Strong Customer Authentication for Apple Pay on Apple Watch Series 10 with S10 running watchOS 11.4

Security Target

Version 1.2
June 18, 2025

Apple
One Apple Park Way
Cupertino, CA 95014

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the Target of Evaluation (TOE) for meeting the requirements of Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market (PSD2) and the Commission Delegated Regulation (EU) 2018/389 of 27 November 2017, focusing on the Strong Customer Authentication and Dynamic Linking for Apple Pay.

1.2. Abbreviations

Abbreviation	Meaning
AP	Application Processor
API	Application Programming Interface
AR	Authorization Random
CDCVM	Consumer Device Cardholder* Verification Method
CL	Contactless
CRS	Contactless Registry Service
CVV	Card Verification Value
HSM	Hardware Security Module
I/O	Input / Output
MAC	Message Authentication Code
NFC	Near Field Communication
OS	Operating System
PNO	Payment Network Operator
PSD2	Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015
SCA	Strong Customer Authentication
SKS	Secure Key Store
SSE	An application in the Secure Enclave managing the pairing between the Secure Enclave and the Secure Element
TSM	Trusted Service Manager
TSP	Token Service Provider
UID	Unique Identifier

* refers to the Apple Pay user.

2. ST Introduction

2.1. Security Target Reference

This Security Target is identified with the following information:

Security Target identifiers	
Title	Strong Customer Authentication for Apple Pay on Apple Watch Series 10 with S10 running watchOS 11.4: Security Target
Version	1.2
Date	June 18, 2025
Developer	Apple Inc.

2.2. Target of Evaluation Reference

TOE identifier
Strong Customer Authentication for Apple Pay on Apple Watch Series 10 with S10 running watchOS 11.4

The Target of Evaluation (TOE) platform is an Apple Watch Series 10 device with the S10 chip running watchOS 11.4, with the following platform identifiers:

Platform identifiers	
Device	Apple Watch Series 10
Operating System	watchOS 11.4 (Build 22T251)
Developer	Apple Inc.

The TOE consists of a range of hardware and software components as listed below, which are all developed by Apple.

TOE Component	Version	Description
Apple Wallet App (Wallet)	App part of watchOS 11.4	Authentication policy on data and services In-app transaction data management
Application Processor (AP)[1]	S10	Authentication policy on data and services In-app transaction data management

[1] Only the parts of the AP related to the TSF are included within the TOE scope. The GPU of the AP is not relevant to the TSF and is therefore not part of the TOE.

TOE Component	Version	Description
Boot Loader	watchOS 11.4	Allows the device to start and boot the operating system
Secure Enclave	sepOS part of watchOS 11.4	Authentication Setup: • Enrollment of the authentication material • User authentication verification Authentication Prover: • Passcode verification • Authentication policy on data
• **SSE**		Manages the pairing between the Secure Enclave and the Secure Element
• **SKS**		Hardware Cryptographic module
watchOS Platform	Device operating system platform (watchOS 11.4) executing on Application Processor (AP) with the following Apple Pay services that are included in the TOE:	
• **Security Framework**	watchOS 11.4	Provides functionality to protect information, establish trust, and control access to software
• **NFCd**	watchOS 11.4	Provides functionality for near field communication
• **Settings**	watchOS 11.4	Allows the user to indicate their preferred settings for the device, operating system, and applications
• **App List**	watchOS 11.4	Provides the functionality for the watchOS user interface
Console	Touchscreen of the Apple Watch Series 10 Device drivers part of watchOS 11.4	Provides the functionality for input/output (I/O)

The Secure Element of the device is separately certified according to the Common Criteria and is therefore out of scope of this evaluation.

Note: *In the evaluated configuration the cryptographic modules are supplied by Apple as part of watchOS and sepOS. Readers may draw some assurance from the conformance to FIPS 140-3 certified by the Cryptographic Module Validation Program for corecrypto for each major release (Apple corecrypto User Space Module, Apple corecrypto Kernel Space Module and the Apple Secure Key Store Cryptographic Module).*

The TOE guidance document is listed in the following table.

Apple Pay Guidance	Reference	Version
Strong Customer Authentication for Apple Pay on Apple Watch Series 10 with S10 running watchOS 11.4: Guidance	[AGD]	1.1

2.3. TOE Overview

2.3.1. TOE type

The TOE is a combination of Hardware and Software components that implement Strong Customer Authentication and Dynamic Linking for Apple Pay on the Apple Watch Series 10 with S10 running watchOS 11.4.

2.3.2. TOE usage and major security features

The TOE includes the components implementing Strong Customer Authentication (SCA) and Dynamic Linking for Apple Pay. The TOE is the Apple Watch Series 10 device with S10 chip running watchOS 11.4 operating system.

The operating system manages the device hardware, provides Apple Pay and Apple Cash functionalities, and provides the technologies required to enforce Strong Customer Authentication and Dynamic Linking for Apple Pay and Apple Cash e-commerce transactions. Dynamic Linking for a transaction is the link between the authentication code generated upon successful SCA, with both the transaction's original specific amount and the identity of the payee.

The operating system provides a consistent set of capabilities allowing the supervision of enrolled devices. This includes the preparation of devices for deployment, the subsequent management of the devices, and the termination of management.

The TOE platform protects itself by having its own code and data protected from unauthorized access (using hardware provided memory protection features), by securing user and TOE Security Functionality (TSF) data, by ensuring the integrity and authenticity of TSF updates and downloaded applications, and by locking the TOE upon user request or via the Wrist Detection feature.

The TOE provides protection of data at rest, and access control mechanisms for use by applications. Access control for data and services, including Apple Pay and Apple Cash, rely on the enforcement of user authentication.

To use Apple Pay, a user must have a passcode set on the device. User authentication to authorize an Apple Pay transaction on an enrolled device is provided by a user-defined passcode. The minimum length of the passcode, passcode rules, and the maximum number of consecutive failed authentication attempts is statically set by Apple for each watchOS release.

The Secure Enclave is responsible for ensuring user authorization (the combination of user authentication and user intent) before a payment is authorized from the device.

Apple Watch can be paired with only one iPhone at a time. When Apple Watch is unpaired, iPhone communicates instructions to erase all content and data from the Watch. The setup of a new Apple Watch is performed by the User on a compatible iOS device. In addition to the preparation of Apple Watch configuration, a link is established between the two devices: the watch is paired with the iOS device and a pairing secret is exchanged.

The paired iPhone is used to manage Apple Watch content (OS version, Wallet cards and configuration, Apps installed, and critical settings like Wrist Detection). The pairing secret allows the watch and the paired iPhone to exchange sensitive data like the unlock secret.

2.3.3.Non-TOE hardware/software/firmware

The TOE environment includes the Secure Element (Hardware, Operating System, CRS applet and payment applets in the Secure Element) and the NFC communication layer (the NFC Controller (NFCC)).

The Secure Element is included in the device but is outside the scope of the TOE boundary and is separately evaluated to Common Criteria. The Secure Element is contained in the same package as the NFCC. Payment applets in the Secure Element manage the payment process for Apple Pay transactions.

The TOE relies on its environment to facilitate Apple Pay transactions (and Apple Cash transfers); the transaction data is always processed by the Secure Element. The Secure Element only allows payment data to be sent from the device after it receives authorization from the Secure Enclave. For each transaction, the payment applets hosted on the Secure Element generate a payment cryptogram. This cryptogram and the Device Account Number form a transaction-specific dynamic security code, which is sent from the device to the card issuer (or its tokenization service provider such as a payment network) to use to verify each transaction.

When interacting with Near Field Communication (NFC) enabled payment terminals, the TOE uses the device's out-of-the-TOE NFC antenna for the communication.

The subsystems of the TOE are listed in Section 2.4 of this Security Target. All other components and subsystems of watchOS (including user space and kernel software), hardware and subsystems included in the device (including the networking subsystem), and the paired iPhone are all considered to be part of the TOE environment. The networking subsystem provides connectivity to the Apple servers responsible for managing Apple Pay transactions.

The "Unlock with iPhone" feature relies on a secure pairing process between the TOE and the iPhone, and a secure unlock process thereafter.

2.4. TOE Description

2.4.1.TOE Architecture

The TOE platform includes the components implementing Strong Customer Authentication (SCA) and dynamic linking for Apple Pay.

User authentication is managed by the Secure Enclave. The Secure Enclave is a dedicated secure subsystem integrated into Apple systems on chip (SoCs). The Secure Enclave is isolated from the main processor to provide an extra layer of security and is designed to keep sensitive user data secure even if the Application Processor kernel were to be compromised.

watchOS allows the Apple Pay services and other security functions of the TOE to operate.

The Secure Element (outside of the TOE) is the secure component that holds the Apple Pay secrets and processes the Apple Pay transactions.

The guidance documentation of the TOE is listed in the *Apple Pay Guidance* table of section 2.2.

The identifiers of the TOE components are given in the *TOE Component* table of section 2.2.

The distribution channels for Users to obtain the devices include:

- The "Apple Store" which is either physical store or online store at https://www.apple.com
- Apple retailer

- Resellers

Other specific channels for Government and business

The component parts of the TOE are shown in Figure 1.

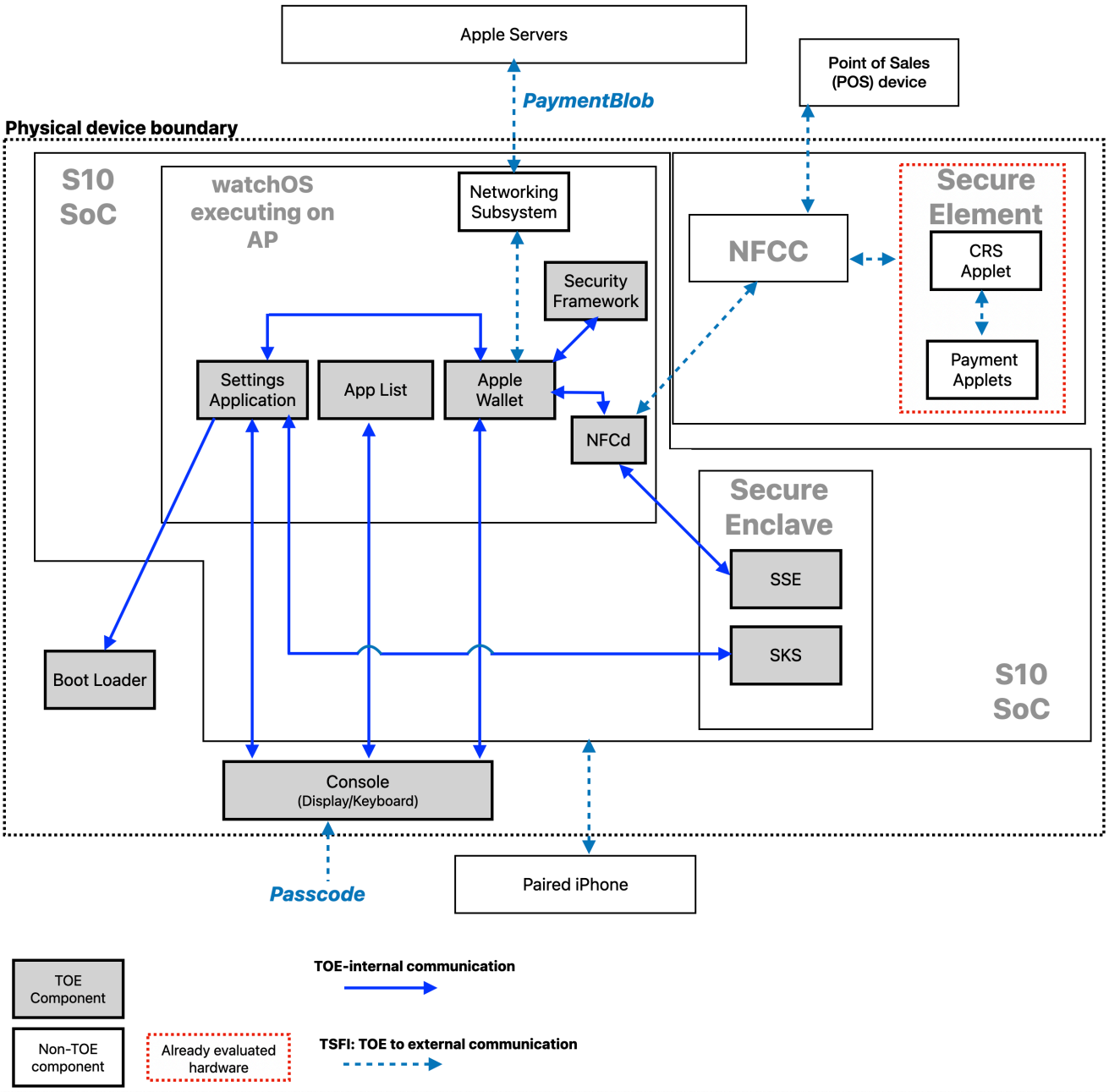

Figure 1: TOE Components and subsystems

2.4.2.Subsystems of the TOE

This section further breaks down the TOE components, providing more detail about the subsystems of the TOE.

