 7.6 **ISO/IEC 27001:2013** A.13.1.1, A.13.2.1 **NIST SP 800-53 Rev. 4** AC-4, AC-17, AC-18, CP-8, SC-7

Appendix B Assumptions Underlying the Build

This project is guided by the following assumptions. Implementers are advised to consider whether the same assumptions can be made based on current policy, process, and information-technology (IT) infrastructure. Where applicable, appropriate guidance is provided to assist this process as described in the following subsections.

B.1 Identity Proofing

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-63A, *Enrollment and Identity Proofing* [19], addresses how applicants can prove their identities and become enrolled as valid subjects within an identity system. It provides requirements for processes by which applicants can both proof and enroll at one of three different levels of risk mitigation, in both remote and physically present scenarios. NIST SP 800-63A contains both normative and informative material. Organizations should use NIST SP 800-63A to develop and implement an identity proofing plan within their enterprise.

B.2 Mobile Device Security

Mobile devices can add to an organization's productivity by providing employees with access to business resources at any time. Not only has this reshaped how traditional tasks are accomplished, but organizations are also devising entirely new ways to work. However, mobile devices may be lost or stolen. A compromised mobile device may allow remote access to sensitive on-premises organizational data or any other data that the user has entrusted to the device. Several methods exist to address these concerns (e.g., using a device lock screen, setting shorter screen timeouts, forcing a device wipe in case of too many failed authentication attempts). It is up to the organization to implement these types of security controls, which can be enforced with Enterprise Mobility Management (EMM) software (see Section B.4).

NIST SP 1800-4, *Mobile Device Security: Cloud & Hybrid Builds* [20], demonstrates how to secure sensitive enterprise data that is accessed by and/or stored on employees' mobile devices. The NIST *Mobile Threat Catalogue* [21] identifies threats to mobile devices and associated mobile infrastructure to support the development and implementation of mobile security capabilities, best practices, and security solutions to better protect enterprise IT. We strongly encourage organizations implementing this practice guide in whole or in part to consult these resources when developing and implementing a mobile device security plan for their own organizations.

B.3 Mobile Application Security

The security qualities of an entire platform can be compromised if an application (app) exhibits vulnerable or malicious behavior. Application security is paramount in ensuring that the security controls implemented in other architecture components can effectively mitigate threats. The practice of making sure that an application is secure is known as software assurance (SwA). This is defined as “the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle, and that the software functions in the intended manner” [22].

In an architecture that largely relies on third-party—usually closed-source—applications to handle daily user functions, good SwA hygiene can be difficult to implement. To address this problem, NIST has released guidance on how to structure and implement an application-vetting process (also known as “app vetting”) [23]. This takes an organization through the following steps:

1. understanding the process for vetting the security of mobile applications
2. planning for the implementation of an app-vetting process
3. developing app security requirements
4. understanding the types of app vulnerabilities and the testing methods used to detect those vulnerabilities
5. determining whether an app is acceptable for deployment on the organization’s mobile devices

Public safety organizations (PSOs) should carefully consider their application-vetting needs. Though major mobile application stores, such as Apple’s iTunes Store and Google’s Play Store, have vetting mechanisms to find vulnerable and malicious applications, organizations may have needs beyond these proprietary tools. Per NIST SP 800-163, *Vetting the Security of Mobile Applications* [23]:

> *App stores may perform app vetting processes to verify compliance with their own requirements. However, because each app store has its own unique, and not always transparent, requirements and vetting processes, it is necessary to consult current agreements and documentation for a particular app store to assess its practices. Organizations should not assume that an app has been fully vetted and conforms to their security requirements simply because it is available through an official app store. Third party assessments that carry a moniker of “approved by” or “certified by” without providing details of which tests are performed, what the findings were, or how apps are scored or rated, do not provide a reliable indication of software assurance. These assessments are also unlikely to take organization specific requirements and recommendations into account, such as federal-specific cryptography requirements.*

The First Responder Network Authority (FirstNet) provides an app store specifically geared toward first responder applications. Through the FirstNet App Developer Program [24], app developers can submit mobile apps for evaluation against its published development guidelines. The guidelines include security, scalability, and availability, along with other requirements. Compliant apps can be selected for inclusion in the FirstNet App Store. This provides first responder agencies with a repository of apps that have been tested to a known set of standards.

PSOs should avoid the unauthorized “side loading” of mobile applications that are not subject to organizational vetting requirements.

B.4 Enterprise Mobility Management

The rapid evolution of mobile devices has introduced new paradigms for work environments, along with new challenges for enterprise IT to address. EMM solutions, as part of an EMM program, provide a variety of ways to view, organize, secure, and maintain a fleet of mobile devices. EMM solutions can vary greatly in form and function, but, in general, they make use of platform-provided application programming interfaces (APIs). Sections 3 and 4 of NIST SP 800-124 [25] describe the two basic approaches of EMM, along with components, capabilities, and their uses. One approach, commonly known as “fully managed,” controls the entire device. Another approach, usually used for bring-your-own-device situations, wraps or “containerizes” apps inside a secure sandbox so that they can be managed without affecting the rest of the device.

