ARIADNEXT

WE DO IDENTITY.

Cible de sécurité - MobileId Authenticator SDK pour iOS

Version 1.5, 26/07/2021: Validé

MobileID.io
DIGITAL IDENTITY
BY ARIADNEXT

Table des matières

Préface

Identification du document

Titre	Cible de sécurité - MobileId Authenticator SDK pour iOS
Référence	MOBILEID_AUTH_SDK_IOS_SEC_TARGET
Niveau de diffusion	Public
Version	1.5
Statut	Validé
Auteurs	Laurent ROUSSEL
Valideurs	Christian QUIVY, Marc NORLAIN
Date	26/07/2021

Suivi des modifications

Version	Date	Auteur	Motifs
1.0	18/03/2020	L. ROUSSEL	Création du document
1.1	03/06/2020	L. ROUSSEL	Mise à jour suite aux retours de l'ANSSI et à la réunion du 12/05/2020.
1.2	09/07/2020	L. ROUSSEL	Mise à jour suite aux retours de l'ANSSI datés du 18/06/2020 et à l'évolution de la sécurisation pour le stockage des clés.

Version	Date	Auteur	Motifs
1.3	07/10/2020	L. ROUSSEL	• Correction de la référence à une fonction de sécurité dans la couverture des menaces ; • En raison d'une gestion externe à la TOE, la fonction de sécurité FSEC_SIGN_FWK_SDK a été requalifiée en une hypothèse sur l'environnement ; • La formulation relative aux versions de TLS supportées a été revue afin de prendre en compte la v1.3 ; • Mise à jour de la version du SDK.
1.4	31/05/2021	L. ROUSSEL	Intégration des remarques et suggestions du CESTI.
1.5	26/07/2021	L. ROUSSEL	Mise à jour du document pour publication sur le site de l'ANSSI.

Diffusion

Conformément à la **Procédure de Classification de l'Information** en vigueur chez ARIADNEXT, ce document est communiqué sans aucune restriction.

Chapitre 1. Introduction

1.1. Objet du document

Ce document constitue la cible de sécurité du produit "MobileID Authenticator SDK" en vue d'une qualification au niveau élémentaire du RGS reposant sur une Certification de Sécurité de Premier Niveau (CSPN).

Note : Le terme TOE désignera, dans la suite du document, le produit faisant l'objet de l'évaluation en vue de l'obtention de la CSPN.

1.2. Identification du produit évalué

L'évaluation concerne le produit "MobileID Authenticator SDK" qui est un SDK mobile décliné pour l'OS iOS.

En complément de ce composant, des serveurs applicatifs gérant l'activation des terminaux, l'émission des requêtes d'authentification, et le contrôle des signatures en back-end, et un serveur d'authentification interviennent aussi dans les cas d'usage décrits Paragraphe 2.2 mais ne rentrent pas dans le périmètre de l'évaluation.

L'identification du produit est donnée par le tableau suivant :

Organisation éditrice	ARIADNEXT
Lien vers l'organisation	https://www.ariadnext.com/
Lien vers le produit	https://www.mobileid.io/
Catégorie de produit	Identification, authentification et contrôle d'accès
Nom commercial du produit	MobileID Authenticator SDK for iOS
Numéro de version évaluée	2.0.12

Le produit s'intègre dans une solution plus globale appelée "MobileID Authenticator".

1.3. Contexte

"MobileID Authenticator" est intégré au sein de la solution MobileID qui est un système de gestion d'identité électronique. Ce système fournit un moyen d'identification électronique au sens du règlement eIDAS (règlement EU n° 910/2014). "MobileID Authenticator" fournit un service d'authentification à ce système.

1.4. Documents de référence

[MOBILEID_AUTH_CRYPT_MECHA]	Document décrivant les mécanismes cryptographiques sur lesquels reposent certaines fonctions de sécurité du produit.
[MOBILEID_AUTH_TECHNICAL_SPEC]	Document d'architecture générale de la solution MobileID.

1.5. Définitions

Keychain	Il s'agit du système de gestion de mots de passe développé par Apple initialement pour macOS et aujourd'hui disponible sur iOS. Il peut contenir différents types de données : mots de passe, clés privés, certificats et notes sécurisées.
Moyen d'authentification	Un moyen d'authentification est un élément matériel et/ou immatériel utilisé pour s'authentifier pour un service en ligne.
Moyen d'identification électronique	Un moyen d'identification électronique est un élément matériel et/ou immatériel contenant des données d'identification personnelles et utilisé pour s'authentifier pour un service en ligne.
Secure Element	Le Secure Element est une micropuce distincte, avec son propre processeur, son stockage, sa RAM, etc., spécialement conçue à des fins de sécurité. Les SE résistent à une grande variété d'attaques, logiques et physiques, telles que les attaques par canal auxiliaire. La puce d'une carte SIM et celle d'une carte de crédit sont des exemples significatifs de SE.