The subsystems of the TOE consist of:

- Secure Enclave: The software components of the TOE residing in the Secure Enclave. This subsystem includes several applications executing on the Secure Enclave operating system:
  - The SKS (Secure Key Store) is a hardware cryptographic module. The module is embedded inside the Secure Enclave and packaged within the Application Processor.
  - The SSE (Secure Enclave-Secure Element) manages the pairing between the Secure Enclave and the Secure Element, allowing the Secure Element to process only genuine and authorized Apple Pay transactions. The SSE application maintains sensitive pairing material, allowing Secure Element and Secure Enclave to perform a mutual authentication before exchanging data.
- Apple Wallet: The Wallet app subsystem is an application executing as part of watchOS that handles the enrollment of payment applications and governs the payment operation.
- watchOS components executing on Application Processor (AP):
  - App List: The component of watchOS that handles I/O with the console, and thereby provides the functionality for the watchOS user interface
  - NFCd: This daemon facilitates the communication between Apple Wallet and the Secure Element
  - Security Framework: This is an API[2] provided by the OS to provide cryptographic support that can be used to protect information, establish trust, and control access to software
  - Settings application: This application allows the user to modify various system settings
- Device components:
  - Boot-loader: This subsystem consists of code that is executed during the boot sequence of the device. The boot-loader is responsible for ensuring that the device boots using software with assured integrity and authenticity
  - Console: This is the hardware and associated drivers that handles user input via the keyboard, and displays output via the screen. The touchscreen hardware is also part of the TOE

All other device hardware and watchOS components are outside of the TOE, including the Secure Element together with the NFCC hardware, and the XNU watchOS kernel. The watchOS subsystem components are individual applications.

2.4.3.TOE Lifecycle

The TOE lifecycle phases are as follows:

Lifecycle Phases			
Design	HW/FW design	SW design	Secure Element Applet design
Fabrication	HW fabrication	SW implementation	Applet development

[2] Refer to https://developer.apple.com/documentation/security/

Integration	Watch integration *Assembly, Trust provisioning, FW integration, SW and applet loading*		
Device Issuance	Watch delivery to User		
Initialization	User account creation Account setup: iCloud, Apple Account		
Enrollment/ Provisioning	User Authentication setup *Passcode setup*		Apple Pay provisioning
Usage	Device Usage *Device unlock, User Authentication, watchOS use, (optional) watchOS update*		Apple Pay transaction
Termination	Physical destruction	watchOS reset	Apple Pay termination

The Design, Fabrication, and Integration phases are entirely within the control of Apple Inc.

The Device Issuance phase is the physical delivery of the device to the User. This can be directly, in the case of an individual, or through a third party or an entity that is responsible for providing the device to the User.

Apple's model of the Initialization phase requires that the device is claimed by the User by associating it with the User's iCloud account and Apple Account. Before that point, Strong User Authentication and associated TSFs are not relevant, and Apple Pay is not accessible. Apple Pay and Apple Cash services require that a valid iCloud account and user authentication credentials are set up (passcode).

The Usage phase describes the period when the Apple Pay service is activated and used by the associated User.

The Termination phase describes deactivation of the Apple Pay service for the User and may involve physical destruction of the device as well as a complete reset of watchOS or Apple Pay service termination by the User.

When the device is received, the model of the device should be verified against those listed in Section 2.2. This can be accomplished using the Apple Watch app on the paired iPhone (tap the My Watch tab, then tap General > About and find the Model field).

2.4.4. TOE security features

The logical security features of the TOE are summarized as follows:

- User authentication and management
- Secure channel between the Secure Enclave and the Secure Element
- Secure channel between the Secure Enclave and the iPhone
- Card Data management
- Apple Pay payment transaction processing and management
- Operating System update
- iCloud logout and device reset

2.5. Description of the Apple Pay Service

This section contains a generic description of the Apple Pay service in general, which includes the TOE as well as the TOE environment and non-TOE hardware, software, and services.

2.5.1. Card provisioning

When a user adds a credit, debit, or prepaid card (including store cards) to Apple Wallet, the device encrypts the card information and securely sends it, along with other information about the user's account and device, through Apple Pay servers to the card issuer or the card issuer's authorized service provider (usually the payment network). Using this information, the card issuer (or its service provider) will determine whether to approve the user's request to add the card to Apple Wallet.

As part of the card provisioning process, Apple Pay uses three server-side calls to send and receive communication with the card issuer or payment network:

- Required Fields
- Check Card
- Link and Provision

The card issuer or payment network uses these calls to enable the card issuer to verify, approve, and add cards to Apple Wallet. The confidentiality and integrity of these client server sessions are protected using TLS 1.2 or later.

The full card numbers are never stored on the device or on Apple servers. Instead, a unique Device Account Number is created by the card issuer, sent encrypted to Apple, and then stored in the Secure Element. This unique Device Account Number is encrypted in such a way that Apple cannot access it. The Device Account Number is unique and different from most credit or debit card numbers; the card issuer or payment network can prevent its use on a magnetic stripe card, over the phone, or on websites. The Device Account Number located in the Secure Element is isolated from the TOE, is never stored on Apple servers, and never backed up to iCloud.

Cards for use with Apple Watch are provisioned for Apple Pay using the Apple Watch app on iPhone, or within a card issuer's iOS or watchOS app. Adding a card to Apple Watch requires that:

- *When paired with an iPhone:* The watch and iPhone are within Bluetooth communications range of each other
- *When set up without an iPhone:* The watch must have internet access using Wi-Fi or cellular

Cards are specifically enrolled for use with Apple Watch and have their own Device Account Numbers, which are stored within the Secure Element on the Apple Watch.

With watchOS 10 or later, when a user provisions an eligible payment card, the user can push provision the card to other Apple Pay-capable devices on the same iCloud account using the Multi-device provisioning feature. Nothing is copied from the original device; the other devices provision using the same flow they would use in Setup Assistant.

2.5.1.1. Adding credit or debit cards manually

To add a card manually, the name, card number, expiration date, and card verification value (CVV) are used to facilitate the provisioning process. From within Setup Assistant, Settings, Apple Wallet, or the Apple Watch app, users can enter the required information either by typing or by capturing it using the

device's camera. When the camera captures the card information, Apple Wallet attempts to populate the name, card number, and expiration date. The photo is never saved to the device or stored in the photo library. After all the fields are filled in, the Check Card process verifies the fields other than the CVV. They are then encrypted and sent to the Apple Pay server for routing to the card issuer or payment network.

If a terms and conditions ID is returned with the Check Card process, Apple downloads and displays the terms and conditions of the card issuer to the user. If the user accepts the issuer's terms and conditions, Apple sends the ID of the terms that were accepted as well as the CVV to the Link and Provision process.

2.5.1.2. Adding credit or debit cards from an iTunes Store account

For a credit or debit card on file with iTunes, the user may be required to reenter their Apple Account password. The card number is retrieved from iTunes and the Check Card process is initiated. If the card is eligible for Apple Pay, the Apple Wallet application downloads and displays terms and conditions of the card issuer, then sends along the terms and conditions ID and the card security code to the Link and Provision process. Additional verification may occur for iTunes account cards on file.

2.5.1.3. Adding credit or debit cards from a card issuer's app

When the app is registered for use with Apple Pay, keys are established for the app and for the card issuer's server. These keys are used to encrypt the card information that is sent to the card issuer. This is designed to prevent the information from being read by the Apple device. The provisioning flow is similar to that used for manually added cards, as described previously, except one-time passwords are used in lieu of the CVV.

2.5.1.4. Adding credit or debit cards from a card issuer's website

Some card issuers provide the ability to initiate the card provisioning process for Apple Wallet directly from their websites. In this case, the user initiates the task by selecting a card to provision on the card issuer's website. The user is then redirected to a self-contained Apple sign-in experience (contained within Apple's domain) and is asked to sign in with their Apple Account. Upon successfully signing in, the user then chooses one or more devices to provision the card to and is required to confirm the provisioning result on each respective target device.

2.5.1.5. Additional verification

A card issuer can decide whether a credit or debit card requires additional verification. Depending on what is offered by the card issuer, the user may be able to choose between different options for additional verification, such as a text message, email, a customer service call, or a method in the app of an approved card issuer to complete the verification. For text messages or email, the user is presented an option to select from contact information the issuer already holds on file. A code is sent, which must be entered into Apple Wallet, Settings, or the Apple Watch app. For customer service or verification using an app, the issuer performs their own communication process.

2.5.2. Payment authorization with Apple Pay

For devices having a Secure Element, a payment can be made only after it receives authorization from the Secure Enclave. For a payment to be made on Apple Watch, the device must be unlocked with passcode or paired iPhone and the side button must be double-clicked.

If wrist detection is enabled, the device locks automatically soon after it's removed from the user's wrist. With wrist detection enabled, the watch remains securely unlocked as long as it is on the user's wrist, allowing for payment authorization without re-entering the passcode. If wrist detection is disabled, Control Center provides an option for locking Apple Watch. When Apple Watch is locked, Apple Pay can be used only by entering the passcode on the Apple Watch.

2.5.2.1. Using a shared pairing key

Communication between the Secure Enclave and the Secure Element takes place over a serial interface, with the Secure Element connected to the Near Field Communication (NFC) controller, which in turn is connected to the Application Processor. Though not directly connected, the Secure Enclave and the Secure Element can communicate securely using a shared secret generated at runtime. In the factory, the Secure Enclave and Secure Element are securely provisioned with each other's long-term public keys. Secure Enclave's long-term public key is generated from its UID key and the Secure Element unique identifier. The corresponding private key is held in hardware and not visible to Secure Enclave software. At runtime, the long-term public keys are used to generate a shared secret using ECDH. The shared secret is then using to provide confidentiality and integrity over the communication link as needed.

2.5.2.2. Authorizing a secure transaction

When the user authorizes a transaction, which includes a physical gesture communicated directly to the Secure Enclave, the Secure Enclave sends signed data about the type of authentication and details about the type of transaction (contactless or e-commerce) to the Secure Element, tied to an Authorization Random (AR) value. The AR value is generated in the Secure Enclave when a user first provisions a credit card and persists while Apple Pay is enabled, protected by the Secure Enclave encryption and anti-rollback mechanism. It is securely delivered to the Secure Element by leveraging the pairing key. On receipt of a new AR value, the Secure Element marks any previously added cards as deleted.

2.5.2.3. Using a payment cryptogram for dynamic security

Payment transactions originating from the payment applets include a payment cryptogram along with a Device Account Number. This cryptogram, a one-time code, is computed using a transaction counter and a key. The transaction counter is incremented for each new transaction. The key is provisioned in the payment applet during personalization and is known by the payment network or the card issuer or both. Depending on the payment scheme, other data may also be used in the calculation, including:

- A Terminal Unpredictable Number, for near-field-communication (NFC) transactions
- An Apple Pay server nonce, for transactions within apps
- User verification results, such as Cardholder Verification Method (CVM) information

These security codes are provided to the payment network and to the card issuer, which allows the issuer to verify each transaction. The length of these security codes may vary based on the type of transaction.

2.5.3.Paying with cards using Apple Pay

Apple Pay can be used to pay for purchases in stores, within apps, and at websites.

2.5.3.1.Paying with cards in stores

If the device is on and detects an NFC field, it presents the user with the requested card (if automatic selection is turned on for that card) or the default card, which is managed in Settings. The user can also go to Apple Wallet and choose a card, or when the device is locked, can double-click the side button to select a card to be used.

Next, before payment information is transmitted, the user must authenticate. When Apple Watch is unlocked, double-clicking the side button activates the default card for payment. No payment information is sent without user authentication.

After the user authenticates, the Device Account Number and a transaction-specific dynamic security code are used when processing the payment. Neither Apple nor a user's device sends the full actual credit or debit card numbers to merchants. Apple may receive anonymous transaction information such as the approximate time and location of the transaction, which helps improve Apple Pay and other Apple products and services.

2.5.3.2.Paying with cards within apps

Apple Pay can also be used to make payments within watchOS apps. When users pay within apps using Apple Pay, Apple receives the encrypted transaction information to route to the specific developer or merchant to which the user is making a payment. Before that information is sent to the developer or merchant, Apple re-encrypts it with a developer-specific key. This is to help ensure that only an authorized developer with the key-pair can decrypt the information. Apple Pay retains anonymous transaction information such as approximate purchase amount. This information can't be tied to the user and never includes what the user is buying.

When an app initiates an Apple Pay payment transaction, the Apple Pay servers receive the encrypted transaction from the device prior to the merchant receiving it. The Apple Pay servers then re-encrypt the transaction with a merchant-specific key before relaying it to the merchant.

When an app requests a payment, it calls an API to determine whether the device supports Apple Pay and whether the user has credit or debit cards that can make payments on a payment network accepted by the merchant. The app requests any pieces of information it needs to process and fulfill the transaction, such as the billing and shipping address, and contact information. The app then asks watchOS to present the Apple Pay sheet, which requests information for the app, as well as other necessary information, such as the card to use.

At this time, the app is presented with city, state, and postal code information to calculate the final shipping cost. The full set of requested information isn't provided to the app until the user authorizes the payment by double-clicking the side button of the unlocked Apple Watch. After the payment is authorized, the information presented in the Apple Pay sheet is transferred to the merchant.