EMM capabilities can be grouped into four general categories:

1. General policy – centralized technology to enforce security policies of particular interest for mobile device security, such as accessing hardware sensors like global positioning system (GPS), accessing native operating-system (OS) services like a web browser or email client, managing wireless networks, monitoring when policy violations occur, and limiting access to enterprise services if the device is vulnerable or compromised

2. Data communication and storage – automatically encrypting data in transit between the device and the organization (e.g., through a virtual private network [VPN]); strongly encrypting data at rest on internal and removable media storage; and wiping the device if it is being reissued to another user, has been lost, or has surpassed a certain number of incorrect unlock attempts

3. User and device authentication – requiring a device password/passcode and parameters for password strength, remotely restoring access to a locked device, automatically locking the device after an idle period, and remotely locking the device if needed

4. Applications – restricting which app stores may be used, restricting which apps can be installed, requiring specific app permissions (such as using the camera or GPS), restricting the use of OS synchronization services, verifying digital signatures to ensure that apps are unmodified and sourced from trusted entities, and automatically installing/updating/removing applications according to administrative policies

Public safety and first responder (PSFR) organizations will have different requirements for EMM; this document does not prescribe any specific process or procedure, but assumes that they have been established in accordance with agency requirements. However, sections of this document refer to the NIST Mobile Threat Catalogue (MTC) [21], which does list the use of EMM solutions as mitigations for certain types of threats.

B.5 FIDO Enrollment Process

Fast Identity Online (FIDO) provides a framework for users to register a variety of different multifactor authenticators and use them to authenticate to applications and identity providers (IdPs). Before an authenticator can be used in an online transaction, it must be associated with the user's identity. This process is described in NIST SP 800-63B [10] as *authenticator binding*. NIST SP 800-63B specifies requirements for binding authenticators to a user's account both during initial enrollment and after enrollment, and recommends that relying parties (RPs) support binding multiple authenticators to each user's account to enable alternative strong authenticators in case the primary authenticator is lost, stolen, or damaged.

Authenticator binding may be an in-person or remote process, but, in both cases, the user's identity and control over the authenticator being bound to the account must be established. This is related to identity proofing, discussed in Section B.1, but requires that credentials be issued in a manner that maintains a tight binding with the user identity that has been established through proofing. PSFR organizations will have different requirements for identity and credential management; this document does not prescribe any specific process or procedure, but assumes that they have been established in accordance with agency requirements.

As an example, in-person authenticator binding could be implemented by having administrators authenticate with their own credentials and authorize the association of an authenticator with an enrolling user's account. Once a user has one enrolled authenticator, it can be used for online enrollment of other authenticators at the same assurance level or lower. Allowing users to enroll strong, multifactor authenticators based on authentication with weaker credentials, such as username and password or knowledge-based questions, can undermine the security of the overall authentication scheme and should be avoided.

Appendix C Architectural Considerations for the Mobile Application Single Sign-On Build

This appendix details architectural considerations relating to single sign-on (SSO) with Open Authorization (OAuth) 2.0, Internet Engineering Task Force (IETF) Request for Comments (RFC) 8252, and AppAuth open-source libraries; federation; and types of multifactor authentication (MFA).

C.1 SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries

As stated above, SSO streamlines the user experience by enabling a user to authenticate once and to subsequently access different applications (apps) without having to authenticate again. SSO on mobile devices is complicated by the sandboxed architecture, which makes it difficult to share the session state with back-end systems between individual apps. Enterprise Mobility Management (EMM) vendors have provided solutions through proprietary software development kits (SDKs), but this approach requires integrating the SDK with each individual app, and does not scale to a large and diverse population, such as the public safety and first responder (PSFR) user community.

OAuth 2.0, when implemented in accordance with RFC 8252 (the *OAuth 2.0 for Native Apps* Best Current Practice [BCP]), provides a standards-based SSO pattern for mobile apps. The OpenID Foundation's AppAuth libraries [14] can facilitate building mobile apps in full compliance with IETF RFC 8252, but any mobile app that follows RFC 8252's core recommendation of using a shared external user-agent for the OAuth authorization flow will have the benefit of SSO.

To implement SSO with OAuth 2.0, this practice guide recommends that app developers choose one of the following options:

- They can implement IETF RFC 8252 themselves. This RFC specifies that OAuth 2.0 authorization requests from native apps should be made only through external user-agents, primarily the user's browser. This specification details the security and usability reasons for why this is the case and how native apps and authorization servers can implement this best practice. RFC 8252 also recommends the use of Proof Key for Code Exchange (PKCE), as detailed in RFC 7636 [26], which protects against authorization code interception attacks.
- They can integrate the AppAuth open-source libraries (that implement RFC 8252 and RFC 7636) for mobile SSO. The AppAuth libraries make it easy for application developers to enable standards-based authentication, SSO, and authorization to application programming interfaces (APIs). This was the option chosen by the implementers of this build.

When OAuth is implemented in a native app, it operates as a *public client;* this presents security concerns with aspects like client secrets and redirected uniform resource identifiers (URIs). The AppAuth pattern mitigates these concerns and provides several security advantages for developers. The primary benefit of RFC 8252 is that native apps use an external user-agent (e.g., the Chrome for Android web browser), instead of an embedded user-agent (e.g., an Android WebView) for their OAuth authorization requests.

