StrongKey Android Client Library (SACL) Release Notes

1—Preview Release 1

1.1—Introduction

The StrongKey Android Client Library is an open source, native Android library providing support for the FIDO2 protocol for native Android apps. It provides the following features:

- It is supported on Android 9 (API 28) "Pie" or greater
- It supports the Java programming language, and does *not* require the use of JavaScript or the WebView component to deliver FIDO capability—it does *not* support the use of external *Security Keys*
- It uses the *AndroidKeystore*—which takes advantage of the *Trusted Execution Environment (TEE)* or a *Secure Element (SE)*, if present—for key generation, storage, and usage. It is always used as a *user verifying platform authenticator (UVPA)*
- It supports *registration, authentication,* and *transaction authorization* using "dynamic linking"—a core requirement of the European Union's *Payment Services Directive, Revised (PSD2)* regulation for *Strong Customer Authentication (SCA)*
- It supports Android's *BiometricPrompt* API for verifying users before enabling use of the FIDO key (or alternate device authentication schemes—PIN, pattern—where biometric capability is unavailable);
- It has out-of-the-box integration with the open-source FIDO Certified® ***StrongKey FIDO Server (SKFS)***—just add your mobile app to the flow;
- It includes a sample e-commerce web application—the *Sample FIDO App for E-commerce (SFAECO)*—demonstrate four basic functions:
  - User enrollment
  - FIDO registration
  - FIDO authentication and
  - *Authorization* of business transactions with the registered FIDO key
- The server side components of the SFAECO app are installed on a demo
- It includes a sample browser-based web-application to work in concert with the SFAECO app to perform sample back-office business functions. But, the primary purpose is to demonstrate the use of FIDO for strong authentication and to review business transactions performed by app users, as well as see data collected by the app when performing *transaction authorization (TXA)*
- It includes a second sample browser-based FIDO Key Management web application to demonstrate the newly integrated *single sign-on (SSO)* capability of SKFS with *JSON Web Tokens (JWTs)* using X.509-based *JSON Web Signatures (JWS)*
- These web applications have been installed

SACL has been tested using Essential PH-1, Google Pixel 3a, and Google Pixel 4a phones—

the first two running Android 9 (Pie) with API 28, and the Pixel 4a with Android R (API 30). The device must have a fingerprint enrolled to support the use of SACL. While it is likely to work on most Android devices with biometric capability, your mileage may vary.

This is a preview release; some things may be a little rough around the edges. However, it provides insight into the capability the completed product will have, and enables early adopter customers to design and code native Android business application(s) to work with the SACL. StrongKey's goal is to deliver a production quality release by the end of June 2021.

Feedback on the SACL may be sent to getsecure@strongkey.com. Alternatively, post ideas on the forum of the repository where you downloaded the distribution. Thank you for using the SACL to secure your customers' and your data.

1.2—Notes and Known Issues

#	Issue
SACL-0001	In this *Preview Release 1 (PR1)*, SACL is designed to allow multiple credentials to be registered from the SFAECO app, to the same *Relying Party ID (RPID)*. Normally, this would not be allowed, since an Authenticator designed for use by a single user must only have one credential registered to an RPID active at any time. However, this can be very cumbersome when designing your business app and testing it with one mobile device. As such, PR1 does not enforce uniqueness of FIDO keys to a specific RPID within the device. When StrongKey finalizes a Production release later this year, this enforcement will become default.
SACL-0002	On the Pixel 3a and 4a, WiFi services will not automatically turn on, resulting in error messages when attempting any function requiring access to the SFAECO services. Please check for these messages in Logcat if they don't show up in a Toast message. **Workaround:** If your phone is configured to work with your WiFi router, but fails to connect, it could be for a variety of reasons: Location Services are off; Data Saver is on; Advanced Network settings permit the phone to turn off network services when the device goes to sleep, etc. Use the browser on your phone to visit any website. This triggers network services to start. Once started, the SFAECO app will work fine. However, if the device goes to sleep even for a few seconds, it is likely to turn networks services off. You may have to repeat the process to get back on the WiFi network—or turn Location Services on to keep the service on.
SACL-0003	When a user enrolls, and if the device goes to sleep, the SACL will sometimes not persist the *counter* value for the Authenticator. This will result in error messages or crashes because the SKFS default security policy is to require that Authenticator counter values always be incremented. Logcat messages will show exceptions with an *HTTP 500* error. **Workaround:** Should this occur, **Force Stop** the app in your *Application Settings*. Make sure that your network is working (by using a browser to visit any website), and try it again. You should notice that the *counter* value should be greater than "1" in the *Registered Key* page.