2.5.3.3. How users authorize, and merchants verify, app payments

Users and merchants ensure secure app payments by passing information to the Apple Pay servers, the Secure Element, the device, and the app's API. First, when the user authorizes an app payment, the app obtains a cryptographic anti-replay value, or nonce, by calling the Apple Pay servers. The servers send this value and other transaction data to the Secure Element to compute a payment credential that is encrypted with an Apple key. The Secure Element then returns the encrypted payment credential to the Apple Pay servers, which decrypt the credential, verify the nonce in the credential against the nonce that the Apple Pay servers originally sent, and re-encrypt the payment credential with the merchant key associated with the Merchant ID. The Apple Pay servers then return the payment to the device, which hands it back to the app through the API. The app then passes it along to the merchant system for processing. The merchant can then decrypt the payment credential with its private key. This, together with the signature from Apple's servers, allows the merchant to verify that the transaction was intended for this particular merchant, and ensures dynamic linking of the transaction with its amount and the payee.

The APIs require a Merchant ID Entitlement that specifies the supported Merchant IDs. An app can also include additional data (such as an order number or customer identity) to send to the Secure Element to be signed, ensuring that the transaction cannot be diverted to a different customer. This is accomplished by the app developer, who can specify applicationData on the PKPaymentRequest. A hash of this data is included in the encrypted payment data. The merchant is then responsible for verifying that their applicationData hash matches what's included in the payment data.

In order to process payments, the merchant takes the following steps:

- Send the payment information to their server, along with the other information needed to process the order.
- Verify the hashes and signature of the payment data.
- Decrypt the encrypted payment data, and confirm the validity of the transactionId, currencyCode, transactionAmount, and applicationData fields.
- Submit the payment data to the payment processing network and the order to their order-tracking system.

2.5.4. Rendering cards unusable with Apple Pay

Credit, debit, and prepaid cards added to the Secure Element can only be used if the Secure Element is presented with authorization using the same pairing key and Authorization Random (AR) value from when the card was added. On receipt of a new AR value, the Secure Element marks any previously added cards as terminated. This allows the operating system to instruct the Secure Enclave to render cards unusable by marking its copy of the AR as invalid under the following scenarios:

- The passcode is disabled
- The user signs out of iCloud
- The user selects Erase All Content and Settings
- The device is restored from Recovery Mode
- The device is unpaired from the paired iPhone

When a user erases the entire device—using Erase All Content and Settings, using Find My, or restoring their device—that device instructs the Secure Element to mark all cards as terminated. This has the effect

of immediately changing the cards to an unusable state until the Apple Pay servers can be contacted to fully erase the cards from the Secure Element. Independently, the Secure Enclave marks the Authorization Random as invalid so that further payment authorizations for previously enrolled cards are not possible. When the device is online, it attempts to contact the Apple Pay servers to help ensure that all cards in the Secure Element are erased.

2.5.4.1. Suspending, removing, and erasing cards

Users can suspend Apple Pay, or remove and erase their cards from the device using the following methods:

- Find My
- Apple Wallet
- Remove device from Apple Account
- Watch app on paired iPhone
- Remove device from icloud.com
- Delete Apple Account from the Apple Data and Privacy page

When a user suspends or removes cards from Apple Pay, the ability to make payments using those cards is suspended or revoked by the card issuer or respective payment network, even if the device is offline and not connected to a Wi-Fi or cellular network. Users can also contact their card issuer to suspend or remove cards from Apple Pay.

2.5.5. Apple Card

On supported devices, a user can securely apply for an Apple Card.

2.5.5.1. Apple Card application

On devices with iOS 12.4 or later, iPadOS 13.1 or later, mac OS 10.14.6 or later, watchOS 5.3 or later, and visionOS 1.0 or later, Apple Card can be used with Apple Pay to make payments in stores, in apps, and on the web. Apple Card is currently only available for qualifying applicants in the United States.

To apply for Apple Card, the user must be signed into their iCloud account on an Apple Pay–compatible iOS or iPadOS device and have two-factor authentication set up on the iCloud account, or they can apply at https://apply.applecard.apple/ after signing in with their Apple Account. When the application is approved, Apple Card is available in the Apple Wallet app or within Settings > Wallet & Apple Pay across any of the eligible devices the user has signed in with their Apple Account.

When a user applies for Apple Card, user identity information is securely verified by Apple's identity provider partners and then shared with Goldman Sachs Bank USA for the purposes of identity and credit evaluation.

Information such as the social security number or ID document image provided during the application is securely transmitted to Apple's identity provider partners and/or Goldman Sachs Bank USA encrypted with their respective keys. Apple can't decrypt this data.

The income information provided during the application, and the bank account information used for bill payments, are securely transmitted to Goldman Sachs Bank USA encrypted with their key. The bank account information is saved in Keychain. Apple can't decrypt this data.

When adding Apple Card to the Apple Wallet app, the same information as when a user adds a credit or debit card may be shared with the Apple partner bank, Goldman Sachs Bank USA, and with Apple Payments Inc. This information is used only for troubleshooting, fraud prevention, and regulatory purposes.

In iOS 14.6 or later, iPadOS 14.6 or later, and watchOS 7.5 or later, the organizer of an iCloud family with an Apple Card can share their card with their iCloud Family members over the age of 13. User authentication is required to confirm the invitation. Apple Wallet uses a key in the Secure Enclave to compute a signature that binds the owner and the invitee. That signature is validated on Apple servers.

Optionally, the organizer can set a transaction limit for the participants. Participant cards can also be locked to pause their spending at any time through the Apple Wallet app. When a co-owner or participant over the age of 18 accepts the invitation and applies, they go through the same application process in the Apple Wallet app as defined above.

2.5.5.2.Apple Card usage

A physical card can be ordered from Apple Card in Apple Wallet. After the user receives the physical card, it's activated using the NFC tag that's in the bifold envelope of the physical card. The tag is unique per card and can't be used to activate another user's card. Alternatively, the card can be manually activated in Apple Wallet settings. Additionally, the user can also choose to lock or unlock the physical card at any time from Apple Wallet.

2.5.5.3.Apple Card payments and Apple Wallet pass details

Payments due on the Apple Card account can be made from a web browser or the Apple Wallet app in iOS with Apple Cash and a bank account. Bill payments can be scheduled as recurring or as a one-time payment at a specific date with Apple Cash and a bank account. When a user makes a payment, a call is made to the Apple Pay servers to obtain a cryptographic nonce similar to Apple Cash. The nonce, along with the payment setup details, is passed to the Secure Element to compute a signature. The signature is then returned to the Apple Pay servers. The authentication, integrity, and correctness of the payment are verified through the signature and the nonce by Apple Pay servers, and the order is passed on to Goldman Sachs Bank USA for processing.

The Apple Card number is retrieved by Apple Wallet by presenting a certificate. The Apple Pay server validates the certificate to confirm the key was generated in the Secure Enclave. It then uses this key to encrypt the Apple Card number before returning it to Apple Wallet, so that only the iPhone that requested the Apple Card number can decrypt it. After decryption, the Apple Card number is saved in iCloud Keychain.

Displaying the Apple Card number details in the pass using the Apple Wallet app requires user authentication with a Biometric or a passcode. It can be replaced by the user in the card information section and disables the previous one.

2.5.5.4.Advanced Fraud Protection

In iOS 15 or later and iPadOS 15 or later, the Apple Card user can enable Advanced Fraud Protection in the Apple Wallet app. When enabled, the Card Security Code refreshes every few days.

2.5.6.Apple Cash

On devices with iOS 11.2 or later, iPadOS 13.1 or later, watchOS 4.2 or later, and visionOS 1.0 or later, Apple Cash can be used to send, receive, and request money from other users. When a user receives

money, it is added to an Apple Cash account that can be accessed in the Apple Wallet app or within Settings > Wallet & Apple Pay across any of the eligible devices the user has signed in with their Apple Account. Apple Cash is currently only available to users in the United States.

On devices with OS 14 or later, iPadOS 14 or later, and watchOS 7 or later, the organizer of an iCloud family who has verified their identity with Apple Cash can enable Apple Cash for their family members under the age of 18. Optionally, the organizer can restrict the money sending capabilities of these users to family members only or contacts only. If the family member under the age of 18 goes through an Apple Account recovery, the organizer of the family must manually reenable the Apple Cash card for that user. If the family member under the age of 18 is no longer part of the iCloud family, their Apple Cash balance is automatically transferred to the organizer's account.

When the user sets up Apple Cash, the same information as when the user adds a credit or debit card may be shared with Apple's partner bank in the United States, Green Dot Bank, and with Apple Payments Inc., a wholly owned subsidiary created to protect the user's privacy by storing and processing information separately from the rest of Apple, and in a way that the rest of Apple doesn't know. This information is used only for troubleshooting, fraud prevention, and regulatory purposes.

2.5.6.1. Using Apple Cash in iMessage

To use person-to-person payments and Apple Cash, a user must be signed into their iCloud account on an Apple Cash compatible device and have two-factor authentication set up on the iCloud account. Money requests and transfers between users are initiated from within the Messages app or by asking Siri. When a user attempts to send money, iMessage displays the Apple Pay sheet. The Apple Cash balance is always used first. If necessary, additional funds are drawn from a second credit or debit card the user has added to the Apple Wallet app.

2.5.6.2. Using Tap to Cash

When using Tap to Cash, a user must be signed in to their iCloud account on the device and have two-factor authentication set up on the iCloud account. To send money, customers must enter the amount to send, hold their device near another device, and authenticate by double-clicking the side button of the unlocked Apple Watch.

After authentication, devices must be held together for several seconds to establish a connection that results in the money being sent.

2.5.6.3. Using Apple Cash in stores, apps, and on the web

The Apple Cash card in the Apple Wallet app can be used with Apple Pay to make payments in stores, in apps, and on the web. Money in the Apple Cash account can also be transferred to a bank account. In addition to money being received from another user, money can be added to the Apple Cash account from a debit or prepaid card in the Apple Wallet app.

Apple Payments Inc. stores, and may use, the user's transaction data for troubleshooting, fraud prevention, and regulatory purposes once a transaction is completed. The rest of Apple doesn't know who the user sent money to, received money from, or where the user made a purchase with their Apple Cash card.

When the user sends money with Apple Pay, adds money to an Apple Cash account, or transfers money to a bank account, a call is made to the Apple Pay servers to obtain a cryptographic nonce, which is similar to the value returned for Apple Pay within apps. The nonce, along with other transaction data, is passed to the Secure Element to compute a payment signature. The signature is returned to the Apple

Pay servers. The authentication, integrity, and correctness of the transaction is verified through the payment signature and the nonce by Apple Pay servers. Money transfer is then initiated, and the user is notified of a completed transaction.

If the transaction involves:

- A debit card for adding money to Apple Cash
- Providing supplemental money if the Apple Cash balance is insufficient

An encrypted payment credential is also produced and sent to Apple Pay servers, similar to how Apple Pay works within apps and websites.

After the balance of the Apple Cash account exceeds a certain amount or if unusual activity is detected, the user is prompted to verify their identity. Information provided to verify the user's identity - such as social security number or answers to questions (for example, to confirm a street name the user lived on previously) - is securely transmitted to the Apple partner and encrypted using their key. Apple can't decrypt this data. The user is prompted to verify their identity again if they perform an Apple Account recovery, before regaining access to their Apple Cash balance.

2.5.7. Credit and debit cards for transit

In some cities, transit readers accept EMV (smart) cards to pay for transit rides. When users present an EMV credit or debit card to those readers, user authentication is required just as with "Apple Pay can be used to pay for purchases in stores, within apps, and at websites.

Paying with cards in stores."

On devices with watchOS 5.2.1 or later, some existing EMV credit or debit cards in the Apple Wallet app can be enabled for Express Transit. Express Transit allows the user to pay for a trip at supported transit operators without requiring a passcode. When a user provisions an EMV credit or debit card, the first card provisioned to the Apple Wallet app is enabled for Express Transit.

To disable Express Transit, the user can open the Apple Watch App on their iPhone and tap the My Watch tab, scroll down and tap Wallet & Apple Pay, tap Express Transit Card, then tap None. The user can also select a different credit or debit card as their Express Transit card via the Watch App. Passcode authentication is required to re-enable or select a different card for Express Transit.

Note: Apple Card and Apple Cash are eligible for Express Transit.

Payments with credit and debit cards for Express Transit are not in the scope of this Security Target.

2.6. TOE Use Cases

The TOE covers the following use cases:

Use Case	Description
UC.Device_Usage	Device usage The user can manage the device's authentication credentials, including changing the passcode.

Use Case	Description
UC.OS_Update	Device Software Update The User can perform an update of the software in the device to a new version. This use case requires that the user verifies the device's passcode. This use case ensures preservation of the User settings on the device: - No change to the User's authentication credentials (passcode) - No change to the User's data within the Secure Element unless specified by the data's issuer
UC.Apple_Pay_Install_Init	Apple Pay installation and initialization The User can provision a new card in Apple Wallet.
UC.Apple_Pay_Usage	Apple Pay usage The User can perform Apple Pay transactions.
UC.End_Of_Service	Termination by User The User can end the Apple Pay mode of operation by performing a card removal in Apple Wallet. The User can also end all current Apple Pay services by un-registering their iCloud account. Termination by card issuer The card issuer can perform a de-registration of an Apple Pay card that was provisioned on the User's device, following a card revocation or a user account termination.
UC.End_Of_Life	Termination of device The User can clear a device from all their settings and data by performing a device full reset. The User could also end the life of their device by physically destroying it.