An embedded user-agent is demonstrably less secure and user-friendly than an external user-agent. Embedded user-agents potentially allow the client to log keystrokes, capture user credentials, copy session cookies, and automatically submit forms to bypass user consent. In addition, because session information for embedded user-agents is stored on a per-app basis, this does not allow for SSO functionality, which users generally prefer and which this practice guide sets out to implement. Recent versions of Android and iPhone operating system (iOS) both provide implementations of “in-app browser tabs” that retain the security benefits of using an external user-agent, while avoiding visible context-switching between the app and the browser; RFC 8252 recommends their use where available. In-app browser tabs are supported in Android 4.1 and higher, and iOS 9 and higher.

AppAuth also requires that public client apps eschew client secrets in favor of PKCE, which is a standard extension to the OAuth 2.0 framework. When using the AppAuth pattern, the following steps are performed:

1. The user opens the client app and initiates a sign-in.
2. The client uses a browser to initiate an authorization request to the authentication server (AS).
3. The user authenticates to the identity provider (IdP).
4. The OpenID Connect (OIDC) / security assertion markup language (SAML) flow takes place, and the user authenticates to the AS.
5. The browser requests an authorization code (“grant”) from the AS.
6. The browser returns the grant to the client.
7. The client uses its grant to request and obtain an access token.

There is a possible attack vector at the end user’s device in this workflow if PKCE is not enabled. During Step 6, the AS redirects the browser to a URI on which the client app is listening, so that the client app can receive the grant. However, a malicious app could register for this URI, and attempt to intercept the grant so that it may obtain an access token. PKCE-enabled clients use a dynamically generated random *code verifier* to ensure proof of possession for the grant. If the grant is intercepted by a malicious app before being returned to the client, the malicious app will be unable to use the grant without the client’s secret verifier.

AppAuth also outlines several other actions to consider, such as three types of redirect URIs, native app client registration guidance, and using reverse domain-name-based schemes. These are supported and/or enforced with secure defaults in the AppAuth libraries. The libraries are open-source and include sample code for implementation. In addition, if Universal Second Factor (U2F) or Universal Authentication Framework (UAF) is desired, that flow is handled entirely by the external user-agent, so client apps do not need to implement any of that functionality.

The AppAuth library takes care of several boilerplate tasks for developers, such as caching access tokens and refresh tokens, checking access-token expiration, and automatically refreshing access tokens. To implement the AppAuth pattern in an Android app using the provided library, a developer needs to perform the following actions:

- add the Android AppAuth library as a Gradle dependency
- add a redirect URI to the Android manifest
- add the Java code to initiate the AppAuth flow, and to use the access token afterward
- register the app's redirect URI with the AS

To implement the AppAuth pattern *without* using a library, the user will need to follow the general guidance laid out in RFC 8252, review and follow the OS-specific guidance in the AppAuth documentation [14], and adhere to the requirements of both the OAuth 2.0 framework documented in RFC 6749 [27], and PKCE.

C.1.1 Attributes and Authorization

Authorization, in the sense of applying a policy to determine the rights and privileges that apply to application requests, is beyond the scope of this practice guide. OAuth 2.0 provides delegation of user authorizations to mobile apps acting on their behalf, but this is distinct from the authorization policy enforced by the application. The guide is agnostic to the specific authorization model (e.g., role-based access control [RBAC], attribute-based access control [ABAC], capability lists) that applications will use, and the SSO mechanism documented here is compatible with virtually any back-end authorization policy.

While applications could potentially manage user roles and privileges internally, federated authentication provides the capability for the IdP to provide user attributes to relying parties (RPs). These attributes might be used to map users to defined application roles, or used directly in an ABAC policy (e.g., to restrict access to sworn law enforcement officers). Apart from authorization, attributes may provide identifying information useful for audit functions, contact information, or other user data.

In the build architecture, the AS is an RP to the user's IdP, which is either a SAML IdP or an OIDC provider. SAML IdPs can return attribute elements in the SAML response. OIDC providers can return attributes as claims in the identification (ID) token, or the AS can request them from the user information endpoint. In both cases, the AS can validate the IdP's signature of the asserted attributes to ensure their validity and integrity. Assertions can also optionally be encrypted, which both protects their confidentiality in transit and enforces audience restrictions because only the intended RP will be able to decrypt them.

Once the AS has received and validated the asserted user attributes, it could use them as issuance criteria to determine whether an access token should be issued for the client to access the requested scopes. In the OAuth 2.0 framework, *scopes* are individual access entitlements that can be granted to a client application. In addition, the attributes could be provided to the protected resource server to enable the application to enforce its own authorization policies. Communications between the AS and protected resource are internal design concerns for the software-as-a-service (SaaS) provider. One method of providing attributes to the protected resource is for the AS to issue the access token as a JavaScript object notation (JSON) web token (JWT) containing the user's attributes. The protected resource could also obtain attributes by querying the AS's token introspection endpoint, where they could be provided as part of the token metadata in the introspection response.

C.2 Federation

The preceding section discussed the communication of attributes from the IdP to the AS for use in authorization decisions. In the build architecture, it is assumed that the SaaS provider may be an RP of many IdPs supporting different user organizations. Several first responder organizations have their own IdPs, each managing its own users' attributes. This presents a challenge if the RP needs to use those attributes for authorization. Local variations in attribute names, values, and encodings would make it difficult to apply a uniform authorization policy across the user base. If the SaaS platform enables the sharing of sensitive data between organizations, participants would need some assurance that their partners were establishing and managing user accounts and attributes appropriately—promptly removing access for terminated employees, and performing appropriate validation before assigning attributes that enable privileged access. Federations attempt to address this issue by creating common profiles and policies governing the use and management of attributes and authentication mechanisms, which members are expected to follow. This facilitates interoperability, and members are also typically audited for compliance with the federation's policies and practices, enabling mutual trust in attributes and authentication.