1.6. Glossaire

A Asset, préfixe utilisé dans l'identifiant de certains biens

CSPN Certification de Sécurité de Premier Niveau

ECDH Elliptic Curve Diffie-Hellman

ECIES Elliptic Curve Integrated Encryption Scheme

eIDAS Electronic IDentification Authentication and trust Services

FC FranceConnect

HSM Hardware Security Module

HMAC keyed-Hash Message Authentication Code

MAC Message Authentication Code

OS Operating System

OTP One-Time Password

RGS Référentiel Général de Sécurité

SDK Software Development Kit

SA Sensitive Asset, préfixe utilisé dans l'identifiant des biens sensibles

SA_D-ENC	Sensitive Asset Doubly ENCrypted, préfixe utilisé dans la déclinaison doublement chiffrée d'un bien sensible
SA_ENC	Sensitive Asset ENCrypted, préfixe utilisé dans la déclinaison chiffrée d'un bien sensible
SE	Secure Element
SGBD	Système de Gestion de Base de Données
TOE	Target Of Evaluation

Chapitre 2. Argumentaire du produit

2.1. Description générale

"MobileID Authenticator" est une suite logicielle permettant à un usager de s'authentifier au travers d'une application mobile intégrant le SDK, installée sur son téléphone, et s'interfaçant avec le serveur.

Elle offre un cadre sécurisé pour activer, utiliser et révoquer un moyen d'authentification par la mise en oeuvre :

- d'un protocole d'échange de clés ;
- d'un mécanisme de signature ;
- de services REST.

La solution offre une authentification à deux facteurs par :

- la possession d'un téléphone hébergeant l'application mobile intégrant le "MobileID Authenticator SDK" ;
- la connaissance d'un code confidentiel.

Elle est conçue pour permettre à des usagers de s'authentifier à des services tiers en ligne, notamment ceux qui sont fournisseurs de service FC.

La solution "MobileID Authenticator" est constituée de 2 composants :

- Le "MobileID Authenticator SDK" qui est une brique logicielle fournie aux développeurs d'applications mobiles, pour la génération des clés, l'activation du moyen d'authentification et l'implémentation de fonctions d'authentification et de signature ;
- Le "MobileID Authenticator Server" qui offre l'implémentation des mécanismes cryptographiques utilisés côté serveur pour les générations de clés, la vérification des signatures et qui assure la gestion des activations et l'historisation des transactions.

2.2. Utilisation du produit

2.2.1. Utilisateurs

Utilisateur final : il s'agit de la personne physique ouvrant et utilisant l'application mobile intégrant le SDK. Cette application lui permet de s'authentifier à des services tiers par la saisie d'un code confidentiel.

Service tiers : il s'agit du système client intégrant la solution "MobileID Authenticator" en tant que service d'authentification. Ce système est à l'origine du déclenchement des demandes d'authentification émises par la solution.

Administrateur : il s'agit d'une personne physique pouvant désactiver le moyen d'authentification créé et consulter l'historique des connexions et des opérations liées à l'utilisateur final.

2.2.2. Création du moyen d'authentification

La création du moyen d'authentification s'effectue lors d'une phase d'enrôlement de l'usager.

La figure suivante illustre les étapes à réaliser avant toute utilisation de la fonctionnalité d'authentification via l'application mobile