3. Evaluation Assurance

3.1. Common Criteria Reference

This Security Target is based on the following Common Criteria ™ (CC) publications:

Common Criteria	CC Version	Revision	Date
Part 1: Introduction and general model	CC:2022	R1	November 2022
Part 2: Security functional requirements	CC:2022	R1	November 2022
Part 3: Security assurance requirements	CC:2022	R1	November 2022
Part 4: Framework for the specification of evaluation methods and activities	CC:2022	R1	November 2022
Part 5: Pre-defined packages of security requirements	CC:2022	R1	November 2022

3.2. CC Conformance claim

This Security Target is **conformant** to CC Part 2 and CC Part 3.

3.3. Protection Profile Conformance claim

This Security Target does not claim any conformance to an existing Protection Profile.

3.4. Assurance Level

The evaluation assurance level (EAL) for this work is EAL 2 augmented with ADV_FSP.3 and ALC_FLR.3:

Assurance Class	
ADV: Development	ADV_ARC.1 Security architecture description
	ADV_FSP.3 Functional specification with complete summary
	ADV_TDS.1 Basic design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.2 Use of a CM system
	ALC_CMS.2 Parts of the TOE CM coverage
	ALC_DEL.1 Delivery procedures
	ALC_FLR.3 Systematic flaw remediation
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction
	ASE_OBJ.2 Security objectives
	ASE_REQ.2 Derived security requirements

Assurance Class	
	ASE_SPD.1 Security problem definition
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.1 Evidence of coverage
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

4. Security Problem Definition

4.1. Assets

The assets of the TOE are:

Asset	Sensitivity*	Type	Definition
D.User_Passcode	I, C	User	Passcode value setup by the User, used to wake up the device after power loss, unlock the device, and to authorize payments (Apple Pay transactions or Apple Cash transfers).
D.Card_Data	I, C	User, TSF	Apple Pay card data, including credit/debit card number, User's name, expiration date, CVV, and transaction data (e.g., transaction history). Apple Cash card data, including Apple Cash card information and transfer data (e.g., transfer history). Note: The card data is used as TSF data when the TOE sends it encrypted to the card issuer, via Apple servers, in order to generate the Device Account Number stored in the Secure Element.
D.User_Intent	I	User	State of the device resulting from a physical interaction of the User with the TOE, characterizing the intent of the User to perform a transaction (Apple Pay transactions or Apple Cash transfers): double click on the side button of an unlocked Watch.
D.Payment_Data	I	User	Apple Pay/Apple Cash payment data being authorized by the User. For Apple Pay, critical elements are the amount (including the currency), the emitter, and the recipient (S.Merchant) which constitute the core of the Dynamic Linking. For Apple Cash, critical elements of transfers are the amount (including the currency), the emitter, and the recipient. Additional critical elements can be defined by Apple and the card issuer.
D.OS	I	TSF	OS version currently installed on the device. This asset is not the OS code itself or the version number of that code. This asset is the set of elements that compose an OS version as prepared for that device, including elements verifiable by that device during boot (for instance the sepOS), and by the User through an identified version number.
D.User_Configuration	I	TSF	User configuration is a representation of the user personal configuration including Apple Pay, Apple Cash settings
D.SEP_Configuration	I	TSF	Secure Enclave configuration is a representation of the state of the Secure Enclave in the device. It comprises (but is not limited to) the Secure Enclave OS version and the state of the user authentication functions (passcode, authenticated credentials, retry counters, and authentication based access control settings).
D.SEP_SE	I, C	TSF	Secret data that allows secure communication between the Secure Enclave and the Secure Element during processing of an Apple Pay transaction or Apple Cash transfer.

Asset	Sensitivity*	Type	Definition
D.SEP_iPhone	I, C	TSF	Secret data that allows secure communication between the Secure Enclave on the TOE and the Secure Enclave on the Paired iPhone. The Paired iPhone is used to manage the Watch content (OS version, Wallet cards and configuration, Apps installed, and critical settings like Wrist Detection). The pairing secret allows the TOE and the Paired iPhone to exchange sensitive data like the unlock secret.
D.Unlock_Secret	I,C	TSF	Secret data that is shared by the Secure Enclave on the TOE with the Paired iPhone Secure Enclave to unlock the TOE. It is released by the paired iPhone to facilitate the TOE unlock. This asset is created by the TOE when the Unlock with iPhone feature is enabled by the User and does not exist if this feature is disabled.

*: I = Integrity, C = Confidentiality

4.2. Subjects

The subjects of this Security Target are:

Subject	Definition
S.User	User of Apple Pay/Apple Cash on the device, able to: • Authenticate on the device (passcode) • Manage authentication credentials (passcode) • Manage device configuration (watchOS version, iCloud account, access policies) • Provision/enroll cards • Authorize Apple Pay transactions/Apple Cash transfers by providing consent for the transaction to proceed (watch unlocked and user intent) • Cancel the Apple Pay/Apple Cash service
S.Apple_Servers	Apple servers in charge of: • Management of S.User iCloud account • Management of S.User provisioning/enrollment in Apple Pay/Apple Cash • Management of watchOS releases, including Apple Wallet app • Device's interface for processing Apple Pay transactions/Apple Cash transfers (contact S.Issuer)
S.Issuer	The card issuer (or its service provider) is the third party in charge of: • Management of S.User data for Apple Pay/Apple Cash services • Processing Apple Pay transactions/ Apple Cash transfers
S.Merchant	The merchant is the third-party accepting payment through an Apple Pay transaction.
S.SE	Certified Secure Element of the device including the TOE.
S.iPhone	• The Paired iPhone is also owned by S.User and is used to: • Setup the device • Store the Unlock Secret for the device when Unlock with iPhone feature is enabled • Manage device content (display on iPhone, operation on Apple Watch)

4.3. Assumptions

The assumptions of this Security Target are the following:

Assumptions	Definition
A.DEVICE_AUTH	The User of the device ensures that all the Apple Pay and Apple Cash activities that are performed on the device have been authorized by the user. All authentication credentials (passcode) for the device that are enabled for use with Apple Pay/Apple Cash are owned and protected by the User of the device.
A.PERSO	The Apple Pay and Apple Cash card issuers guarantee the correctness of the card data input in the device during provisioning/enrollment: Data shall uniquely identify a financial payment means and be linked to the account owned by the identified user.
A.CDCVM	The Secure Element and applets hosted on the Secure Element manage the CDCVM requirement for NFC payment transactions and require the TOE to provide user authentication and intent to pay each time CDCVM is needed. By default, applets configured for express mode do not require user authorization for transit use.
A.IPHONE_USER	The User owns the iPhone paired to the Apple Watch and ensures the confidentiality of its authentication credentials.

4.4. Threat Agents

The threat agents of this TOE are the following:

Threat Agent	Definition
S.Attacker	A threat agent trying to interact with the Apple Pay and Apple Cash system fraudulently, trying to modify the configuration and data of genuine Users' devices or forging data on their own device.

4.5. Threats

The threats to assets of this TOE are the following:

Threat	Name	Definition	Assets
T.CORRUPT	Corrupted Transaction or Transfer	An attacker attempts to corrupt an Apple Pay transaction or an Apple Cash transfer. To gain from the attack, the attacker could be the emitter of the transaction or transfer, and attempt to reduce what it intends to pay (debit amount from attacker's account) from what it was supposed to pay (credit amount request or transfer amount agreed between attacker and victim). The attacker could also attempt to corrupt a transaction or transfer when it is the recipient, and increase what it supposed to receive (credit transaction amount or transfer credit amount) from what it was supposed to receive (debit amount agreed). The attacker could also attempt to modify the recipient of a credit transaction by changing the payee in an Apple Pay transaction or changing the recipient of an Apple Cash transfer.	D.Payment_Data D.User_Configuration D.OS
T.PHYSICAL	Physical	The loss or theft of the Mobile Device may give rise to loss of confidentiality of User data including credentials. These physical access threats may involve attacks which attempt to access the device through external hardware ports, through its user interface, and also through direct and possibly destructive access to its storage media. The goal of such attacks is to access Apple Pay or Apple Cash data from a lost or stolen device which is not expected to return to its User. Note: *Defending against device re-use after physical compromise is out of scope.*	D.User_Passcode D.Card_Data D.Unlock_Secret* D.SEP_iPhone*
T.RECOVER	Card Recovery	An attacker attempts to recover Apple Pay or Apple Cash card data from an erased or blocked device and use it to perform a financial transaction or transfer. The attacker will target potential breaches in the process of Apple Pay cancellation, Apple Pay card revocation, Apple Cash disenrollment, Apple Cash card revocation, or watchOS erase all content and settings.	D.User_Configuration D.Card_Data D.OS
T.REPLAY	Replay	An attacker attempts to replay an Apple Pay transaction or an Apple Cash transfer.	D.Payment_Data
T.SILENT	Silent Transaction	An attacker attempts to modify the behavior of the device, in order to perform silent Apple Pay transactions or Apple Cash transfers for some benefit. The attacker would have to perform the attack without knowledge of the device's rightful owner who will be the victim of the attack.	D.User_Intent D.User_Configuration D.SEP_Configuration D.SEP_SE D.OS

T.SKIMMING	Authentication Bypass	An attacker attempts to perform a payment with Apple Pay or Apple Cash, bypassing the required authentication step (unlock with passcode or *S.iPhone*).	D.User_Intent D.Payment_Data D.SEP_Configuration D.User_Configuration D.OS
T.USURP	Card Ownership Usurpation	An attacker could attempt to authenticate on a device, with a goal of using any provisioned cards on that device. The attacker could focus on the card data during Apple Pay provisioning or Apple Cash enrollment.	D.User_Passcode D.User_Configuration D.Card_Data D.OS D.Unlock_Secret D.SEP_iPhone

* Partial threat: By unlocking the device, the attacker could have a higher chance to access Apple Pay data.

Assets & Threats mapping table:

Asset - Property - I = Integrity C = Confidentiality		**T.CORRUPT**	**T.PHYSICAL**	**T.RECOVER**	**T.REPLAY**	**T.SKIMMING**	**T.SILENT**	**T.USURP**
D.Unlock_Secret	I,C		X					X
D.User_Passcode	I,C		X					X
D.User_Intent	I					X	X	
D.Payment_Data	I	X			X	X		
D.Card_Data	I,C		X	X				X
D.OS	I	X		X		X	X	X
D.User_Configuration	I	X		X		X	X	X
D.SEP_Configuration	I					X	X	
D.SEP_iPhone	I,C		X					X
D.SEP_SE	I,C						X	

4.6. Organizational Security Policies

The organizations associated with the Apple Pay service shall comply with the following Organizational Security Policies as security rules, procedures, practices, or guidelines imposed by an organization upon its operations:

OSP	Definition
P.UPDATE	Apple ensures that only an authenticated User (*S.User*) can update their device's operating system to a newly released watchOS. Apple also informs the Apple Pay and Apple Cash card issuers and PNOs of new applicable features of new releases.
P.DYN_LINK	Apple maintains the enforcement of Dynamic Linking for e-Commerce payments using Apple Pay or Apple Cash cards, on device side and server side. This Organizational Security Policy guarantees that Apple preserves the following properties from design to feature release for Apple Pay and Apple Cash e-Commerce payments: (a) The payer is made aware of the amount of the payment transaction and of the payee; (b) The authentication code generated is specific to the amount of the payment transaction and the payee agreed to by the payer when initiating the transaction; (c) The authentication code accepted by the payment service provider corresponds to the original specific amount of the payment transaction and to the identity of the payee agreed to by the payer; (d) Any change to the amount or the payee results in the invalidation of the authentication code generated
P.IPHONE	The TOE allows the User to set the device to unlock automatically when the user unlock the *S.iPhone*. When the feature "Unlock Apple Watch when you unlock your iPhone" is enabled, the TOE can be unlocked by *S.iPhone* if the iPhone is unlocked.