As an example, National Identity Exchange Federation (NIEF) is a federation serving law-enforcement organizations and networks, including the Federal Bureau of Investigation (FBI), the Department of Homeland Security (DHS), the Regional Information Sharing System (RISS), and the Texas Department of Public Safety. NIEF has established SAML profiles for both web-browser and system-to-system use cases, and a registry of common attributes for users, resources, and other entities. NIEF attributes are grouped into attribute bundles, with some designated as mandatory, meaning that all participating IdPs must provide those attributes, and participating RPs can depend on their presence in the SAML response.

The architecture documented in this build guide is fully compatible with NIEF and other federations, though this would require configuring IdPs and RPs in compliance with the federation's policies. The use of SAML IdPs is fully supported by this architecture, as is the coexistence of SAML IdPs and OIDC providers.

NIST SP 800-63-3 [16] defines Federation Assurance Levels (FALs) and their implementation requirements. FALs are a measure of the assurance that assertions presented to an RP are genuine and unaltered, pertain to the individual presenting them, are not subject to replay at other RPs, and are protected from many additional potential attacks on federated authentication schemes. A high-level summary of the requirements for FALs 1–3 is provided in Table C-1.

Table C-1 FAL Requirements

FAL	Requirement
1	Bearer assertion, signed by IdP
2	Bearer assertion, signed by IdP and encrypted to RP
3	Holder of key assertion, signed by IdP and encrypted to RP

IdPs typically sign assertions, and this functionality is broadly supported in available software. For SAML, the IdP's public key is provided in the SAML metadata. For OIDC, the public key can be provided through the discovery endpoint, if supported; otherwise, the key would be provided to the RP out of band. Encrypting assertions is also relatively trivial and requires providing the RP's public key to the IdP. The build architecture in this guide can support FAL-1 and FAL-2 with relative ease.

The requirement for holder of key assertions makes FAL-3 more difficult to implement. A SAML holder of key profile exists, but has never been widely implemented in a web-browser SSO context. The OIDC Core specification does not include a mechanism for a holder of key assertions; however, the forthcoming token binding over the Hypertext Transfer Protocol (HTTP) specification [28] and related RFCs may provide a pathway to supporting FAL-3 in an OIDC implementation.

C.3 Authenticator Types

When considering MFA implementations, PSFR organizations should carefully consider organizationally defined authenticator requirements. These requirements may include, but are not limited to:

- the sensitivity of data being accessed and the commensurate level of authentication assurance needed
- environmental constraints, such as gloves or masks, that may limit the usability and effectiveness of certain authentication modalities

- costs throughout the authenticator life cycle, including authenticator binding, loss, theft, unauthorized duplication, expiration, and revocation
- policy and compliance requirements, such as the Health Insurance Portability and Accountability Act (HIPAA) [29], the Criminal Justice Information System (CJIS) Security Policy [30], or other organizationally defined requirements
- support of current information-technology (IT) infrastructure, including mobile devices, for various authenticator types

The new, third revision of NIST SP 800-63, *Digital Identity Guidelines* [16], is a suite of documents that provide technical requirements and guidance for federal agencies implementing digital identity services, and may assist PSFR organizations when selecting authenticators. The most significant difference from previous versions of NIST SP 800-63 is the retirement of the previous assurance rating system, known as the Levels of Assurance (LOA), established by Office of Management and Budget Memorandum M-04-04, *E-Authentication Guidance for Federal Agencies*. In the new NIST SP 800-63-3 guidance, digital identity assurance is split up into three ordinals, as opposed to the single ordinal in LOA. The three ordinals are listed below:

- identity assurance level
- authenticator assurance level (AAL)
- FAL

This practice guide is primarily concerned with AALs and how they apply to the reference architecture outlined in Table 3-2.

The strength of an authentication transaction is measured by the AAL. A higher AAL means stronger authentication, and requires more resources and capabilities by attackers to subvert the authentication process. We discuss a variety of multifactor implementations in this practice guide. NIST SP 800-63-3 gives us a reference to map the risk reduction of the various implementations recommended in this practice guide.

The AAL is determined by authenticator type and combination, verifier requirements, reauthentication policies, and security controls baselines, as defined in NIST SP 800-53, *Security and Privacy Controls for Federal Information Systems and Organizations* [31]. A summary of requirements at each of the levels is provided in Table C-2.

A memorized secret (most commonly implemented as a password) satisfies AAL1, but this alone is not enough to reach the higher levels shown in Table C-2. For AAL2 and AAL3, some form of MFA is required. MFA comes in many forms. The architecture in this practice guide describes two examples. One example is a multifactor software cryptographic authenticator, where a biometric authenticator application is installed on the mobile device—the two factors being possession of the private key and the biometric. The other example is a combination of a memorized secret and a single-factor cryptographic device, which performs cryptographic operations via a direct connection to the user endpoint.