#	Issue
SACL-0004	SACL has a default value of 5 minutes for enabling use of the *AndroidKeystore* after the user has successfully unlocked the mobile device using their PIN, pattern, or fingerprint. This is designed to simulate the *Regulatory Technical Specification (RTS*) of the EU PSD2 regulation. As a result, no biometric prompt will show up during FIDO key generation or during authentication with the FIDO key. However, a biometric prompt will *always* show up when the user is authorizing a payment transaction, displaying the "dynamic link" with appropriate payment information. The app was deliberately designed to show different modes in which a FIDO operation can work depending on how the app chooses to use the SACL—either by relying upon the default timeout for the use of *AndroidKeystore* on an unlocked device, or explicitly prompting for a user's fingerprint when confirming a transaction. If a FIDO registration or authentication operation fails in the application, this is most likely due to network timeouts; a secondary possibility is that it has been more than 5 minutes since the user unlocked the device. **Workaround:** Lock the phone by pressing the Power button, unlock it with PIN, pattern, or fingerprint; then restart the application to perform the necessary operation. It will succeed this time.
SACL-0005	The *Authenticator Attestation Globally Unique Identifier (AAGUID)* used by the Preview release of SACL is **CAFEBABECAFEBEEF0123456789ABCDEF**. When SKFS goes into *Generally Available (GA)* status as a production quality release, the AAGUID will become **5341434C323032304b4F52D999D03ECB**.
SACL-0006	Testing has not been performed on multiple apps using SACL on the same mobile device. Ergo, the behavior of this use case is currently unknown. We anticipate testing this capability in the next update and providing guidance.

1.3—Distribution

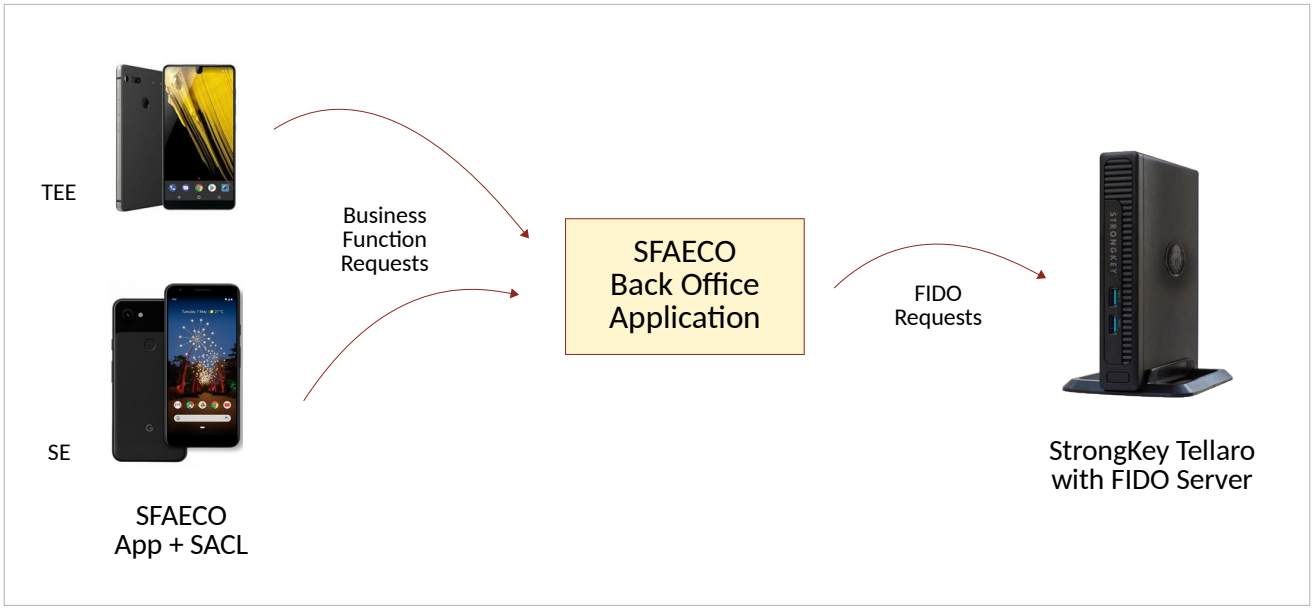
The distribution is available from Github and SourceForge.