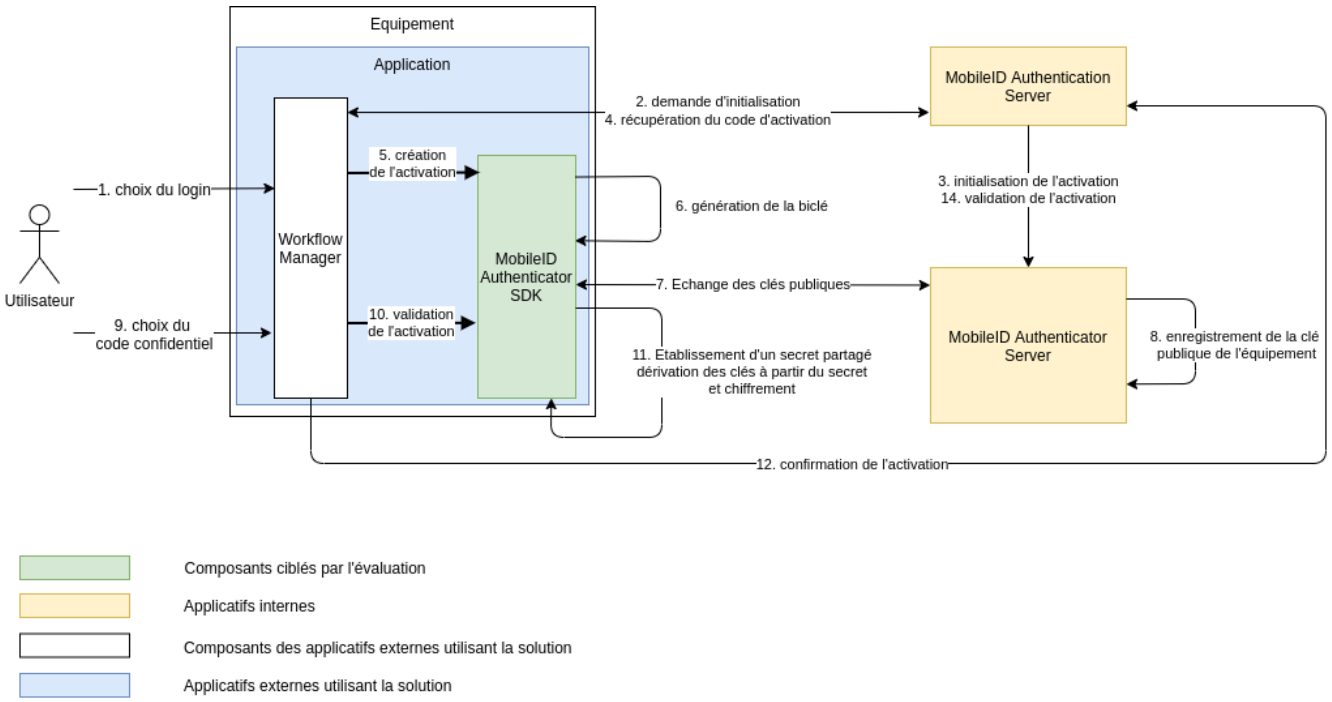

Figure 1. Etapes de création du moyen d'authentification

Figure 2. Cinématique des utilisateurs

L'application demande à l'utilisateur de définir l'identifiant de connexion qui sera utilisé lors des demandes d'authentification. Elle initie ensuite l'activation du moyen d'authentification côté serveur. Ce dernier émet un code d'activation qui sera récupéré par l'application mobile et qui aura pour effet de déclencher la génération, dans le SDK, d'une biclé liée au mobile. La clé publique créée est transmise au serveur qui génère à son tour une biclé et renvoie, en réponse, la clé publique nouvellement produite. Les données d'activation (nom du terminal, code d'activation) sont elles aussi transmises au serveur.

A partir de cet échange de clés, un secret partagé est établi entre le "MobileID Authenticator SDK" et le server (via le protocole d'échange de clés ECDH). Une série de clés symétriques sont dérivées de ce secret parmi lesquelles une clé associée à la possession du téléphone et une clé associée à la connaissance d'un code confidentiel.

Côté serveur, les mêmes jeux de clés symétriques sont dérivés. Cependant la dérivation est réalisée à la volée pour chaque contrôle de signature. Les clés dérivées ne sont donc jamais sauvegardées. Seule, la clé publique de l'équipement et la clé privée associée à l'activation côté serveur sont stockées en base de données. Cette clé privée est chiffrée avant son enregistrement en base.

A l'issue de cette phase de génération, l'application demande à l'utilisateur de définir le code confidentiel à 6 chiffres avec lequel sera dérivée une clé utilisée pour chiffrer la clé de connaissance. La

clé de possession est elle aussi chiffrée par le "MobileID Authenticator SDK" avec une clé symétrique générée à partir de données matérielles du téléphone et logicielles de l'application. Les résultats sont une nouvelle fois, chiffrés par la clé de chiffrement générée et gérée par iOS, lors de leur insertion, pour stockage sur le terminal, dans le keychain iOS. La clé de chiffrement générée par iOS est stockée au sein du SE du terminal.

A l'issue de ce processus, l'application notifie l'activation du moyen d'authentification au serveur tout en lui fournissant le login auquel il est associé.

2.2.3. Authentification à un service tiers depuis un mobile

Son moyen d'authentification créé, l'utilisateur peut l'exploiter pour accéder aux services tiers raccordés au serveur d'authentification.

La figure suivante illustre les étapes relatives à l'authentification à un service tiers