Reauthentication requirements also become more stringent for higher levels. AAL1 requires reauthentication only every 30 days, but AAL2 and AAL3 require reauthentication every 12 hours. At AAL2, users may reauthenticate using a single authentication factor, but, at AAL3, users must reauthenticate using both of their authentication factors. At AAL2, 30 minutes of idle time is allowed, but only 15 minutes is allowed at AAL3.

For a full description of the different types of multifactor authenticators and AAL requirements, please refer to NIST SP 800-63B [10].

Table C-2 AAL Summary of Requirements

Requirement	AAL1	AAL2	AAL3
Permitted authenticator types	Memorized Secret; Look-up Secret; Out-of-Band; Single Factor (SF) One-time Password (OTP) Device; Multifactor (MF) OTP Device; SF Crypto Software; SF Crypto Device; MF Crypto Software; MF Crypto Device	MF OTP Device; MF Crypto Software; MF Crypto Device; or Memorized Secret plus: ▪ Look-up Secret ▪ Out-of-Band ▪ SF OTP Device ▪ SF Crypto Software ▪ SF Crypto Device	MF Crypto Device; SF Crypto Device plus Memorized Secret; SF OTP Device plus MF Crypto Device or Software; SF OTP Device plus SF Crypto Software plus Memorized Secret
Federal Information Processing Standard (FIPS) 140-2 verification	Level 1 (government agency verifiers)	Level 1 (government agency authenticators and verifiers)	Level 2 overall (MF authenticators) Level 1 overall (verifiers and SF Crypto Devices) Level 3 physical security (all authenticators)
Reauthentication	30 days	12 hours, or after 30 minutes of inactivity; MAY use one authentication factor	12 hours, or after 15 minutes of inactivity; SHALL use both authentication factors
Security controls	NIST SP 800-53 Low Baseline (or equivalent)	NIST SP 800-53 Moderate Baseline (or equivalent)	NIST SP 800-53 High Baseline (or equivalent)

Requirement	AAL1	AAL2	AAL3
Man-in-the-middle resistance	Required	Required	Required
Verifier-impersonation resistance	Not required	Not required	Required
Verifier-compromise resistance	Not required	Not required	Required
Replay resistance	Not required	Required	Required
Authentication intent	Not required	Recommended	Required
Records retention policy	Required	Required	Required
Privacy controls	Required	Required	Required

The FIDO Alliance has published specifications for two types of authenticators based on UAF and U2F. These protocols operate agnostic of the FIDO authenticator, allowing public safety organizations (PSOs) to choose any FIDO-certified authenticator that meets operational requirements and to implement it with this solution. As new FIDO-certified authenticators become available in the marketplace, PSOs may choose to migrate to these new authenticators if they better meet PSFR needs in their variety of duties.

C.3.1 UAF Protocol

The UAF protocol [2] allows users to register their device to the online service by selecting a local authentication mechanism, such as swiping a finger, looking at the camera, speaking into the microphone, or entering a Personal Identification Number (PIN). The UAF protocol allows the service to select which mechanisms are presented to the user. Once registered, the user simply repeats the local authentication action whenever they need to authenticate to the service. The user no longer needs to enter their password when authenticating from that device. UAF also allows experiences that combine multiple authentication mechanisms, such as fingerprint plus PIN. Data used for local user verification, such as biometric templates, passwords, or PINs, is validated locally on the device and is not transmitted to the server. Authentication to the server is performed with a cryptographic key pair, which is unlocked after local user verification.

C.3.2 U2F Protocol

The U2F protocol [3] allows online services to augment the security of their existing password infrastructure by adding a strong second factor to user login, typically an external hardware-backed cryptographic device. The user logs in with a username and password as before, and is then prompted to present the external second factor. The service can prompt the user to present a second-factor device at any time that it chooses. The strong second factor allows the service to simplify its passwords (e.g., four-digit PIN) without compromising security. During registration and authentication, the user presents the second factor by simply pressing a button on a universal serial bus (USB) device or tapping over Near Field Communication (NFC).

The user can use their FIDO U2F device across all online services that support the protocol. On desktop operating systems, the Google Chrome and Opera browsers currently support U2F. U2F is also supported on Android through the Google Authenticator app, which must be installed from the Play Store. The 2.0 iteration of the FIDO standards will support the World Wide Web Consortium's (W3C) work-in-progress Web Authentication standard [32]. As a draft W3C recommendation, Web Authentication is expected to be widely adopted by web browser developers and to provide out-of-the-box U2F support, without the need to install additional client apps or extensions.

C.3.3 FIDO Key Registration

From the perspective of an IdP, enabling users to authenticate themselves with FIDO-based credentials requires that users register a cryptographic key with the IdP and associate the registered key with the username or distinguished name known to the IdP. FIDO registration might be repeated for each authenticator that the user chooses to associate with their account. FIDO protocols are different from most authentication protocols, in that they permit registering multiple cryptographic keys (from different authenticators) to use with a single account. This is convenient for end users, as it provides a natural backup solution to lost, misplaced, or forgotten authenticators—users may use any one of their registered authenticators to access their applications.