5. Security Objectives

5.1. Security Objectives for the TOE

Security Objectives	Definition
OT.User_Auth	The TOE enforces the following authentication policy: • Passcode: – Update passcode – Update the OS to a new version signed by Apple – Setup Unlock with iPhone – Unlock of the device – Authorize Apple Pay transactions/Apple Cash transfers (if the Wrist detection feature is disabled) • Paired iPhone (*S.iPhone*): – Unlock of the device – Modify security attributes of the Apple Watch (requires user authentication)
OT.Card_Data	The TOE enforces that sensitive card data: • Is not accessible after sent to the Apple server (by *S.iPhone*).
OT.Passcode_Delete	The TOE enforces that removing the passcode: • Disables Apple Pay/Cash.
OT.Card_Delete	The TOE securely triggers the deletion of each individual Apple Pay/Apple Cash card when: • A card is removed from Apple Wallet • The TOE handles the revocation of a card by its card issuer, • The use of Apple Pay is disabled (from iCloud, the Settings or passcode removal) • The iCloud account is no longer associated with the device. When a card is removed, the TOE also instructs the Secure Element to mark it as deleted.
OT.Auth_SE	The TOE provides the Secure Element with passcode authentication features for Apple Pay/Apple Cash payment/transfer approval (if the Wrist detection feature is disabled).
OT.Payment	For eCommerce transactions and Apple Cash transfers, the TOE enforces that transaction details are displayed to the User (including the card to be used from the Apple Wallet, the amount, and the payee) before the User shows their intent to pay. The TOE ensures that these details cannot be corrupted between the payment validation and the moment when details are sent to the Secure Element. For NFC transactions, if Apple Watch is unlocked, the User can manually select the card by going to Apple Wallet and selecting it in the UI. Otherwise, the user can double-click the side-button to pre-arm Apple Pay, which brings up the default card, and then the User can switch to a different card. The TOE also requests explicit intent from the User for Apple Pay transactions and Apple Cash transfers: double click on the side button of the device.

Security Objectives	Definition
OT.Device_Reset	TOE securely deletes all User data when the User launches an "Erase All Content and Settings" on the OS or a device erase from iCloud. This also initiates the disabling of Apple Pay/Apple Cash and plans the destruction of Apple Pay/Apple Cash cards that will be effective when the device is restored.
OT.Anti_Replay	The TOE ensures that each payment processed by an Apple Pay/Apple Cash card holds the unique identifier.
OT.OS_Update	TOE enforces security measures ensuring preservation of User data when an OS update is installed in the TOE. This objective protects the user authentication credentials (passcode), the Apple Pay card data, the Apple Cash card, and more.
OT.iPhone	The TOE can enforce secured communication with an optional paired iPhone (*S.iPhone*).

5.2. Security Objectives for the environment

Environment Security Objectives	Definition
OE.Card_Data	The card issuer is responsible for using the appropriate security measures to protect the confidentiality and the integrity of the sensitive card data and guaranteeing the authenticity of the card during enrolment. The Secure Element is responsible for securing the card validation exchanges with the card issuer's TSM and for ensuring confidentiality and integrity of each card's sensitive data during storage and use.
OE.Perso	The card issuer is responsible for verifying that the User is authorized to perform a transaction on the account of the card used as a reference, before allowing the card personalization. The card issuer also ensures the robustness of the personalization data, to prevent attacks like forgery, counterfeit, or corruption.
OE.Card_Delete	The issuers of all payment cards provisioned on a device are informed after the User removes a card from that device, removes that device from the iCloud account or performs a device Erase All Content and Settings. The card issuers ensure these cards are removed from the User's account (i.e., the unlinking process of the DPAN from the FPAN, which is done by the card issuer or the corresponding TSP). The Secure Element is responsible for securely deleting the stored card sensitive data (private/secret).
OE.Anti_Replay	The Apple Pay server verifies that each payment (e-Commerce Apple Pay transaction or Apple Cash transfer) is not replayed. The payment is invalidated if this verification fails. For in-store transactions (i.e. NFC transactions), a similar anti-replay mechanism is used with the participation of the NFC terminal.
OE.Transaction_Verification	For Apple Pay, the cryptogram released by the Secure Element for an Apple Pay transaction is verified by the card issuer (or its service provider). The cryptogram validation result allows the card issuer to approve or reject the transaction. The payment is invalidated if this verification fails.

	For Apple Cash, the Apple Cash server ensures that no Apple Cash transfer can be executed if the submitted quote (received by the server before the User approves) does not match the transaction data (received by the server once device completes transfer processing). The modifications that the server is able to detect cover but are not limited to, the amount and the recipient.
OE.Dynamic_Linking	For eCommerce transactions, the Apple Pay server preserves and the card issuer server verifies the cryptographic based dynamic linking of the transaction data (including amount and payee). The payment is invalidated if this verification fails.
OE.Statement	For Apple Pay, the card issuers ensure that the statement associated to the card (list of transactions) is fully accurate and includes, but is not restricted to, the amount and recipient of each transaction. For Apple Cash, the card issuer ensures that the ledger associated to an Apple Cash account (list of transfers including completed, canceled, and pending) is fully accurate.
OE.Genuine_Wallet	The Apple Wallet application is provided and signed by Apple.
OE.CDCVM	Express mode compatible applets hosted on the Secure Element are responsible for checking the CDCVM state, including it within transaction data where required, and allowing normal or Express transit transactions as applicable. Payment networks or card issuers are responsible for ensuring that Express transactions can only be accepted for transit-specific use by requiring that non-transit Apple Pay payment transactions have a successful CDCVM.
OE.iPhone	The S.iPhone is responsible for ensuring the confidentiality of the unlock secret provided by the watch during all its lifetime: from inception at enabling of the “Unlock with iPhone” feature, during its storage, during its release for unlocking the watch and when it is deleted when the feature is disabled. The S.iPhone is responsible for ensuring that it is protected by a passcode and not allowing someone else to enroll their own biometric templates.
OE.User	The S.User is responsible for ensuring that: • They authorize all the Apple Pay/Cash activities that are performed on the device • The passcode is robust and protected • Only their own iPhone is paired with the TOE and the paired iPhone (S.iPhone) is protected. This includes abiding by the iOS Software License Agreement and protecting the confidentiality of the paired iPhone’s passcode

5.3. Rationale of the Security objectives for the security problem definition

The following table details the rationale for each element of the security problem definition. For all the objectives for the TOE, OT.OS_Update also ensures the authentication configuration is preserved during the OS update.

Element	Rationale
A.DEVICE_AUTH	OE.User and OE.iPhone cover A.DEVICE_AUTH ensuring the User owns and protects all the authentication credentials and the User authorizes the Apple Pay and Apple Cash activities that are performed on the device.
A.PERSO	OE.Perso covers A.PERSO ensuring the provisioning process verifies the ownership of the provisioned cards.
A.CDCVM	OE.CDCVM covers A.CDCVM ensuring the Secure Element uses the TSF features when required.
A.IPHONE_USER	OE.User and OE.iPhone covers A.IPHONE_USER ensuring the User of the device owns and protects their iPhone and related credentials.
T.SKIMMING	OT.User_Auth, OT.Auth_SE and OT.Payment ensure that an Apple Pay transaction or an Apple Cash transfer is always authenticated and authorized by the User. OT.User_Auth and OE.Genuine_Wallet ensure that the Apple release of a new OS or Apple Wallet app implementing the authentication functions and payment functions are controlled, and that it preserves the confidentiality and integrity of the authentication and payment data. OT.Passcode_Delete ensures that if the passcode is turned off, the Apple Pay authentication is not available anymore.
T.USURP	OT.Card_Data, OT.OS_Update and OE.Card_Data ensure that the Apple Pay/Apple Cash card data are kept confidential and not altered from an attacker during storage and use in the Secure Element. OT.User_Auth, OT.Auth_SE and OT.Payment prevent the attacker from attempting to perform Apple Pay transactions or Apple Cash transfers, enforcing user authentication with set credentials (which are not known or owned by the attacker according to OE.User) and preventing the attacker from replacing a User's passcode with their own. OT.User_Auth and OE.Genuine_Wallet additionally ensure that the installation of a new Apple released OS or Apple Wallet app preserves the confidentiality and integrity of the payment data. OT.Passcode_Delete ensures that if passcode login is disabled, the Apple Pay is not available anymore.
T.RECOVER	OT.Card_Data, OT.OS_Update, and OE.Card_Data ensure that the confidential card data is only stored by the Secure Element which protects them from disclosure. OT.Device_Reset covers the physical erase of the content and settings of the device where the TOE will trigger secure erase all provisioned card data. OE.Card_Delete ensures that the issuers of provisioned cards are securely removing the deleted cards from the User's account so that no transaction can further proceed.
T.REPLAY	OE.Anti_Replay and OT.Anti_Replay ensure that each Apple Pay transaction or Apple Cash transfer is uniquely identified and cannot be replayed. OT.User_Auth and OE.Genuine_Wallet additionally ensure that the installation of a new Apple released OS or Apple Wallet application preserves transaction replay protection.
T.CORRUPT	OT.Payment enforces the dynamic linking of the Apple Pay transaction data, ensuring that the critical content (such as emitter, recipient, amount) cannot be changed after the transaction was processed using a provisioned card. OE.Dynamic_Linking ensures that the Apple Pay server is verifying the integrity of the Apple Pay transaction data Dynamic Linking. OE.Statement provides an additional verification point for the account holder as the card issuer ensures that all processed Apple Pay transactions appear on the statement of the account associated to the card.

	OT.User_Auth and OE.Genuine_Wallet ensure that the Apple release of a new OS or Apple Wallet application implementing the payment functions are controlled.
T.SILENT	OT.User_Auth and OT.Payment ensure that an Apple Pay transaction or Apple Cash transfer is always authorized by the User. OE.Statement ensures that the Apple Pay card issuers provide account holder verification material (in the form of transaction statements) allowing them to identify any fraudulent activity on their account. The Apple Cash card issuer ensures that the ledger associated to an Apple Cash account (list of transfers including completed/canceled/pending) is fully accurate. OT.User_Auth and OE.Genuine_Wallet ensure that the Apple release of a new OS or Apple Wallet application implementing the authentication functions and payment functions is controlled. OT.Passcode_Delete ensures that if passcode is turned off, the Apple Pay authentication is not available anymore.
T.PHYSICAL	The User credentials maintained by the TOE are secured by OT.User_Auth enforcing the secure verification process of passcode. The TOE lifecycle is secure through the management of the device within the Apple iCloud environment where the User is able to remove the device from its account (OT.Card_Delete) and reset the device's content (OT.Device_Reset). This binding ensures that the User's critical data is safe in case device is lost or stolen. OT.Card_Data, OT.OS_Update and OE.Card_Data ensure that the provisioned card data is kept confidential in the Secure Element and cannot be extracted.
P.UPDATE	OT.User_Auth and OE.Genuine_Wallet ensure that the Apple release of a new OS or Apple Wallet application implementing the authentication functions and payment functions are controlled, and that it preserves the confidentiality and integrity of the authentication and payment data.
P.DYN_LINK (Dynamic Linking)	P.DYN_LINK is covered by OT.Payment, OT.User_Auth, OT.Anti_Replay, OE.Anti_Replay, OE.Transaction_Verification, and OE.Dynamic_Linking, which all participate in the enforcement of the Dynamic Linking requirements on e-Commerce payments. The mapping is as follows: **(a)** OT.Payment ensures that there is a step, part of the user intent confirmation phase, when the User (payer) is made aware of the amount of the payment transaction and payee. **(b)** OT.User_Auth ensures that the user authentication was performed to ensure agreement by the payer to authorize the Apple Pay transaction data (specific to the payer, amount and payee), OT.Anti_Replay ensures that each payment is uniquely identified, and OT.Payment ensures that the payment data is integrity protected. **(c)** OE.Anti_Replay, OE.Transaction_Verification and OE.Dynamic_Linking ensure that the received Apple Pay transaction data correspond to what was agreed to by the payer: The unique identifier to prevent replay is verified, the cryptogram for the data is verified, and the dynamic linking of the user authentication and the payment data integrity is verified. **(d)** OE.Anti_Replay, OE.Transaction_Verification and OE.Dynamic_Linking ensure that any change to the amount or the payee results in the invalidation of the payment and its unique identifier so no replay is attempted.
P.IPHONE	P.IPHONE is covered by OT.User_Auth, OT.iPhone, OE.iPhone, and OE.User which ensure that only the user's iPhone can be used for the dedicated features.

Security Objectives mapping table:

	T.CORRUPT	T.PHYSICAL	T.RECOVER	T.REPLAY	T.SILENT	T.SKIMMING	T.USURP	P.DYN_LINK	P.UPDATE	P.IPHONE	A.PERSO	A.DEVICE_AUTH	A.CDCVM	A.IPHONE_USER
OT.Anti_Replay				x				x						
OT.Card_Data		x	x				x							
OT.Payment	x				x	x	x	x						
OT.Card_Delete		x												
OT.OS_Update		x	x				x							
OT.Auth_SE						x	x							
OT.User_Auth	x	x		x	x	x	x	x	x	x				
OT.iPhone										x				
OT.Passcode_Delete					x	x	x							
OT.Device_Reset		x	x											
OE.Anti_Replay				x				x						
OE.Card_Data		x	x				x							
OE.Perso											x			
OE.Dynamic_Linking	x							x						
OE.Statement	x				x									
OE.Transaction_Verification								x						
OE.CDCVM													x	
OE.iPhone										x		x		x
OE.User							x			x		x		x
OE.Genuine_Wallet	x			x	x	x	x		x					
OE.Card_Delete			x											

6. Security Functional Requirements

6.1. SFR supporting definitions

6.1.1. Security Functional Policies (SFP)

Access Control SFPs are given in the table below:

Authentica- tion_SFP	Authentication policy enforcing authentication, re-authentication and authorization rules as defined by OT.User_Auth. This SFP includes information flows between the TOE and S.iPhone (for device unlock).
Payment_SFP	Security policy enforcing that processing a payment requires the User to confirm the intent to pay to allow processing of the related data and its exportation.
Card_Perso_SFP	Security policy enforcing that: • Importing D.Card_Data is only allowed if: a passcode is configured • Confidential parts of the card data are protected before being sent to the Apple servers • Confidential parts of the card data are not imported in Apple Wallet

6.1.2. Subjects and Objects

Objects are the Assets identified in Section 4.1.
Subjects are listed in Sections 4.2 and 4.4.