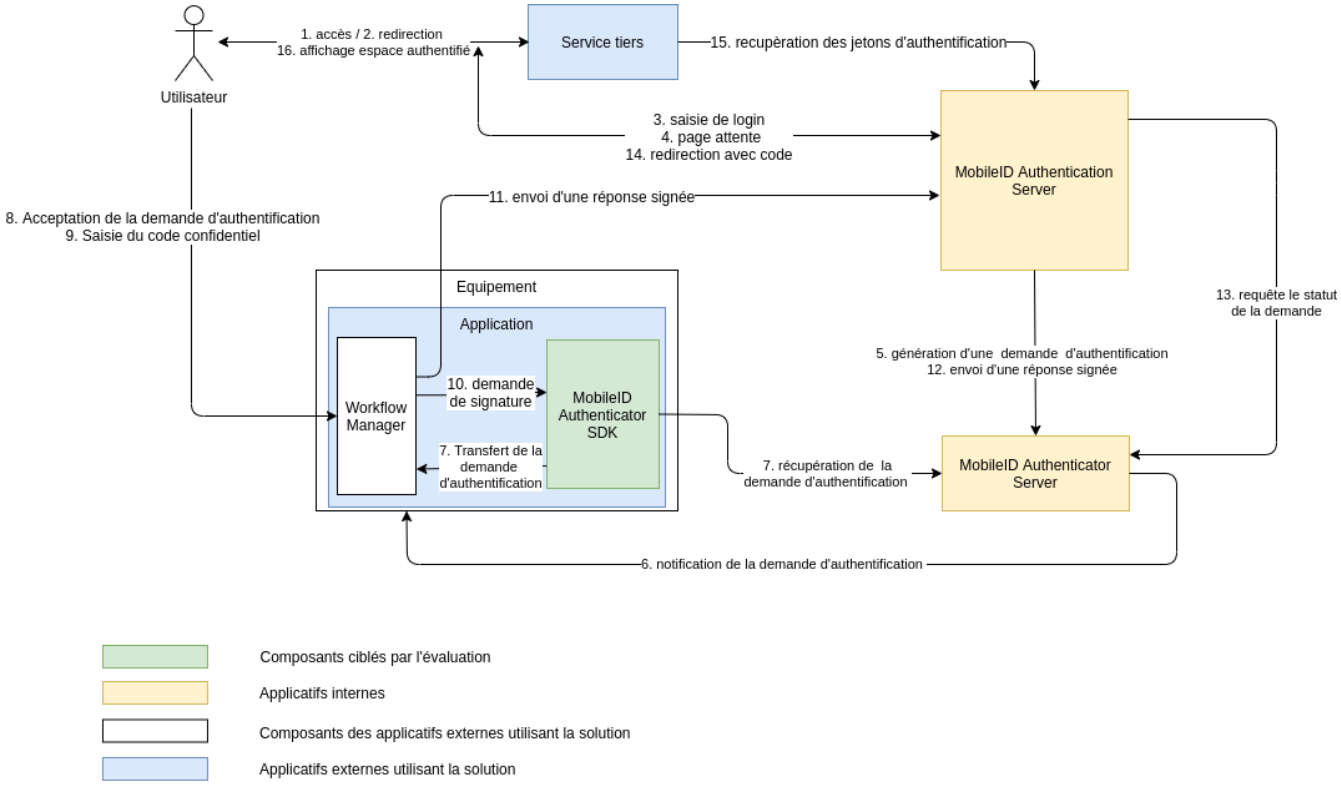

Figure 3. Etapes d'authentification à un service tiers

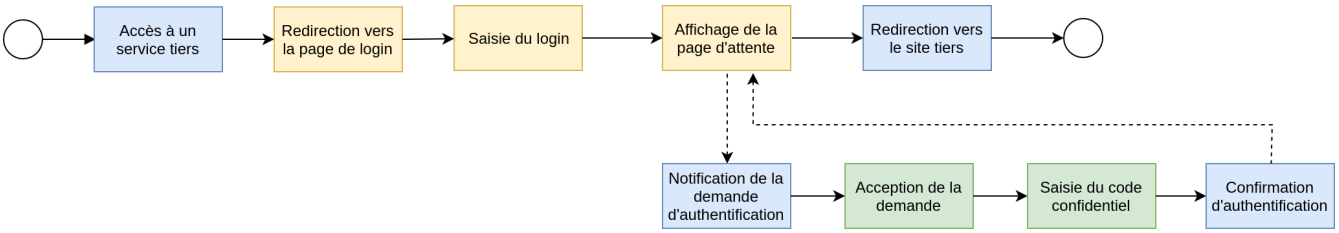

Figure 4. Cinématique des utilisateurs

L'utilisateur demande l'accès à un service tiers nécessitant son authentification. Ce dernier le redirige vers la page de login du serveur d'authentification dans laquelle il saisit son identifiant de connexion (un numéro de téléphone portable ou une adresse mail par exemple).

Le serveur d'authentification affiche une page d'attente à l'utilisateur et demande la génération d'une

demande d'authentification au "MobileID Authenticator Server". Ce dernier émet une notification indiquant qu'une demande d'authentification est en cours sur l'équipement concerné. L'application mobile récupère l'ensemble des informations liées à cette demande auprès du serveur.