The process of a first-time FIDO key registration is fairly simple:

1. A user creates an account for themselves at an application site, or one is created for them as part of a business process.
2. The user registers a FIDO key with the application through one of the following processes:
    a. as part of the account self-creation process
    b. as part of receiving an email with an invitation to register
    c. as part of a registration process, after an authentication process within an organization application
    d. A FIDO authenticator with a temporary, preregistered key is provided so that the user can strongly authenticate to register a new key with the application, at which point the temporary key is deleted permanently. Authenticators with preregistered keys may be combined with shared secrets given/sent to the user out-of-band to verify their identity before enabling them to register a new FIDO key with the organization's application.
    e. as part of a custom process local to the IdP

Policy at the organization dictates what might be considered most appropriate for a registration process.

C.3.4 FIDO Authenticator Attestation

To meet AAL requirements, RPs may need to restrict the types of FIDO authenticators that can be registered and used to authenticate. They may also require assurances that the authenticators in use are not counterfeit or vulnerable to known attacks. The FIDO specifications include mechanisms that enable the RP to validate the identity and security properties of authenticators, which are provided in a standard metadata format.

Each FIDO authenticator has an attestation key pair and certificate. To maintain FIDO's privacy guarantees, these attestation keys are not unique for each device, but are typically assigned on a manufacturing batch basis. During authenticator registration, the RP can check the validity of the attestation certificate and validate the signed registration data to verify that the authenticator possesses the private attestation key.

For software authenticators, which cannot provide protection of a private attestation key, the UAF protocol allows for surrogate basic attestation. In this mode, the key pair generated to authenticate the user to the RP is used to sign the registration data object, including the attestation data. This is analogous to the use of self-signed certificates for HTTPS, in that it does not actually provide cryptographic proof of the security properties of the authenticator. A potential concern is that the RP could not distinguish between a genuine software authenticator and a malicious lookalike authenticator that could provide registered credentials to an attacker. In an enterprise setting, this concern could be mitigated by delivering the valid authenticator app by using EMM or another controlled distribution mechanism.

Authenticator metadata would be most important in scenarios where an RP accepts multiple authenticators with different assurance levels and applies authorization policies based on the security properties of the authenticators (e.g., whether they provide Federal Information Processing Standard [FIPS] 140-2-validated key storage [33]). In practice, most existing enterprise implementations use a single type of authenticator.

C.3.5 FIDO Deployment Considerations

To support any of the FIDO standards for authentication, some integration needs to happen on the server side. Depending on how the federated architecture is set up—whether with OIDC or SAML—this integration may look different. In general, there are two servers where a FIDO server can be integrated: the AS (also known as the RP) and the IdP.

FIDO Integration at the IdP

Primary authentication already happens at the IdP, so logic follows that FIDO authentication (e.g., U2F, UAF) would as well. This is the most common and well-understood model for using a FIDO authentication server, and, consequently, there is solid guidance for setting up such an architecture. The

IdP already has detailed knowledge of the user and directly interacts with the user (e.g., during registration), so it is not difficult to insert the FIDO server into the registration and authentication flows. In addition, this gives PSOs the most control over the security controls that are used to authenticate their users. However, there are a few downsides to this approach:

- The PSO must now budget, host, manage, and/or pay for the cost of the FIDO server.
- The only authentication of the user at the AS is the bearer assertion from the IdP, so an assertion intercepted by an attacker could be used to impersonate the legitimate user at the AS.

FIDO Integration at the AS

Another option is to integrate FIDO authentication at the AS. One benefit of this is that PSOs will not be responsible for the expenses of maintaining a FIDO server. In addition, an attacker who intercepted a valid user's SAML assertion or ID token could not easily impersonate the user because of the requirement to authenticate to the AS as well. This approach assumes that some mechanism is in place for tightly binding the FIDO authenticator with the user's identity, which is a nontrivial task. In addition, this approach has several downsides:

- Splitting authentication into a two-stage process that spans the IdP and AS is a less well-understood model for authentication, which may lead to subtle issues.
- The AS does not have detailed knowledge of—or direct action with—users, so enrollment is more difficult.
- Users would have to register their FIDO authenticators at every AS that is federated to their IdP, which adds complexity and frustration to the process.
- PSOs would lose the ability to enforce which kinds of FIDO token(s) their users utilize.

Appendix D Acronyms

AAL	Authenticator Assurance Level
ABAC	Attribute-Based Access Control
API	Application Programming Interface
AS	Authorization Server
BCP	Best Current Practice
CA	Certificate Authority
CJIS	Criminal Justice Information System
CRADA	Cooperative Research and Development Agreement
CSF	Cybersecurity Framework
CVE	Common Vulnerabilities and Exposures
DHS	Department of Homeland Security
EMM	Enterprise Mobility Management
FAL	Federation Assurance Level
FBI	Federal Bureau of Investigation
FIDO	Fast Identity Online
FIPS	Federal Information Processing Standard
FirstNet	First Responder Network Authority
FOIA	Freedom of Information Act
GPS	Global Positioning System
HIPAA	Health Insurance Portability and Accountability Act
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IA	Information Assurance
ID	Identification
IdP	Identity Provider
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
iOS	iPhone Operating System
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
JSON	JavaScript Object Notation
JWT	JSON Web Token
LES	Law Enforcement Sensitive
LOA	Levels of Assurance
MF	Multifactor
MFA	Multifactor Authentication
MMS	Multimedia Messaging Service
MSSO	Mobile Single Sign-On