When unzipped, the `.ZIP` file expands to show the following folders and source code:

Folder Name	Notes
sfaeco	The NetBeans project that represents the server-side back end of the Sample FIDO App for E-commerce (SFAECO). The project is a Java Enterprise Edition 5 (JEE5) application written in Java8 using NetBeans 11.3 with OpenJDK 1.8.0_272.
skrepo	The folder with the Android Studio projects of the SACL, and the sample app (SFAECO) that uses the library. The projects were written in Java8, using OpenJDK 1.8.0_272 with Android Studio 3.6.1.
sfaeco/ sfaeco-client	The NetBeans module that represents command line (CLI) tools to test enrolling users. Not all web services of the application can be called by this CLI tool, since most of the web services require a FIDO Authenticator.
sfaeco/ sfaeco-ear	The NetBeans module that represents the Enterprise Archive. Its primarily a "container" project to build the `sfaeco-ear-1.0.ear` file which includes the necessary modules to run the backend application within a JEE5 Application Server (such as Payara).
sfaeco/ sfaeco-ejb	The NetBeans module that contains enterprise JavaBeans with the business logic and data model used by the sample application.
sfaeco/ sfaeco-web	The NetBeans module that contains servlet and exposes *Representational State Transfer (REST)* web services the Android app uses to interact with the application and the SKFS.
skrepo/ mobile/ android/ SampleSACL FidoEcoApp/	The Android sample app representing a simulated e-commerce application that demonstrates how to use SACL for the following functions: ▸ Enrolling Users ▸ Registering a FIDO2 key pair for the user ▸ Authenticating with the newly registered FIDO2 key ▸ Choosing one or more products from the Product Gallery to purchase ▸ Choosing a Payment Option to pay for the purchase transaction ▸ Confirming the transaction with a biometric response ▸ Seeing the result of a successful purchase transaction along with FIDO Alliance-EMVCo-defined data elements that can be relayed to issuing banks for authorization over 3DS (see the FIDO Alliance's FIDO Authentication and EMV 3-D Secure—Using FIDO for Payment Authentication white paper for details).
skrepo/ mobile/ android/ StrongKeyAndroidClientLibrary/	The Android client library representing a FIDO Authenticator that uses native Android APIs to support use of the FIDO2 protocol for registration, authentication, and transaction authorization with SKFS. The client library uses the following Android capabilities: ▸ *AndroidKeystore*, using either the Trusted Execution Environment (TEE) or a Secure Element (SE), depending on what is available on the phone ▸ *BiometricPrompt* to verify the user before enabling the use of *AndroidKeystore* for key generation and key usage with FIDO operations ▸ *RoomDB* to store Authenticator data locally on the device

1.4—Architecture of the Sample Application

The application and its back end have the following high-level architecture:



The app and SACL have been tested on the Essential PH-1 phone with Android 9; this device has a TEE within the device.

The app and SACL have also been tested on Google Pixel 3a and Google Pixel 4a phones, which have the Google Titan chip (SE) embedded within the device.

The app communicates over TLS to the SFAECO *Back Office Application (BOA)*, consuming REST web services running by default on StrongKey's PSD2 DEMO server on the internet at https://psd2demo.strongkey.com.

The StrongKey Tellaro also has the latest SKFS release running on the same appliance. While the DEMO uses the StrongKey Tellaro, SKFS can be deployed within virtual machines if desired (with some risk to the integrity of stored objects within the database of the FIDO server; please contact StrongKey if you wish to learn more about the risks of deploying key management solutions in multi-tenant public cloud environments).

1.5—Building the App

Once the distribution is downloaded and verified, unzip SampleFIDOAndroidApp.tgz in a location where you normally work with your Android projects.