L'utilisateur reçoit la demande et saisit son code confidentiel. A partir du code confidentiel, le "MobileID Authenticator SDK" redérive la clé symétrique pour déchiffrer la clé de connaissance. (La version simplement chiffrée de la clé de possession, est récupérée par le "MobileID Authenticator SDK" auprès du conteneur sécurisé iOS, le keychain). Le "MobileID Authenticator SDK" regénère aussi la clé symétrique pour déchiffrer la clé de possession. (La version simplement chiffrée de la clé de possession est récupérée par le "MobileID Authenticator SDK" auprès du conteneur sécurisé iOS, le keychain). Clé de connaissance et clé de possession sont utilisées pour signer la réponse à la demande et la transférer au "MobileID Authenticator Server".

Ce dernier contrôle la signature en produisant le même calcul de signature par régénération des clés de possession et de connaissance à partir de la clé maîtresse, elle-même reconstruite à partir de la clé publique de l'équipement et de la clé privée associée à l'activation usager.

En cas de résultat identique, le "MobileID Authenticator Server" valide la demande ce qui a pour effet de débloquer la page d'attente par le "MobileID Authentication Server" pour rediriger l'utilisateur authentifié vers le service tiers.

En cas de résultat différent, la demande d'authentification est refusée et une page d'échec est affichée à l'usager par le "MobileID Authentication Server".

2.2.4. Elements complémentaires

Identifiant de connexion de l'usager

Le type de données utilisé en tant qu'identifiant de connexion est choisi par le client intégrant le "MobileID Authenticator SDK" au sein d'une application mobile utilisée comme moyen d'authentification.

Fonction de récupération

Il n'existe pas de fonction de récupération ou de changement du code confidentiel en cas d'oubli par l'usager. Lorsque ces cas surviennent le moyen d'authentification est systématiquement invalidé et l'utilisateur doit nécessairement recommencer le processus de création de son moyen d'authentification.

2.3. Environnement d'utilisation

La solution "MobileID Authenticator" est destinée à être intégrée en tant que moyen d'authentification au sein d'un fournisseur d'identité FranceConnect qui est le serveur d'authentification utilisé par la plupart des services publics français. Depuis le décret du 08/11/2018, "les personnes morales de droit privé proposant des services en ligne dont l'usage nécessite, conformément à des dispositions législatives ou réglementaires la vérification de l'identité de leurs utilisateurs ou de celle de certains de leurs attributs" peuvent s'interfacer avec FranceConnect. Concrètement, cela signifie que son utilisation concernera les services bancaires, les assurances, les sites de jeux en lignes etc.

La solution pourra également être utilisée dans tout autre contexte nécessitant une authentification multifactorielle.

L'environnement technique est décrit Chapitre 3.

2.4. Dépendances

Le composant "MobileID Authenticator SDK" possède des dépendances logicielles sur :

- OpenSSL ;
- PowerAuth Mobile SDK ;
- TrustKit.

Il ne possède pas de dépendance matérielle.

2.5. Périmètre de l'évaluation

Le périmètre de l'évaluation est limité au composant en vert sur la figure suivante :