6.1.3. Security Attributes

Security Attribute	Definition	Possible values
Card Data Confidential parts	Parts of the Card Data that should stay confidential and not been stored in Apple Wallet	- Secret parts of the card number - CVV - ...
User Authorization	Part of D.Payment_Data specifying the explicit authorization of the S.User	- "yes" - "no"
Wrist Detection	Part of D.User_Configuration which, when enabled, indicates that the watch automatically locks when removed from the wrist by the User.	- "enabled" (default) - "disabled"
Erase Data	Part of D.User_Configuration which, when enabled, erases the data on the device after 10 consecutive attempts to unlock it using the wrong passwords.	- "enabled" - "disabled" (default)
iPhone Unlock	Authorization to use S.iPhone to unlock the TOE.	- "enabled" - "disabled" (default)
Apple OS public key	The public key used to check the authenticity of a new version of D.OS.	- Part of installed D.OS
OS_signature	Part of the OS file that is checked by the TOE before updating D.OS	- Part of the update file

Passcode_off	Part of D.SEP_Configuration, configuration of the OS allowing to use the device without any authentication. When enabled, the TSF should disable Apple Pay.	- "disabled" (default) - "enable"

6.1.4. Writing conventions for the SFR operations

Iterations are identified by a slash character "/" followed by the name of the iteration.

Assignments and selections are done with *italicized text*.

Refinements are identified with the prefix "Refinement:".

6.2. Identification and authentication

6.2.1. User authentication

FIA_UID.2 User identification before any action

FIA_UID.2.1	The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

FIA_UAU.2 User authentication before any action

FIA_UAU.2.1	The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Note: this gives S.User the role of "Authenticated User". According to this requirement, S.User is authenticated by the TSF before performing any of the operations listed in the following requirements.

FIA_UAU.5 Multiple authentication mechanisms

FIA_UAU.5.1	The TSF shall provide *passcode authentication, iPhone unlock (S.iPhone)* to support user authentication.
FIA_UAU.5.2	The TSF shall authenticate any user's claimed identity according to the: • *Passcode authentication as default authentication* • *iPhone unlock (if iPhone Unlock = enabled)* • *Rules defined in FIA_UAU.6 Re-authenticating and FIA_AFL.1/Erase, FIA_AFL.1/Delay Authentication failure handling*.

FIA_AFL.1/Erase Authentication failure handling

FIA_AFL.1.1/Erase	The TSF shall detect when *10 (ten)* unsuccessful authentication attempts occur related to *passcode validation*.
FIA_AFL.1.2/Erase	When the defined number of unsuccessful authentication attempts has been *met*, the TSF shall *erase data on Apple Watch if Erase Data = enable*

FIA_AFL.1/Delay Authentication failure handling

FIA_AFL.1.1/Delay	The TSF shall detect when *3 (three)* unsuccessful authentication attempts occur related to *passcode validation*.

FIA_AFL.1.2/ Delay	When the defined number of unsuccessful authentication attempts has been *met*, the TSF shall *start delaying further passcode validation attempts and require passcode validation.*

FIA_UAU.6 Re-authenticating

FIA_UAU.6.1	The TSF shall re-authenticate the user under the conditions *that the user requests:* • *Device unlock: with Wrist Detection feature "Enabled", the removal of the Apple Watch from the wrist triggers a lock.* • *Payment authorization: with Wrist Detection feature "Disabled", the user needs to provide the passcode to authorize the payment.* • *Lowering of security attributes: "Passcode_off", "Wrist detection", "iPhone Unlock"*

6.2.2. Data Authentication

FDP_DAU.1 Basic Data Authentication

FDP_DAU.1.1	The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of *D.Payment_Data (including S.Merchant and S.User data)*.
FDP_DAU.1.2	The TSF shall provide *S.SE* with the ability to verify evidence of the validity of the indicated information.

6.2.3. User attribute definition

FIA_ATD.1 User attribute definition

FIA_ATD.1.1	The TSF shall maintain the following list of security attributes belonging to individual users: *D.User_Passcode, D.Card_Data, iPhone Unlock, Wrist Detection, Erase Data.*
Application Note: The update of D.OS shall not modify these user attributes.	

6.2.4. Specification of secrets

FIA_SOS.2 TSF Generation of secrets

FIA_SOS.2.1	The TSF shall provide a mechanism to generate secrets that meet *FIPS 140-3 validated cryptographic module.*
FIA_SOS.2.2	The TSF shall be able to enforce the use of TSF generated secrets for *creation of the D.Unlock_Secret during the Unlock with iPhone Setup.*

6.3. Access/Flow Control SFRs

6.3.1. Authentication_SFP

FDP_ACC.2/Authentication_SFP Complete access control

FDP_ACC.2.1/ Authentication_SFP	The TSF shall enforce the *Authentication_SFP* on: Subjects: *S.User, S.iPhone* Objects: *the TSF* and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/ Authentication_SFP	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

FDP_ACF.1/Authentication_SFP Security attribute-based access control

FDP_ACF.1.1/ Authentication_SFP	The TSF shall enforce the *Authentication_SFP* to objects based on the following: Subjects: *S.User, S.iPhone* Objects: *the TSF* Security attributes: *"iPhone Unlock"*
FDP_ACF.1.2/ Authentication_SFP	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: *S.User is authenticated according to FIA_UAU.5.*
FDP_ACF.1.3/ Authentication_SFP	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: *If "iPhone Unlock" set to "enabled", S.iPhone is allowed to unlock the TOE.*
FDP_ACF.1.4/ Authentication_SFP	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: *"Passcode_off" is enabled.*

6.3.2. Payment_SFP

FDP_ETC.2/Transaction Export of user data with security attributes

FDP_ETC.2.1/ Transaction	The TSF shall enforce the *Payment_SFP* when exporting user data, controlled under the SFP(s), outside of the TOE.
FDP_ETC.2.2/ Transaction	The TSF shall export the user data with the user data's associated security attributes.
FDP_ETC.2.3/ Transaction	The TSF shall ensure that the security attributes, when exported outside the TOE, are unambiguously associated with the exported user data.

FDP_ETC.2.4/ Transaction	The TSF shall enforce the following rules when user data is exported from the TOE:
	- *exported Payment_SFP details (the card to be used from Apple Wallet, the amount and the payee in the case of e-commerce payments) are those displayed to S.User for payment authorization (during re-authentication request (FIA_UAU.6 Re-authenticating) or user intent request).*
	- *D.Payment_Data includes a unique identifier, which can be either:*
	  - *A Terminal Unpredictable Number, for near-field-communication (NFC) transactions, or*
	  - *An Apple Pay server nonce, for transactions within apps, or Apple Cash transfers.*

FDP_ACC.2/Payment_SFP Complete access control

FDP_ACC.2.1/ Payment_SFP	The TSF shall enforce the *Payment_SFP* on: Subjects: *S.User* Objects: *D.Payment_Data (including S.User and S.Merchant Data)* , and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/ Payment_SFP	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Note: the only possible operation on D.Payment_Data is to export it as a transaction order to the Secure Element.

FDP_ACF.1/Payment_SFP Security attribute based access control

FDP_ACF.1.1/ Payment_SFP	The TSF shall enforce the *Payment_SFP* to objects based on the following: Subjects: *S.User* Objects: *D.Payment_Data (including S.User and S.Merchant Data), the Secure Element (through a trusted channel)* Security attributes: *"User Authorization", "Passcode_off"*
FDP_ACF.1.2/ Payment_SFP	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

	- *Export of D.Payment_Data with "User Authorization" set to "yes" to the Secure Element is allowed if:*
	  - *S.User shows their intent to pay (double clicking the side button)*
	  - *S.User had been successfully re-authenticated for transaction validation (FIA_UAU.6) if "Wrist Detection" set to "disable".*

FDP_ACF.1.3/ Payment_SFP	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: *none*.
FDP_ACF.1.4/ Payment_SFP	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: *"Passcode_off" is "enabled"*.

6.3.3.Card_Perso_SFP

FDP_ACC.2/Card_Perso_SFP Complete access control

FDP_ACC.2.1/ Card_Perso_SFP	The TSF shall enforce the *Card_Perso_SFP* on Subjects: *S.User, S.Apple_Servers* Objects: *D.Card_Data* and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/ Card_Perso_SFP	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

FDP_ACF.1/Card_Perso_SFP Security attribute based access control

FDP_ACF.1.1/ Card_Perso_SFP	The TSF shall enforce the *Card_Perso_SFP* to objects based on the following: Subjects: *S.User, S.Apple_Servers* Objects: *D.Card_Data* Security attributes: *Passcode_off*
FDP_ACF.1.2/ Card_Perso_SFP	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: - *S.User is connected to its iCloud account, and is authenticated on the TOE*.
FDP_ACF.1.3/ Card_Perso_SFP	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: *none*.
FDP_ACF.1.4/ Card_Perso_SFP	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: "*Passcode_off" is enabled*.

FDP_ETC.2/Card_Perso_SFP Export of user data with security attributes

FDP_ETC.2.1/ Card_Perso_SFP	The TSF shall enforce the *Card_Perso_SFP* when exporting user data, controlled under the SFP(s), outside of the TOE.
FDP_ETC.2.2/ Card_Perso_SFP	The TSF shall export the user data with the user data's associated security attributes.
FDP_ETC.2.3/ Card_Perso_SFP	The TSF shall ensure that the security attributes, when exported outside the TOE, are unambiguously associated with the exported user data.
FDP_ETC.2.4/ Card_Perso_SFP	The TSF shall enforce the following rules when user data is exported from the TOE: - *D.Card_Data is encrypted before being exported to S.Apple_Servers* - *"Card Data Confidential parts" are not kept on the TOE after being exported.*

FPT_ITC.1 Inter-TSF confidentiality during transmission

FPT_ITC.1.1	The TSF shall protect all TSF data transmitted from the TSF to another trusted IT product from unauthorised disclosure during transmission.

FDP_ITC.1 Import of user data without security attributes

FDP_ITC.1.1	The TSF shall enforce the *Card_Perso_SFP* when importing user data, controlled under the SFP, from outside of the TOE.
FDP_ITC.1.2	The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.
FDP_ITC.1.3	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: *None*.

Note: security attributes of the Card Data are "Card Data Confidential parts" in section 6.1.3. These requirements specify that the confidential part of the cards data is used for card enrollment by the *S.iPhone* but are not stored on the TOE.

6.4. Secure Enclave/Secure Element Trusted Channel

FTP_ITC.1/SE Inter-TSF trusted channel

FTP_ITC.1.1/SE	The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
FTP_ITC.1.2/SE	The TSF shall permit *the TSF* to initiate communication via the trusted channel.
FTP_ITC.1.3/SE	The TSF shall initiate communication via the trusted channel for *Payment initiation and transmission of D.Payment_Data*.

FDP_UCT.1/SE Basic data exchange confidentiality

FDP_UCT.1.1/SE	The TSF shall enforce the *Payment_SFP,* to *transmit* user data in a manner protected from unauthorised disclosure.

FDP_UIT.1/SE Data exchange integrity

FDP_UIT.1.1/SE	The TSF shall enforce the *Payment_SFP,* to *transmit and receive* user data in a manner protected from *modification, insertion and replay* errors.
FDP_UIT.1.2/SE	The TSF shall be able to determine on receipt of user data, whether *modification, insertion or replay* has occurred.

FPT_RPL.1/SE Replay detection

FPT_RPL.1.1/SE	The TSF shall detect replay for the following entities: *S.SE*.
FPT_RPL.1.2/SE	The TSF shall perform *reject data* when replay is detected.

6.5. Secure Enclave/iPhone Trusted Channel

FTP_ITC.1/iPhone Inter-TSF trusted channel

FTP_ITC.1.1/ iPhone	The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
FTP_ITC.1.2/ iPhone	The TSF shall permit *the TSF* to initiate communication via the trusted channel.
FTP_ITC.1.3/ iPhone	The TSF shall initiate communication via the trusted channel for: *unlock the TOE. card provisioning/enrolment, Apple Pay/Apple Card setup*

FDP_UCT.1/iPhone Basic data exchange confidentiality

FDP_UCT.1.1/ iPhone	The TSF shall enforce the *Authentication_SFP,* to *transmit and receive* user data in a manner protected from unauthorised disclosure.

FDP_UIT.1/iPhone Data exchange integrity

FDP_UIT.1.1/ iPhone	The TSF shall enforce the *Authentication_SFP* to *transmit and receive* user data in a manner protected from *modification, insertion and replay* errors.
FDP_UIT.1.2/ iPhone	The TSF shall be able to determine on receipt of user data, whether *modification, insertion or replay* has occurred.