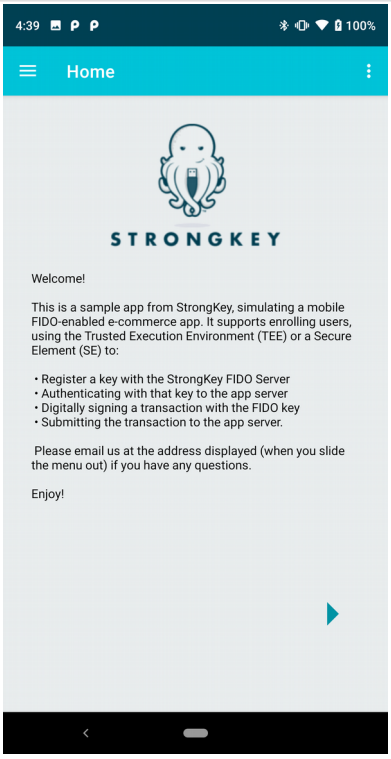
[sha256sum 08a4059c6326fbda30cc244ac5e532434a6ef8057f1aac81d413a87aa16bcc54]

Open the *SampleSACLFidoEcoApp* project. The `settings.gradle` file for the *SampleSACLFidoEcoApp* project must be updated: the *projectDir* variable must reflect where the SACL folder can be found locally on the PC, and then Gradle will connect. Do not update the biometric API; it should be set to 1.0.1.

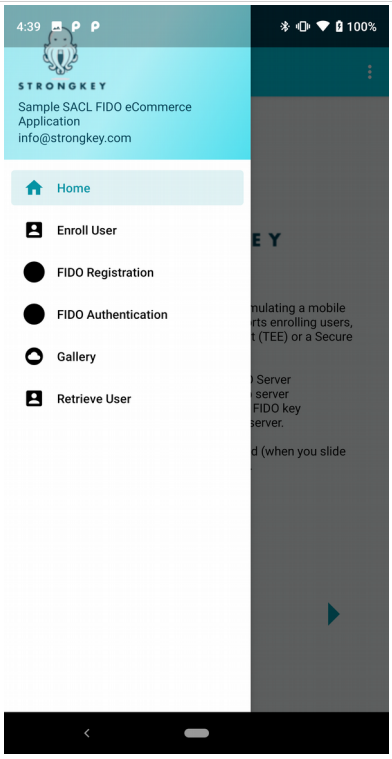
Wait for Android Studio to synchronize all files it needs. When completed, rebuild the app.

Connect your Android 9+ phone to your development PC; this assumes your phone is already setup in Developer Mode.

Launch the app on your phone. Make sure Android Studio is showing Logcat messages and is filtered on the SFAECO app. You should start seeing the following screens on your phone if all goes well:

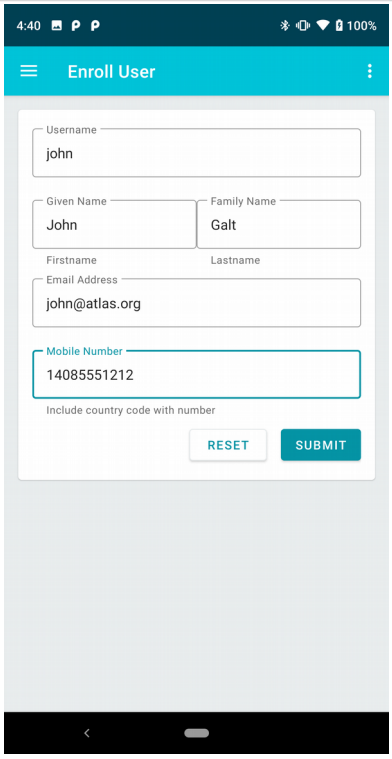
1		The **Home Page** of the app with a Welcome message.