MTC	Mobile Threat Catalogue
NCCoE	National Cybersecurity Center of Excellence
NFC	Near Field Communication
NIEF	National Identity Exchange Federation
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency Report
NTP	Network Time Protocol
OAuth	Open Authorization
OEM	Original Equipment Manufacturer
OIDC	OpenID Connect
OOB	Out-of-Band
OS	Operating System
OTP	Onetime Password
PAN	Personal Area Network
PHI	Protected Health Information
PII	Personally Identifiable Information
PIN	Personal Identification Number
PKCE	Proof Key for Code Exchange
PSCR	Public Safety Communications Research
PSFR	Public Safety and First Responder
PSO	Public Safety Organization
PSX	Public Safety Experience
RBAC	Role-Based Access Control
RCS	Rich Communication Services
REST	Representational State Transfer
RFC	Request for Comments
RISS	Regional Information Sharing System
RP	Relying Party
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SD	Secure Digital
SDK	Software Development Kit
SF	Single Factor
SIM	Subscriber Identity Module
SKCE	StrongKey Crypto Engine
SMS	Short Message Service
SP	Special Publication
SSO	Single Sign-On
SwA	Software Assurance
TLS	Transport Layer Security
TPM	Trusted Platform Module
U2F	Universal Second Factor

UAF	Universal Authentication Framework
UI	User Interface
UICC	Universal Integrated Circuit Card
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module
USSD	Unstructured Supplementary Service Data
VoLTE	Voice over Long-Term Evolution
VPN	Virtual Private Network
W3C	World Wide Web Consortium

1514 # Appendix E References

[1] W. Denniss and J. Bradley, *OAuth 2.0 for Native Apps*, Best Current Practice (BCP) 212, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 8252, October 2017. https://www.rfc-editor.org/info/rfc8252 [accessed February 2018].

[2] S. Machani, R. Philpott, S. Srinivas, J. Kemp, and J. Hodges, *FIDO UAF Architectural Overview: FIDO Alliance Implementation Draft*, FIDO Alliance,Wakefield, MA, 2017. https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html [accessed February 2018].

[3] S. Srinivas, D. Balfanz, E. Tiffany, and A. Czeskis, *Universal 2nd Factor (U2F) Overview: FIDO Alliance Proposed Standard*, FIDO Alliance,Wakefield, MA, 2017. https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html [accessed February 2018].

[4] S. Cantor, J. Kemp, R. Philpott, and E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf [accessed February 2018].

[5] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, *OpenID Connect Core 1.0 incorporating errata set 1*, November 2014. http://openid.net/specs/openid-connect-core-1_0.html [accessed February 2018].

[6] Joint Task Force Transformation Initiative, *Guide for Conducting Risk Assessments*, NIST Special Publication (SP) 800-30 Revision 1, National Institute of Standards and Technology, Gaithersburg, MD, September 2012. https://doi.org/10.6028/NIST.SP.800-30r1 [accessed February 2018].

[7] Joint Task Force Transformation Initiative, *Guide for Applying the Risk Management Framework to Federal Information Systems: A Security Life Cycle Approach*, NIST Special Publication (SP) 800-37 Revision 1, National Institute of Standards and Technology, Gaithersburg, MD, February 2010. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r1.pdf [accessed April 2018].

[8] C. Johnson, L. Badger, D. Waltermire, J. Snyder, and C. Skorupka, *Guide to Cyber Threat Information Sharing*, NIST Special Publication (SP) 800-150, National Institute of Standards and Technology, Gaithersburg, MD, October 2016. https://doi.org/10.6028/NIST.SP.800-150 [accessed February 2018].

[9] C. Brown, S. Dog, J. Franklin, N. McNab, S. Voss-Northrop, M. Peck, and B. Stidham, *Assessing Threats to Mobile Devices & Infrastructure: The Mobile Threat Catalogue*, Draft NISTIR 8144, National Institute of Standards and Technology, Gaithersburg, MD, September 2016. https://nccoe.nist.gov/sites/default/files/library/mtc-nistir-8144-draft.pdf [accessed February 2018].

[10] P. Grassi, J. Fenton, E. Newton, R. Perlner, A. Regenscheid, W. Burr, J. Richer, N. Lefkovitz, J. Danker, Y. Choong, K. Greene, and M. Theofanos, *Digital Identity Guidelines: Authentication and Lifecycle Management*, NIST Special Publication (SP) 800-63B, National Institute of Standards and Technology, Gaithersburg, MD, June 2017. https://doi.org/10.6028/NIST.SP.800-63b [accessed February 2018].

[11] P. Grassi, J. Richer, S. Squire, J. Fenton, E. Nadeau, N. Lefkovitz, J. Danker, Y. Choong, K. Greene, and M. Theofanos, *Digital Identity Guidelines: Federation and Assertions*, NIST Special Publication (SP) 800-63C, National Institute of Standards and Technology, Gaithersburg, MD, June 2017. https://doi.org/10.6028/NIST.SP.800-63c [accessed February 2018].

[12] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers, *Systems and software engineering—System life cycle processes*, ISO/IEC/IEEE 15288:2015, 2015. https://www.iso.org/standard/63711.html [accessed February 2018].

[13] R. Ross, M. McEvilley, and J. Carrier Oren, *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*, NIST Special Publication (SP) 800-160, National Institute of Standards and Technology, Gaithersburg, MD, November 2016. https://doi.org/10.6028/NIST.SP.800-160 [accessed February 2018].

[14] *AppAuth*, AppAuth [Web site], https://appauth.io/ [accessed February 2018].