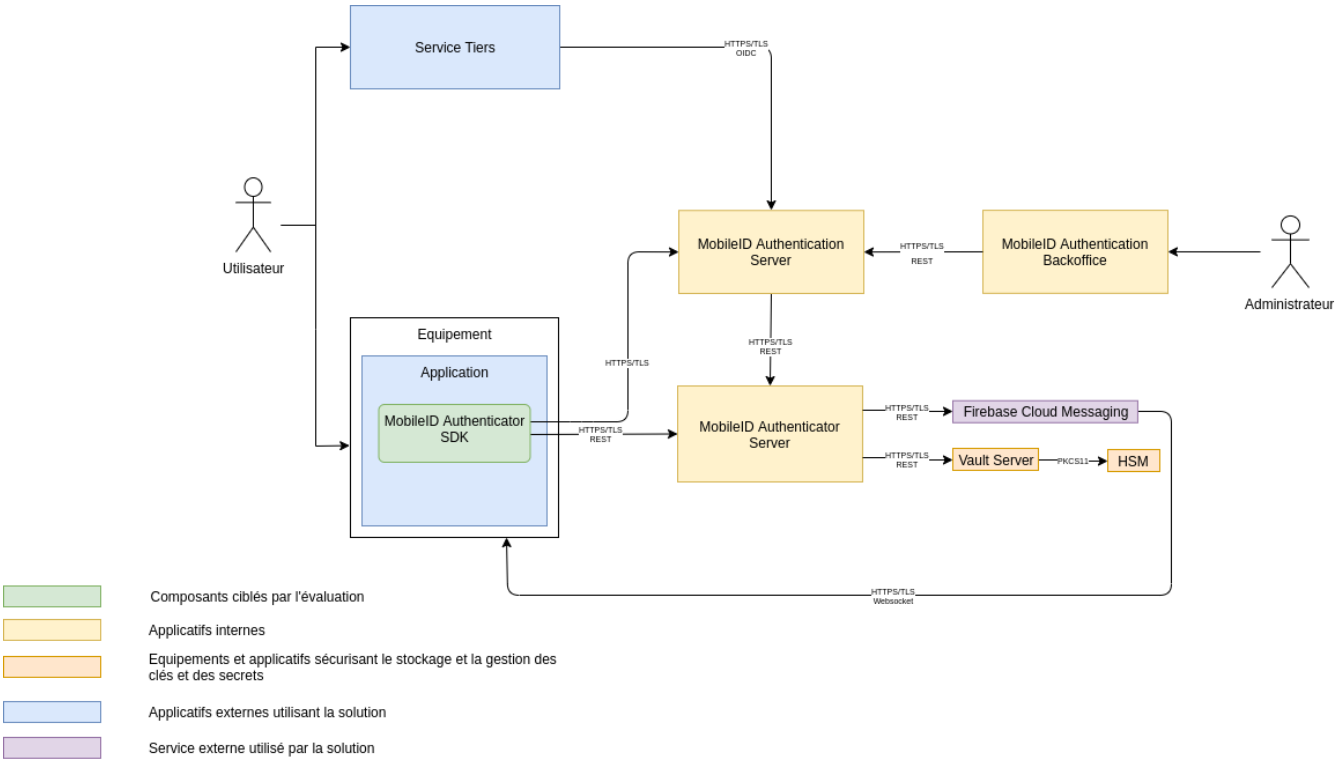

Figure 5. Architecture du système

Les composants nécessaires à l'évaluation ainsi que leur version cible sont listés Paragraphe 3.3.

Chapitre 3. Environnement Technique

3.1. Matériel compatible ou dédié

L'application mobile intégrant le "MobileID Authenticator SDK" doit être installée sur un équipement physique de type téléphone ou tablette disposant d'un système d'exploitation iOS et disposant d'interfaces de communication réseau IP connectées à Internet (via le réseau de données de l'opérateur ou par WIFI).

3.2. Système d'exploitation

Le "MobileID Authenticator SDK" doit être intégré au sein d'une application native iOS.

Il est compatible avec les versions iOS 12 et plus.

3.3. Environnement d'évaluation

Le produit est évalué sur des terminaux ayant une version 14.3 d'iOS installée.

3.4. Description des hypothèses sur l'environnement

Les hypothèses suivantes sont faites sur l'environnement d'exécution :

Identifiant	Description
H_DEVICE_SEC	L'équipement de téléphonie mobile est de confiance seulement pendant l'installation de l'application et lors de la première utilisation par l'utilisateur (enrollement). Ainsi, il est considéré que lors de cette étape ce dernier : • n'est pas jailbreaké ; • posséde un OS à jour en termes de correctifs de sécurité sur sa version ; • est exempt de codes malveillants en son sein (enregistreurs de frappes installés, logiciels de capture d'écran/vidéo, etc) ; • embarque des magasins de certificats de confiance, c'est-à-dire qu'ils contiennent les certificats des autorités racine explicitement reconnues par les constructeurs des OS mobiles. Ce magasin sera utilisé par la fabrique de sockets TLS fournie par l'application cliente au "MobileID Authenticator SDK" lors de l'établissement des connexions TLS avec le "MobileID Authenticator Server". Le certificat du nom de domaine du serveur doit, de ce fait, avoir été signé par une autorité de confiance dont le certificat ou le certificat d'une autorité intermédiaire ou racine est bien présent dans le magasin. Le terminal possède un keychain dont la clé de chiffrement est stockée sur un Secure Element (SE).

Identifiant	Description
H_APP_SEC	Il est considéré que l'application hôte est de confiance lors de l'installation et lors de la première utilisation par l'utilisateur. Le code d'activation est consommé avant l'association du moyen d'authentification à l'identifiant de connexion. L'application hôte intègre un clavier virtuel affichant l'ensemble des chiffres à l'écran dans un ordre aléatoire. L'ordonnancement des chiffres est fourni par le "MobileID Authenticator SDK". Le chiffre saisi ne reste affiché que quelques secondes à l'écran complexifiant l'espionnage. L'application hôte invite l'utilisateur à ne pas choisir de code confidentiel trivial et affiche un message d'avertissement lui demandant de ne jamais communiquer ce code à un tiers. L'application hôte embarque dans un fichier de configuration standard de l'application, l'empreinte de la clé publique du certificat du serveur. L'application hôte référence certains biens sensibles nécessaires à la configuration du "MobileID Authenticator SDK" sous une forme chiffrée : • L'algorithme de génération de la clé de chiffrement est intégré au sein des binaires applicatifs qui seront obfusqués. • La clé est reconstruite uniquement pendant un temps limiter pour initialiser le "MobileID Authenticator SDK". Parmi les biens sensibles référencés, l'application fournit une clé d'application au "MobileID Authenticator SDK" de façon à ce que le serveur puisse vérifier l'authenticité des requêtes émises par ce dernier dans le but, notamment, de les associer à une application légitime. L'application hôte reçoit le message envoyé par le service de notification indiquant qu'une demande d'authentification est en cours. Il doit appeler le "MobileID Authenticator SDK" pour lui indiquer de récupérer les informations associées à cette demande.
H_DISTRIB_SEC	Le "MobileID Authenticator SDK" est distribué sous la forme d'une archive Android signée auprès des clients devant l'intégrer au sein d'une application mobile utilisée comme moyen d'authentification. La signature est vérifiée au moment du build et du packaging de l'application cliente.
H_SRV_SEC	Il est considéré que le "MobileID Authenticator Server" avec lequel communique le "MobileID Authenticator SDK" est de confiance.
H_USR_SEC	L'utilisateur est sensibilisé sur la nécessité de définir un code confidentiel non trivial et de le changer régulièrement.