2



The drawer Menu of the app that slides out from the left

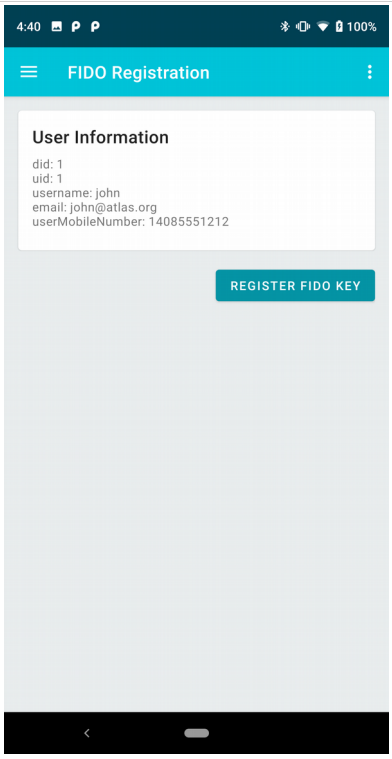
3



The **Enroll User** page where you can create a new user account.

None of the information provided need be real/authentic; the only validation performed is to check the uniqueness of the username, e-mail address, and mobile phone number on the server side—but they can all be fake information.

4



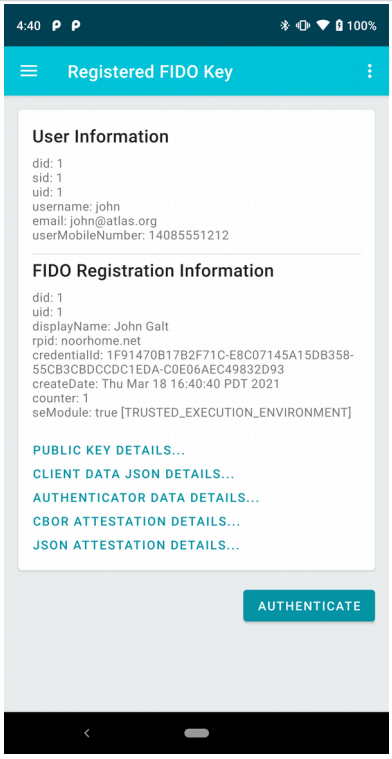
A successfully enrolled user.

Once enrolled, select the **REGISTER FIDO KEY** button to generate your key pair.

If for any reason, you fail to register a key (usually because the network goes to sleep and the TLS connection times out on the app), you can restart the app and choose **FIDO Registration** from the menu. The app will automatically recall the user information last saved on the mobile device.

If by any chance it does have a registered key, information about the registered FIDO key will also be retrieved from RoomDB and displayed.

5



A FIDO key successfully registered with the FIDO server.

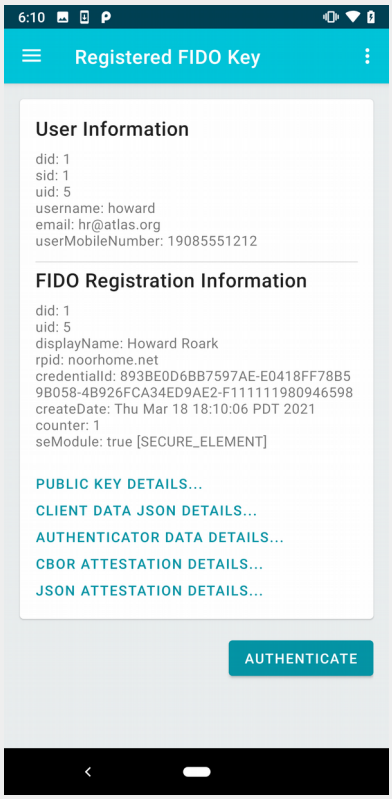
The page displays a fair amount of useful information about the FIDO key and security capability of the mobile device:

- The *relying party ID (RPID)* against which the key is registered
- The credential ID of the FIDO key
- The security capability of the mobile device—in this case, TEE

Each of the highlighted lines in blue provides more detailed information; most of it is not humanly readable, but it is available for developers to see if there is any interest.

Select the **AUTHENTICATE** button to authenticate with the newly registered key to the back-end application.

5A



This image shows the same information as the previous image—with one difference. Since this app was executing on a phone with a **SECURE ELEMENT (SE)**, the SACL detected this and displays this information when a key is registered.

> **NOTE:** In reality, the SACL does not make this determination. When the FIDO keys are generated, an attestation is also generated by *AndroidKeystore*; this attestation—the Android Key Attestation—generates a chain of X.509 digital certificates rooted in the hardware by the manufacturer of the device (in this case, Google, the manufacturer of the Pixel 3a).