FPT_RPL.1/iPhone Replay detection

FPT_RPL.1.1/iPhone	The TSF shall detect replay for the following entities: *S.iPhone*.
FPT_RPL.1.2/iPhone	The TSF shall perform *reject data* when replay is detected.

6.6. Local data protection

FPR_UNO.1 Unobservability

FPR_UNO.1.1	The TSF shall ensure that *S.Attacker* are unable to observe the operation *Passcode set, Passcode check, Passcode Update, Passcode removal, iPhone Unlock activation,* on *D. D.User_Passcode, D.Card_Data, D.SEP_iPhone, D.Unlock_Secret* by *S.User*.

FDP_RIP.1 Subset residual information protection

FDP_RIP.1.1	The TSF shall ensure that any previous information content of a resource is made unavailable upon the *deallocation of the resource from* the following objects: *D.User_Passcode, iCloud Account (in D.User_Configuration)*, iPhone pairing Status (*in D.User_Configuration)*.

Application Note:

- The removal of the iCloud Account by the S.User triggers the deallocation of all the Apple Pay data/configuration
- The revocation of individual card by its issuer triggers the deallocation of the related card data
- The procedure of Reset on the device triggers the deallocation of all security attributes except the D.OS
- The removal of “Card Data Confidential parts” is done by instructing the Secure Element to mark the card as deleted
- Unpairing Apple Watch with the S.iPhone results in erase of the device and removal of cards
- The removal of the passcode by the S.User disables the actual passcode and Unlock Secret; and triggers the deallocation of the iCloud Account information and all the Apple Pay data

FDP_SDI.1 Stored data integrity monitoring

FDP_SDI.1.1	The TSF shall monitor user data stored in containers controlled by the TSF for *integrity errors* on all objects, based on the following attributes: *D.Card_Data, D.OS*.

6.7. TSF management

6.7.1. Roles and Management Functions

FMT_SMR.1 Security roles

FMT_SMR.1.1	The TSF shall maintain the roles *S.User, S.iPhone*.
FMT_SMR.1.2	The TSF shall be able to associate users with roles.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1	The TSF shall be capable of performing the following management functions: *management of security attributes (see FMT_MSA.1)*.

6.7.2. Management of security attributes

FMT_MSA.3 Static attribute initialization

FMT_MSA.3.1	The TSF shall enforce the *Payment_SFP, Card_Perso_SFP* to provide ***"Passcode_off"*** restrictive (disabled) ***"iPhone Unlock"*** restrictive (disabled) ***"Wrist detection"*** permissive (enabled) ***"Erase data"*** restrictive (disabled) default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2	The TSF shall allow *nobody* to specify alternative initial values to override the default values when an object or information is created.

FMT_MSA.1 Management of security attributes

FMT_MSA.1.1	The TSF shall enforce the *Authentication_SFP, Card_Perso_SFP* to restrict the ability to *modify* the security attributes *"Passcode_off", "Wrist detection", "Erase data", "iPhone Unlock"* to *"S.User", "S.iPhone"*.

Application Note:

- When "Passcode_off" is enabled, the TSF removes the D.Card_Data and the Unlock Secret according to FDP_RIP.1.
- Except for "Erase data", all the lowering of the security attributes (changing from restrictive to permissive value) done from the Paired iPhone (i.e. *S.iPhone*) must be confirmed by the Authenticated User (i.e. *S.User*) on the TOE.

6.7.3. Management of TSF Data

FMT_MTD.1 Management of TSF data

FMT_MTD.1.1	The TSF shall restrict the ability to *update* the *D.OS* to *S.User, S.iPhone*.

FMT_MTD.3 Secure TSF data

FMT_MTD.3.1	The TSF shall ensure that only secure values are accepted for *D.OS*.

Application Note: secure value is defined by "OS_signature" is valid and signed by "Apple OS public key".

6.8. Security Requirements Rationale

6.8.1. Security Functional Requirements (SFR) Dependencies

TOE SFR	Required dependencies	Covered by
FIA_UAU.2	FIA_UID.1	FIA_UID.2
FIA_AFL.1/Erase	FIA_UAU.1	FIA_UAU.2
FIA_AFL.1/Delay	FIA_UAU.1	FIA_UAU.2
FDP_ACC.2/Authentication_SFP	FDP_ACF.1	FDP_ACF.1/Authentication_SFP
FDP_ACF.1/Authentication_SFP	FDP_ACC.1 FMT_MSA.3	FDP_ACC.2/Authentication_SFP FMT_MSA.3
FDP_ACC.2/Payment_SFP	FDP_ACF.1	FDP_ACF.1/Payment_SFP
FDP_ACF.1/Payment_SFP	FDP_ACC.1 FMT_MSA.3	FDP_ACC.2/Payment_SFP FMT_MSA.3
FDP_ETC.2/Transaction	FDP_ACC.1, or FDP_IFC.1	FDP_ACC.2/Payment_SFP
FDP_ACC.2/Card_Perso_SFP	FDP_ACF.1	FDP_ACF.1/Card_Perso_SFP
FDP_ACF.1/Card_Perso_SFP	FDP_ACC.1 FMT_MSA.3	FDP_ACC.2/Card_Perso_SFP FMT_MSA.3
FDP_ITC.1	FDP_ACC.1, or FDP_IFC.1 FMT_MSA.3	FDP_ACC.2/Card_Perso_SFP FMT_MSA.3
FDP_ETC.2/Card_Perso_SFP	FDP_ACC.1, or FDP_IFC.1	FDP_ACC.2/Card_Perso_SFP
FDP_UCT.1/SE	FDP_ACC.1, or FDP_IFC.1 FTP_ITC.1, or FTP_TRP.1	FDP_ACF.1/Payment_SFP, and FDP_ACF.1/Card_Perso_SFP FTP_ITC.1/SE
FDP_UIT.1/SE	FDP_ACC.1, or FDP_IFC.1 FTP_ITC.1, or FTP_TRP.1	FDP_ACF.1/Payment_SFP, and FDP_ACF.1/Card_Perso_SFP FTP_ITC.1/SE
FDP_UCT.1/iPhone	FDP_ACC.1, or FDP_IFC.1 FTP_ITC.1, or FTP_TRP.1	FDP_ACF.1/Authentication_SFP FTP_ITC.1/iPhone
FDP_UIT.1/iPhone	FDP_ACC.1, or FDP_IFC.1 FTP_ITC.1, or FTP_TRP.1	FDP_ACF.1/Authentication_SFP FTP_ITC.1/iPhone
FMT_SMR.1	FIA_UID.1	FIA_UID.2
FMT_MSA.3	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1, FMT_SMR.1
FMT_MSA.1	FDP_ACC.1, or FDP_IFC.1 FMT_SMR.1, SMT_SMF.1	FDP_ACC.2/Authentication_SFP, FDP_ACC.2/Authentication_SFP, and FDP_ACC.2/Card_Perso_SFP FMT_SMR.1, FMT_SMF.1
FMT_MTD.1	FMT_SMR.1, SMT_SMF.1	FMT_SMR.1, SMT_SMF.1
FMT_MTD.3	FMT_MTD.1	FMT_MTD.1

Requirements without dependency: FIA_UID.2, FIA_SOS.2, FIA_UAU.6, FIA_UAU.5, FDP_DAU.1, FIA_ATD.1, FPT_ITC.1, FTP_ITC.1/SE, FPT_RPL.1/SE, FTP_ITC.1/iPhone, FPT_RPL.1/iPhone, FPR_UNO.1, FDP_RIP.1, FMT_SMF.1 and FDP_SDI.1

6.8.2. Rationale SFR/Security Objectives for the TOE

Objective reference	Rationale
OT.User_Auth	FIA_UID.2 and FIA_UAU.2 enforce the User of the device is authenticated before being able to do any action in the user interface if the wrist detection is enabled. FIA_UAU.5 specifies different authentication methods. FIA_UAU.6 enforces a re-authentication for the OS update, for changing the authentication configuration or for a transaction validation. FMT_SMR.1 and FMT_SMF.1 SF ensure that the TSF shall maintain the roles Authenticated User (*S.User*) and be capable of performing the management functions. FIA_ATD.1, FMT_MSA.3, FMT_MSA.1 and FMT_MTD.1 specify the management of related information. FMT_MTD.3 specifies the signature verification on the OS update. FIA_AFL.1/Erase and FIA_AFL.1/Delay specify the failure handling in case of wrong passcode authentication. FPR_UNO.1 ensures the non-observability of the passcode.
OT.Card_Data	The SFP Card_Perso_SFP and the associated SFRs (FDP_ACC.2/Card_Perso_SFP, FDP_ACF.1/Card_Perso_SFP, FDP_ITC.1, FDP_ETC.2/Card_Perso_SFP, FPT_ITC.1) enforce card data stored in Apple Wallet does not include sensitive card data as it is only sent to the Secure Element. Also, FTP_ITC.1/SE, FDP_UCT.1/SE and FDP_UIT.1/SE enforce that all the data exchanged between the TSF and S.SE is protected (with their corresponding security property) by a trusted channel. FPR_UNO.1 ensures the non-observability of secret card data. FDP_SDI.1 ensures card data stored in containers controlled by the TSF are monitored for integrity errors.
OT.Passcode_Delete	FDP_ACC.2/Card_Perso_SFP and FDP_ACF.1.4/Card_Perso_SFP enforce the card enrollment is not possible if Passcode_off is enabled. FIA_UAU.6 enforces a re-authentication for changing the Passcode_off configuration. FPR_UNO.1 ensures the non-observability of the passcode.
OT.Card_Delete	FDP_RIP.1 ensures the confidential parts of the card data are securely removed by the Secure Element. FMT_MSA.1.1 ensures the secure delete in case of passcode turning off. FPR_UNO.1 ensures the non-observability of sensitive card data.
OT.Auth_SE	FIA_UID.2, FIA_UAU.2, FIA_UAU.5 specify the base of the authentication feature. FIA_AFL.1/Erase and FIA_AFL.1/Delay protect the TSF and the user data against brute force attacks. FIA_UAU.6 specifies the re-authentication for some functions of the TOE.
OT.Payment	FDP_ETC.2.4/Transaction ensures the transaction details are displayed before a transaction is validated by the User. FIA_UAU.6 ensures each transaction is validated by a re-authentication if the Wrist detection feature is disabled. FDP_DAU.1 provides evidence of the validity of Apple Pay Transaction Data, verifiable by the card issuer.

	FDP_ACC.2/Payment_SFP and FDP_ACF.1/Payment_SFP ensure user authorization is done before each transaction. FPT_RPL.1/SE ensures transaction replay detection.
OT.Device_Reset	FDP_RIP.1 ensures all sensitive data is securely removed during a device reset.
OT.Anti_Replay	FDP_ETC.2.4/Transaction ensures a unique identifier provided by the NFC terminal or Apple server is included in the transaction data, avoiding any replay attack.
OT.OS_Update	FIA_UAU.6.1 ensures *S.User* is re-authenticated before the OS update proceeds. FIA_ATD.1.1 ensures that all the user data with impact on the TSF behavior are not modified during the OS update process. FDP_SDI.1 ensures D.OS stored in containers controlled by the TSF is monitored for integrity errors.
OT.iPhone	FTP_ITC.1/iPhone, FDP_UCT.1/iPhone, FDP_UIT.1/iPhone, FIA_SOS.2, FPR_UNO.1 and FPT_RPL.1/iPhone enforce that all the data exchanged between the TSF and S.iPhone is protected by a trusted channel regarding the confidentiality and the integrity. The protected operations are specified by FDP_ACC.2/Authentication_SFP and FDP_ACF.1/Authentication_SFP.

6.8.3. SAR Dependencies

TOE SAR	Dependencies
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1
ADV_FSP.3	ADV_TDS.1
ADV_TDS.1	ADV_FSP.2
AGD_OPE.1	ADV_FSP.1
AGD_PRE.1	No dependencies
ALC_CMC.2	ALC_CMS.1
ALC_CMS.2	No dependencies
ALC_DEL.1	No dependencies
ALC_FLR.3	No dependencies
ASE_CCL.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.1
ASE_ECD.1	No dependencies
ASE_INT.1	No dependencies
ASE_OBJ.2	ASE_SPD.1
ASE_REQ.2	ASE_OBJ.2, ASE_ECD.1
ASE_SPD.1	No dependencies
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1
ATE_COV.1	ADV_FSP.2, ATE_FUN.1
ATE_FUN.1	ATE_COV.1
ATE_IND.2	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1

All the dependencies are covered.

6.8.4. SAR Rationale

For this evaluation, the SARs of EAL2 have been chosen as they provide assurance by a full security target and an analysis of the SFRs in that ST, using a functional and interface specification, guidance documentation and a basic description of the architecture of the TOE, to understand the security behavior. It was established that this level is appropriate for the model of attack of Apple Pay.

ADV_FSP.3 augmentation has been chosen to enhance the level of information provided related to SFR-enforcing TSFIs and provide a complete summary of the TOE.

ALC_FLR.3 augmentation has been chosen to guarantee the security of the security functions in the scope of this security target during the maintenance of the TOE.

7. TOE Summary Specification

This section describes the security functions of the TOE covering the SFR of the previous chapter.