Chapitre 4. Description des biens sensibles

Par définition, un bien sensible est une donnée (ou fonction) jugée comme ayant de la valeur pour la TOE. Sa valeur est estimée selon des critères de sécurité (aussi appelés besoins de sécurité) : disponibilité, intégrité, confidentialité et authenticité.

Les biens sensibles manipulés par le produit "MobileID Authenticator SDK" pour la création du moyen d'authentification et lors de la phase d'authentification sont les suivants :

4.1. Identifiant du moyen d'authentification

Identifiant	**SA_AUTH_MEAN_ID**
Description	L'identifiant du moyen d'authentification est généré par le serveur lors de la phase de création du moyen d'authentification. Il est réutilisé par le "MobileID Authenticator SDK" dans la requête permettant d'obtenir la liste des demandes d'authentification en cours auprès du serveur.
Besoin de sécurité	Intégrité, Confidentialité

4.2. Identifiant du terminal

Identifiant	**SA_AUTH_DEVICE_ID**
Description	L'identifiant du terminal est une valeur générée par le "MobileID Authenticator SDK" à partir de données matérielles du terminal et logicielles de l'application l'intégrant. Il est transmis au serveur lors de la phase de création du moyen d'authentification afin d'être associé à ce dernier.
Besoin de sécurité	Intégrité, Confidentialité

4.3. Code confidentiel de l'usager

Identifiant	**SA_AUTH_CODE, SA_KNL_ENCRYPT_KEY**
Description	Le code confidentiel de l'usager est utilisé par l'usager pour s'authentifier. Ce code est défini lors de la phase de création du moyen d'authentification. Il est utilisé par le "MobileID Authenticator SDK" pour dériver la clé SA_KNL_ENCRYPT_KEY utilisée pour chiffrer SA_KEY_KNOWLEDGE et déchiffrer SA_ENC_KEY_KNOWLEDGE.
Besoin de sécurité	Intégrité, Confidentialité

4.4. Biclé générée lors de l'activation sur l'équipement

Identifiants	**SA_CLIENT_PUBLIC_KEY, SA_CLIENT_PRIVATE_KEY, SA_ENC_CLIENT_PRIVATE_KEY, SA_D-ENC_CLIENT_PRIVATE_KEY**

Description	La biclé générée par le "MobileID Authenticator SDK" lors de la phase de création du moyen d'authentification est utilisée dans l'échange de clés ECDH du schéma ECIES avec le serveur pour que chacun puissent générer le secret partagé SA_KEY_MASTER_SECRET : • Le "MobileID Authenticator Server" utilise SA_CLIENT_PUBLIC_KEY et ACTV_SERVER_PRIVATE_KEY ; • Le "MobileID Authenticator SDK" utilise SA_CLIENT_PRIVATE_KEY, SA_ACTV_SERVER_PUBLIC_KEY.
Besoin de sécurité	Intégrité, Confidentialité

4.5. Clé maîtresse

Identifiant	**SA_KEY_MASTER_SECRET**
Description	La clé maîtresse est un secret partagé entre le client et le serveur. Elle est générée à partir d'un échange de clés ECDH et utilisée pour dériver de nouvelles clés symétriques (SA_KEY_POSSESSION, SA_KEY_KNOWLEDGE, SA_KEY_TRANSPORT, SA_KEY_ENCRYPTION_VAULT).
Besoin de sécurité	Intégrité, Confidentialité

4.6. Clé de chiffrement du conteneur sécurisé

Identifiant	**SA_KEY_CONTAINER**
Description	Un conteneur de données sécurisé iOS (keychain) est utilisé pour stocker les clés cryptographiques (SA_ENC_KEY_POSSESSION, SA_ENC_KEY_KNOWLEDGE, SA_ENC_KEY_TRANSPORT, SA_ENC_CLIENT_PRIVATE_KEY), le jeton HMAC du terminal (SA_HMAC_TOKEN) et les biens sensibles SA_CTR_DATA et SA_ACTIVATION_ID. Toutes les données insérées au sein de ce composant sont chiffrées par une clé de chiffrement, SA_KEY_CONTAINER, gérée par iOS et stockée au du SE du terminal.
Besoin de sécurité	Intégrité, Confidentialité