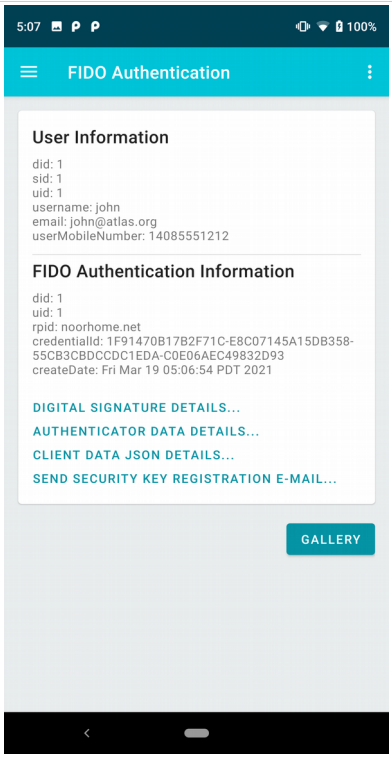
The chain of digital certificates provides information that is parsed by SKFS to determine properties of the public key that was sent for registration and the device in which the key was generated.

That information is relayed back to SACL, which is displayed by the app. This attestation is the assurance a *relying party (RP)* needs to affirm that they are dealing with an authentic device from the manufacturer and the attestation the manufacturer makes about cryptographic keys within that device.

While far more security capabilities are possible based on this technology, this SACL Preview Release is focused on enabling just what the FIDO2 protocol needs to satisfy business requirements.

StrongKey is committed to taking this technology and capability to its fullest potential, delivering some of the strongest security capability with its Tellaro appliance and securing some of the highest-risk business transactions. Please contact us if you would like to learn more.

6



A successfully authenticated user.

The page displays some useful information about the authentication.

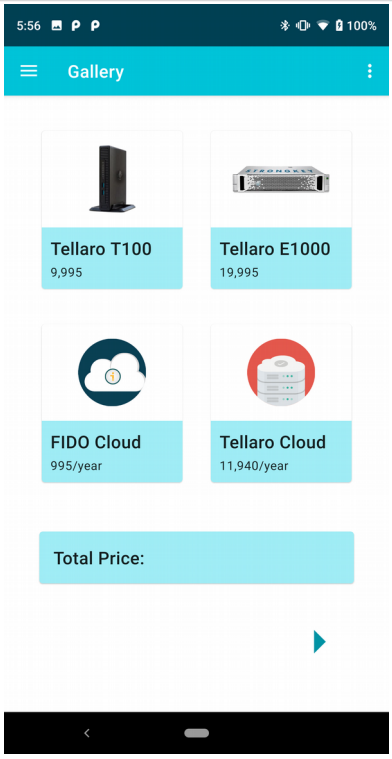
One useful capability (currently not implemented in this Preview) is a prompt to *SEND SECURITY KEY REGISTRATION E-MAIL*.

When a user registers with a site using their mobile device, this FIDO key becomes the strongest way in which the user may interact with the site. Once authenticated, it will be helpful to allow users to email themselves a one-time, time-limited link that allows them to register a second FIDO key using an external *Security Key*.

This has the advantage of allowing the user to use the site from their desktop or laptop, or to allow them to recover their account with a new phone if they lose their current mobile device.

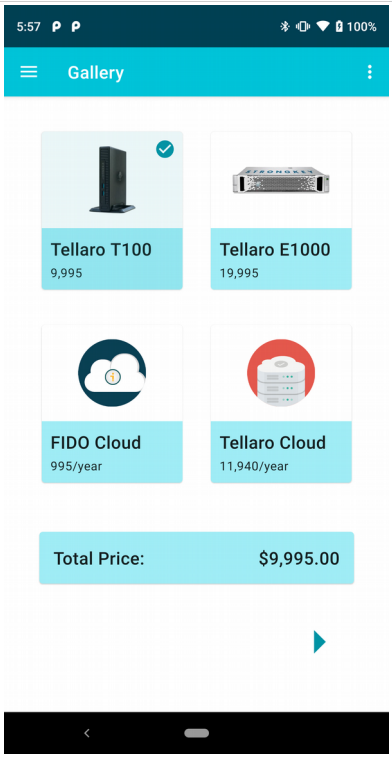
Once authenticated, select the **GALLERY** button to choose a sample product for purchase.