[15] M. Jones and D. Hardt, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 6750, October 2012. https://www.rfc-editor.org/info/rfc6750 [accessed February 2018].

[16] P. Grassi, M. Garcia, and J. Fenton, *Digital Identity Guidelines*, NIST Special Publication (SP) 800-63-3, National Institute of Standards and Technology, Gaithersburg, MD, June 2017. https://doi.org/10.6028/NIST.SP.800-63-3 [accessed February 2018].

[17] T. Lodderstedt, Ed., M. McGloin, and P. Hunt, *OAuth 2.0 Threat Model and Security Considerations*, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 6819, January 2013. https://www.rfc-editor.org/info/rfc6819 [accessed February 2018].

[18] *NIST Internet Time Servers*, NIST [Web site], https://tf.nist.gov/tf-cgi/servers.cgi [accessed February 2018].

[19] P. Grassi, J. Fenton, N. Lefkovitz, J. Danker, Y. Choong, K. Greene, and M. Theofanos, *Digital Identity Guidelines: Enrollment and Identity Proofing*, NIST Special Publication (SP) 800-63A, National Institute of Standards and Technology, Gaithersburg, MD, June 2017. https://doi.org/10.6028/NIST.SP.800-63a [accessed February 2018].

[20] J. Franklin, K. Bowler, C. Brown, S. Edwards, N. McNab, and M. Steele, *Mobile Device Security: Cloud and Hybrid Builds*, NIST Special Publication (SP) 1800-4, National Institute of Standards and Technology, Gaithersburg, MD, November 2015. https://www.nccoe.nist.gov/sites/default/files/library/sp1800/mds-nist-sp1800-4-draft.pdf [accessed February 2018].

[21] C. Brown, S. Dog, J. Franklin, N. McNab, S. Voss-Northrop, M. Peck, and B. Stidham, *Mobile Threat Catalogue*, 2016. https://pages.nist.gov/mobile-threat-catalogue/ [accessed February 2018].

[22] Committee on National Security Systems (CNSS), *National Information Assurance (IA) Glossary*, CNSS Instruction Number 4009, April 2010. https://www.ecs.csus.edu/csc/iac/cnssi_4009.pdf [accessed April 2018].

[23] S. Quirolgico, J. Voas, T. Karygiannis, C. Michael, and K. Scarfone, *Vetting the Security of Mobile Applications*, NIST Special Publication (SP) 800-163, National Institute of Standards and Technology, Gaithersburg, MD, January 2015. https://doi.org/10.6028/NIST.SP.800-163 [accessed February 2018].

[24] *FirstNet App Developer Program*, First Responder Network Authority [Web site], https://www.firstnet.com/apps/app-developer-program [accessed February 2018].

[25] M. Souppaya and K. Scarfone, *Guidelines for Managing the Security of Mobile Devices in the Enterprise*, NIST Special Publication (SP) 800-124 Revision 1, National Institute of Standards and Technology, Gaithersburg, MD, June 2013. https://doi.org/10.6028/NIST.SP.800-124r1 [accessed February 2018].

[26] N. Sakimura, J. Bradley, and N. Agarwal, *Proof Key for Code Exchange by OAuth Public Clients*, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 7636, September 2015. https://www.rfc-editor.org/info/rfc7636 [accessed February 2018].

[27] D. Hardt, Ed., *The OAuth 2.0 Authorization Framework*, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 6749, October 2012. https://www.rfc-editor.org/info/rfc6749 [accessed February 2018].

[28] A. Popov, M. Nystroem, D. Balfanz, A. Langley, N. Harper, and J. Hodges, *Token Binding over HTTP: draft-ietf-tokbind-https-12*, Internet Engineering Task Force (IETF) Internet-Draft, January 2018. https://datatracker.ietf.org/doc/draft-ietf-tokbind-https/ [accessed February 2018].

[29] *Fact Sheet: The Health Insurance Portability and Accountability Act (HIPAA)*, U.S. Department of Labor, Employee Benefits Security Administration [Web site], https://permanent.access.gpo.gov/gpo10291/fshipaa.html [accessed February 2018].

[30] U.S. Department of Justice, Federal Bureau of Investigation, Criminal Justice Information Services Division, *Criminal Justice Information Services (CJIS) Security Policy*, Version 5.6, June 2017. https://www.fbi.gov/services/cjis/cjis-security-policy-resource-center [accessed April 2018].

[31] Joint Task Force Transformation Initiative, *Security and Privacy Controls for Federal Information Systems and Organizations*, NIST Special Publication (SP) 800-53 Revision 4, National Institute of Standards and Technology, Gaithersburg, MD, January 2015. https://dx.doi.org/10.6028/NIST.SP.800-53r4 [accessed February 2018].

[32] V. Bharadwaj, H. Le Van Gong, D. Balfanz, A. Czeskis, A. Birgisson, J. Hodges, M. Jones, R. Lindemann, and J.C. Jones, *Web Authentication: An API for accessing Public Key Credentials Level 1*, W3C Candidate Recommendation, March 2018. https://www.w3.org/TR/webauthn/ [accessed February 2018].

[33] U.S. Department of Commerce. *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards (FIPS) Publication 140-2, May 2001. https://doi.org/10.6028/NIST.FIPS.140-2 [accessed February 2018].