7.1. SF User authentication and management

When using the TOE, before being able to use Apple Pay or Apple Cash, the TOE requires that a passcode is setup. The passcode is required under the following circumstances:

- Update passcode
- Update the OS to a new version signed by Apple
- Setup Unlock with iPhone
- Unlock of the device
- Authorize Apple Pay transactions/Apple Cash transfer (if the Wrist detection feature is disabled)

Apple Watch must be paired with an iPhone. The *S.iPhone* can be used to manage the Watch content (OS version, Wallet cards and configuration, Apps installed, and critical settings like Wrist Detection). It can also be used to unlock the Apple Watch when the feature Unlock with iPhone is enabled.

If Wrist detection is enabled and if the Apple Watch is unlocked, only user intent (double-click of the side button) is required to authorize Apple Pay transactions/Apple Cash transfers. Wrist detection enables the watch to stay securely unlocked on the user's wrist, dispensing with the need to enter a passcode to authorize a transaction. When Wrist detection is disabled, when the User uses Apple Pay on their Apple Watch, they will be prompted to enter their passcode when they double-click the side button to authorize the payment.

These features implement the requirements listed in §6.2 and in §6.3.1 for the authentication, in §6.6 for local data protection and in FMT_SMF.1, FMT_MSA.1, and FMT_MSA.3 for management.

The following table summarizes the functions supporting User authentication.

Function	Description
Passcode authentication	Passcode_Setup: Setup of the device's passcode.
	Passcode_Update: Update of the device's passcode.
	Passcode_Verify: Authentication of the User using their passcode.
	Passcode_Delete: Removal of the passcode
Unlock with iPhone	Unlock_With_iPhone_Setup: Enable the feature « Unlock with iPhone ».
	Unlock_With_iPhone_Check: Secure process to import the Unlock Secret to the TOE's Secure Enclave for unlocking the device.

7.1.1. Passcode_Setup

The TOE offers a configuration setting where the User can set up a passcode that will be used to perform authentication in order to access restricted services on the device: unlock, Apple Pay transactions, Apple Cash Transfers, and more. The TOE enforces strong access control in order to prevent access to these restricted services without authentication (FIA_UID.2).

The TOE ensures the non-observability of the passcode during the setting process within the Secure Enclave (FPR_UNO.1).

7.1.2. Passcode_Verify

The TOE offers passcode entry to the User as part of the device unlock procedure, or during the User authorization step of an Apple Pay transaction or Apple Cash transfer. The TSFs ensure that passcode verification preserves the secrecy of the passcode value set by the User, which is expected in order to prevent an attacker from guessing the code by observing the verification process (FPR_UNO.1). The TOE ensures that the verification process would not validate a code that does not match the passcode set by the User and detect alterations to the configured value (FDP_SDI.1), preventing use of the User account (and related data) and forcing a device panic. To discourage brute-force passcode attacks, the TOE authentication failure policy is escalating time delays after the entry of an invalid passcode (FIA_AFL.1/Delay) or to erase data after 10 attempts if the feature «Erase data » is enabled (FIA_AFL.1/Erase).

7.1.3. Passcode_Update

The User can update the passcode value through the device's settings menu. This functions as a verify operation, as knowledge of the previous passcode value is required to proceed to the setup step where the User types a new value and effectively updates the value in the Secure Enclave (FIA_UAU.2, FDP_ACC.2/Authentication_SFP and FDP_ACF.1/Authentication_SFP). The TOE ensures the non-observability of the old passcode during the verification as well as of the new passcode during the setting process (FPR_UNO.1). The TOE also enforces passcode alteration detection, preventing a corrupted passcode from being used to reset the authentication function (FDP_SDI.1).

7.1.4. Passcode_Delete

If the User wants to fully remove the use of the passcode on the device, and because this would not allow an adequate security level for the processing of the TSF, the actual passcode and Unlock Secret are disabled; and the iCloud Account information and all the Apple Pay data is deallocated. (FDP_RIP.1).

To prevent misuse of the passcode deletion function, the User is required to first verify the current passcode before proceeding (FIA_UAU.2, FDP_ACC.2/Authentication_SFP and FDP_ACF.1/Authentication_SFP), and the TOE protects the passcode secrecy during the verification process in case an attacker was to use this path for attempting an observation-based attack (FPR_UNO.1).

The Secure Element, in the TOE Environment, is responsible for securely deleting the Apple Pay card data and Apple Cash card data during this process

7.1.5. Unlock_With_iPhone_Setup

Optionally, the User can decide to enable a feature called "Unlock With iPhone": Apple Watch is unlocked whenever the User unlocks the paired iPhone (*S.iPhone*). Enabling this feature requires the User to enter the passcode on the Watch (FDP_ACC.2/Authentication_SFP, FDP_ACF.1/Authentication_SFP, FMT_MSA.1, and FMT_MSA.3) before the setup process is started.

During this process, the TOE Secure Enclave Processor creates a non-predictable Unlock Secret (FIA_SOS.2 and FPR_UNO.1) and associates it to the User Authentication mechanism (FMT_SMF.1).

The TOE's Secure Enclave securely exports the Unlock Secret to the paired iPhone's Secure Enclave applying confidentiality measures to the sensitive asset with the Secure Enclave iPhone Pairing Secret (FTP_ITC.1/iPhone, FDP_UCT.1/iPhone, FDP_UIT.1/iPhone, FPT_RPL.1).

Until the feature is disabled, the TOE maintains the security attributes and assets of the Unlock with iPhone process, including the Unlock Secret (FIA_ATD.1).

7.1.6. Unlock_With_iPhone_Check

Optionally, if the User enabled the feature called “Unlock with iPhone”, the TOE is unlocked whenever the User unlocks the paired iPhone (*S.iPhone*).

During this process, the paired iPhone’s Secure Enclave Processor securely imports the Unlock Secret to the TOE’s Secure Enclave applying confidentiality measures to the sensitive asset with the Secure Enclave iPhone Pairing Secret (FTP_ITC.1/iPhone, FDP_UCT.1/iPhone, FDP_UIT.1/iPhone, FPT_RPL.1).

7.2. SF Secure Enclave/Secure Element secure channel

The TSF is able to initialize a secure channel with the Secure Element (FTP_ITC.1/SE). This secure channel protects the exchanged data with its corresponding security properties: against disclosure (FDP_UCT.1/SE), modification (FDP_UIT.1/SE) and replay (FPT_RPL.1/SE).

7.3. SF Secure Enclave/iPhone secure channel

The TSF is able to initialize a secure channel to S.iPhone (FTP_ITC.1/iPhone). This secure channel protects the exchanged data against disclosure (FDP_UCT.1/iPhone), modification (FDP_UIT.1/iPhone) and replay (FPT_RPL.1/iPhone).

7.4. SF Card Data management

The following table summarizes the functions supporting Card Data management.

Function	Description
Apple Pay card data management	AP_Card_Provisioning : Provisioning of a new card for Apple Pay.
	AP_Cancellation: User removes a card from the Apple Pay wallet.
	AP_Revocation: Card issuer initiated suspend/unlink of a payment card in Apple Wallet.
Apple Cash card data management	APC_Enroll : Enroll in Apple Cash services.
	APC_Disenroll: User cancels their enrollment in Apple Cash services.
	APC_Revocation: Card issuer initiated suspend/unlink of user’s Apple Cash services.

7.4.1. AP_Card_Provisioning

On the TOE, there are different ways to add a card into Apple Wallet:

- Adding a card manually
- Adding cards on file from an iTunes Store account to Apple Pay
- Adding cards from a card issuer’s app
- Adding cards from a card issuer’s website (only for Apple Pay, not available for Apple Cash)

- Adding cards that were provisioned on a different device (multi-device provisioning, available with watchOS 10 or later)

The first three modes are only available to an authenticated User (*S.User*) on the paired iPhone (*S.iPhone*), and the Apple Watch with passcode enabled (FIA_UAU.2, FDP_ACC.2/Card_Perso_SFP, and FDP_ACF.1/Card_Perso_SFP).

Multi-device provisioning is available with watchOS 10 or later. When a user provisions an eligible payment card, they will also be able to push provision the card to other Apple Pay-capable devices on the same iCloud account. Nothing is copied from the original device; the other devices provision using the same flow they would use during device setup. The TOE cannot initiate push provisioning of cards, but can receive cards that were initiated by provisioning on one of the User's other devices, e.g., a Mac or iPhone.

When a User adds a card to Apple Wallet, the TSF encrypts card data (FPT_ITC.1) and sends it to Apple servers (FDP_ETC.2/Card_Perso_SFP). Full card numbers are not stored on the device (FDP_ITC.1) or on Apple servers.

Integrity protections are in place to prevent alteration of the enrolled Apple Pay card data. (FDP_SDI.1).

7.4.2.AP_Cancellation

When the User decides to remove an Apple Pay card from the Apple Wallet, the TSF order the Secure Element to securely invalidate and remove the Device Account Number (FDP_RIP.1).

7.4.3.AP_Revocation

When the card issuer decides to suspend or unlink an Apple Pay card from the Apple Wallet of a TOE User, the TSF order the Secure Element to securely invalidate and remove the Device Account Number (FDP_RIP.1).

7.4.4.APC_Enroll

To use person-to-person payments and Apple Cash, a user must be signed into their iCloud account on an Apple Cash compatible device.

The User can add or remove the Apple Cash card in the Watch app on iPhone. Full card numbers are not stored on the device (FDP_ITC.1) or on Apple servers. Integrity protections are in place in the TOE to prevent alteration of the enrolled Apple Cash card data (FDP_SDI.1).

When a user enrolls in Apple Cash, Apple securely sends the registration information to the Apple Cash card issuer, who will determine whether to approve adding an Apple Cash card to Apple Pay on the specified device. Registration information is not stored on the device nor on Apple servers. Integrity protections are in place to prevent alteration of the enrolled Apple Cash card data (FDP_SDI.1).

7.4.5.APC_Disenroll

When the User decides to remove an Apple Cash card from the Apple Wallet, the TSF order the Secure Element to securely invalidate and remove the Device Account Number (FDP_RIP.1).

7.4.6. APC_Revocation

When the card issuer decides to suspend or unlink an Apple Cash card from the Apple Wallet of a TOE User, the TSF order the Secure Element to securely invalidate and remove the Device Account Number (FDP_RIP.1).

7.5. SF Payment management

When a device initiates an Apple Pay transaction, the Secure Element, in the TOE Environment, only allows a payment to be made only after it receives authorization from the Secure Enclave. This involves confirming the User has provided intent and has authenticated (FDP_ACC.2/Payment_SFP, FDP_ACF.1/Payment_SFP, FMT_SMR.1, FMT_SMF.1, FMT_MSA.1, and FMT_MSA.3).

Apple Pay includes an anti-replay mechanism that prevents transactions to be repeated by including in D.Payment_Data:

- A Terminal Unpredictable Number, for near-field-communication (NFC) transactions, or
- An Apple Pay server nonce, for transactions within apps or on the web

The processing of the Apple Pay transaction happens in the TOE Environment, on the Secure Element, using the secret card data, the Transaction Token and producing a payment cryptogram. The TOE ensures that the payment evidence transmitted back to the card issuer for processing (through a Terminal or network) was authorized by the User (FIA_UAU.6 Re-authenticating). In the case of Apple Pay online transactions, which are processed through the Apple Pay server, this integrity protection ensures the Dynamic Linking of the transaction data by a cryptographic based Authentication Code as exposed in the PSD2 regulation. The Apple Pay server ensures the integrity of the Dynamic Linking, and the card issuer verifies that it corresponds to a valid transaction, containing the right Transaction Token and produced by a genuine Apple Pay card (FDP_DAU.1).

The exported user data (transaction data displayed to *S.User*) is controlled by FDP_ETC.2/Transaction.

The Secure Element, in the TOE Environment, is responsible for ensuring the confidentiality of the Apple Pay Card data during the transaction processing.

The following table summarizes the functions supporting Payment management.

TSF	Description
AP_Transaction	Processing of an Apple Pay transaction.
APC_Transaction	Processing of an Apple Cash peer-to-peer transfer.

7.6. SF OS Update

An OS update can be offered at any time by Apple to the User. The User is required to authenticate through a passcode verification, or the update cannot be installed (FIA_UAU.6, FIA_UAU.2, FMT_MTD.3).

The OS update preserves the user attributes, especially the card data, the passcode, and other authentication parameters (FIA_ATD.1).

The ability to update the D.OS is restricted to Authenticated User (*S.User*) (FMT_MTD.1).

7.7. SF iCloud logout & Device reset

An iCloud logout is performed when the User unlinks a device from an iCloud account. When this happens, the TOE ensures that the iCloud Account related data is securely deallocated, and that no residual information is left behind (FDP_RIP.1).

The most destructive security function available to the User is the device reset, as it erases of all the User data. When this is performed, the TOE ensures that all the sensitive data terminated as part of a Device Reset is protected from leaving residual information. This covers the iCloud Account information, all the Apple Pay Card Data, and the User authentication data like passcode (FDP_RIP.1).

Change History

Date	Version	Author	Comments
2024-12-09	1.0	Apple	Initialization of the Security Target
2025-04-03	1.1	Apple	Minor updates
2025-06-18	1.2	Apple	Minor updates