7



The **Product Gallery** displays 4 sample products in tiles that can be selected (or deselected, once selected) as part of the interaction.

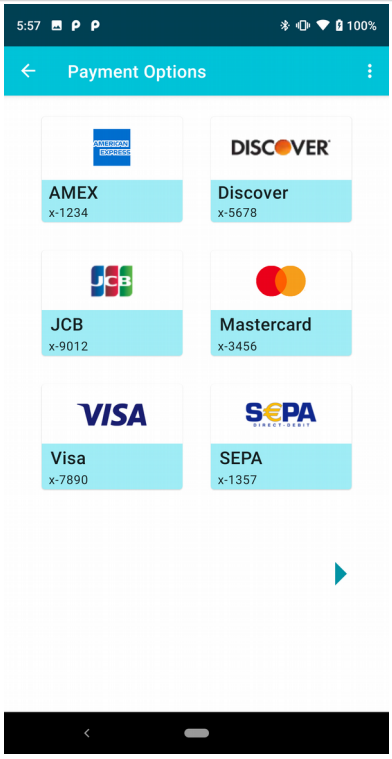
8



This page shows one of the tiles selected in this demonstration.

The arrow at the bottom right advances to next page.

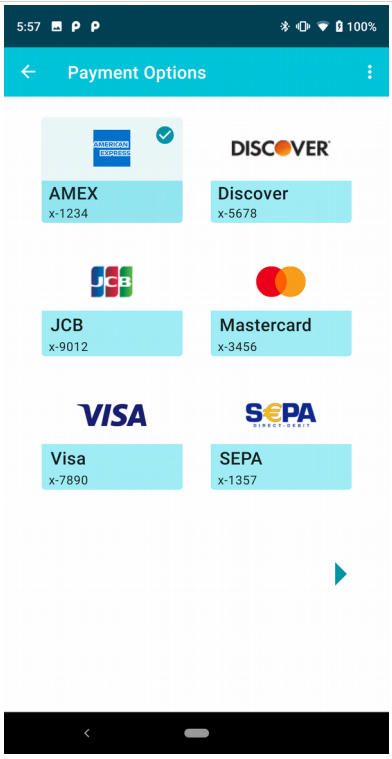
9



The **Payment Options page** displays six sample payment cards in tiles that can be selected\deselected as part of the interaction.

This assumes the app was designed with the capability to store multiple payment options with this site. In this sample app, hard-coded values are used to simulate this function.

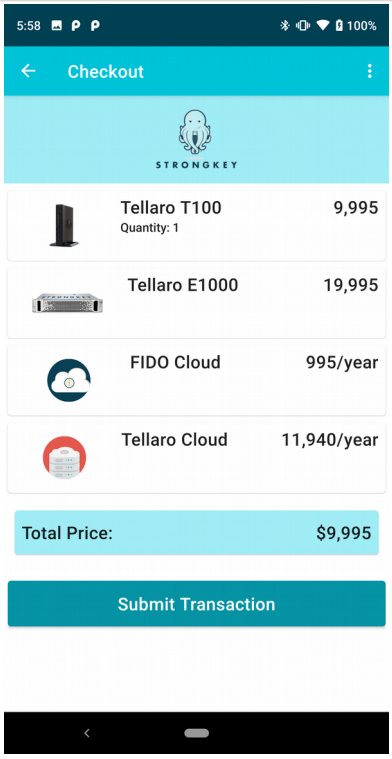
10



This page shows one of the tiles selected in this demonstration.

The arrow at the bottom right advances to next page.

11

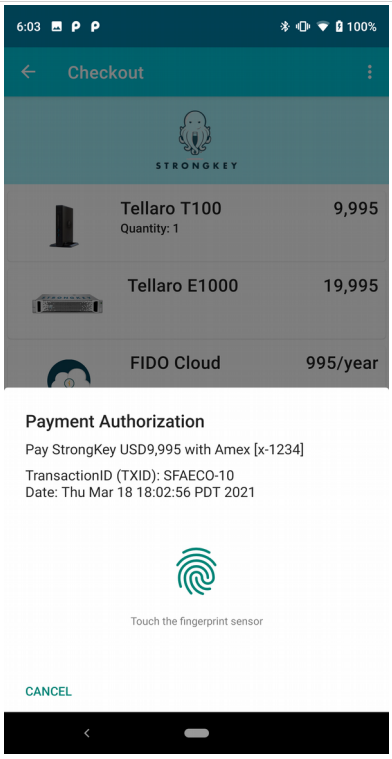


We finally arrive at the **CHECKOUT** page of the sample app.

It shows the which of the four products was chosen from the Product Gallery, and the *Total Price* the user is expected to pay to consummate this transaction.

The **SUBMIT TRANSACTION** is where the magic of FIDO *Transaction Authorization (TXA)* begins within SACL.

12



Having selected the **SUBMIT TRANSACTION** button, the app takes relevant information about the transaction and sends it to the back office application.

The back office application, in turn, processes that information and acquires a unique challenge from the FIDO Server, which takes this transaction's details into account.

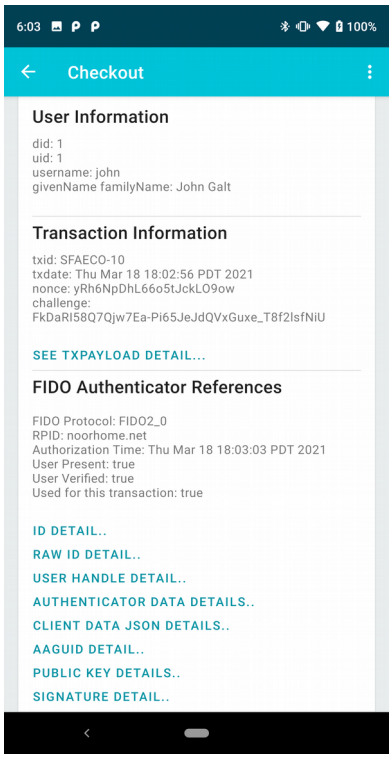
Upon receiving that challenge, the app calls Android's *BiometricPrompt* to display a unique message—a *dynamic link* in PSD2 RTS parlance—that displays necessary information for regulator compliance:

- Payee information
- The amount to be paid, and
- The chosen payment option

The *transaction ID (TXID)* and a Timestamp are added by the app to uniquely identify this transaction within its database.

The user is explicitly prompted to supply their fingerprint to authenticate with the device using the FIDO key to digitally sign the challenge sent by the FIDO server (through the back office application).

13



Upon successfully authenticating to the Android phone, the app—using SACL—uses the FIDO key to digitally sign the unique challenge and confirms the transaction displayed on the secure display.

This page shows the successful transaction with 2 interesting pieces of information:

- The SEE TXPAYLOAD DETAILS is Base64-encoded data of a JSON object with the following details:
  - Merchant name
  - Currency type
  - Total price
  - Card brand
  - Last 4 digits of the card
  - The unique transaction ID
  - The date/time of the transaction
- The information at the bottom, referenced as **FIDO Authenticator References**, refers to data elements standardized by the FIDO Alliance and EMVCo to transmit FIDO-authenticated transaction confirmations over 3DS messages to Issuing Banks for authorization. The information displayed here maps to the details specified in the FIDO Alliance's FIDO Authentication and EMV 3-D Secure—Using FIDO for Payment Authentication white paper.

This concludes the demonstration of the SFAECO app and its use of SACL for using FIDO2 protocols.

SACL can be used for applications in finance, healthcare, education, government, gaming, enterprise apps, etc. While mobile devices have made it significantly easier for users to authenticate to websites (by using biometrics), behind the curtain they still use the ancient password-based authentication schemes that are susceptible to attack and are responsible for more than 95% of all data breaches.

With SACL and SKFS, Android apps can now forever leave passwords behind. We hope you find this opportunity exciting to protect your users, as well as your own sites, by eliminating passwords and all the hassles that come with them.

Drop us a note at getsecure@strongkey.com to tell us what you think.

Enjoy!