4.7. Clé de signature associée au facteur de possession

Identifiant	**SA_KEY_POSSESSION, SA_ENC_KEY_POSSESSION, SA_D-ENC_KEY_POSSESSION**
Description	La clé de signature associée au facteur de possession est dérivée de la SA_KEY_MASTER_SECRET. Elle est utilisée dans l'algorithme de signature de la demande d'authentification et considérée comme premier facteur d'authentification (possession du terminal).
Besoin de sécurité	Intégrité, Confidentialité

4.8. Clé de signature associée au facteur de connaissance

Identifiant	**SA_KEY_KNOWLEDGE, SA_ENC_KEY_KNOWLEDGE, SA_D-ENC_KEY_KNOWLEDGE**
Description	La clé de signature associée au facteur de connaissance est dérivée de la SA_KEY_MASTER_SECRET. Elle est utilisée dans l'algorithme de signature de la demande d'authentification et considérée comme second facteur d'authentification (connaissance du code confidentiel).
Besoin de sécurité	Intégrité, Confidentialité

4.9. Clé de transport

Identifiant	**SA_KEY_TRANSPORT, SA_ENC_KEY_TRANSPORT, SA_D-ENC_KEY_TRANSPORT**
Description	La clé de transport est dérivée de la SA_KEY_MASTER_SECRET. Elle est utilisée par le "MobileID Authenticator SDK" pour déchiffrer les informations relatives au statut d'activation qui lui sont transmises par le serveur.
Besoin de sécurité	Intégrité, Confidentialité

4.10. Clé de stockage

Identifiant	**SA_KEY_ENCRYPTION_VAULT**
Description	La clé de stockage est dérivée de la SA_KEY_MASTER_SECRET. Elle est utilisée par le "MobileID Authenticator SDK" pour chiffrer/déchiffrer la SA_CLIENT_PRIVATE_KEY.
Besoin de sécurité	Intégrité, Confidentialité

4.11. Clé d'application pour les API

Identifiant	**SA_APP_API_KEY**
Description	La clé d'application identifie l'application intégrant le "MobileID Authenticator SDK" auprès du Server. Elle est positionnée dans les requêtes émises par le "MobileID Authenticator SDK" pour accéder à certains services exposés par le serveur. Ces services sont ceux permettant l'activation du moyen d'authentification et la vérification de la signature.
Besoin de sécurité	Intégrité, Confidentialité

4.12. Secret de l'application pour chiffrement ECIES

Identifiant	**SA_APP_SECRET_ECIES**

Description	Le secret de l'application intervient dans le calcul d'un MAC dans le chiffrement ECIES. Il permet de s'assurer que l'échange de clés est bien réalisé par une application légitime.
Besoin de sécurité	Intégrité, Confidentialité

4.13. Clé publique du serveur pour le chiffrement dans le schéma ECIES

Identifiant	**SA_KEY_SERVER_MASTER_PUBLIC**
Description	La clé publique du serveur est utilisée par le client pour générer une clé symétrique servant à chiffrer sa clé publique SA_CLIENT_PUBLIC_KEY dans le schéma ECIES d'échange des clés publiques.
Besoin de sécurité	Intégrité, Confidentialité

4.14. Données d'activation du moyen d'authentification

Identifiant	**SA_ACTIVATION_ID**
Description	L'identifiant d'activation du moyen d'authentification est généré par le "MobileID Authenticator Server". Il est transmis dans les échanges entre le "MobileID Authenticator SDK" et server pour que ce dernier puisse savoir pour quelle activation est réalisée la signature.
Besoin de sécurité	Intégrité

Identifiant	**SA_ACTIVATION_CODE**
Description	Le code d'activation du moyen d'authentification est généré par le "MobileID Authenticator Server". Il est transmis via l'application au "MobileID Authenticator SDK" qui génère sa biclé puis renvoie le code avec la clé publique pour que le serveur puisse l'associer à l'activation en attente de validation.
Besoin de sécurité	Intégrité

Identifiant	**SA_CTR_DATA**
Description	Un compteur est maintenu côté client et côté serveur. Ce compteur est mis à jour dès qu'une signature valide est transmise par le client au serveur. En fonctionnement nominal, client et serveur référencent la même valeur. Ce compteur intervient dans le calcul de la signature.
Besoin de sécurité	Intégrité, Confidentialité

4.15. Jeton HMAC

Identifiant	**SA_HMAC_TOKEN**

Description	Pour pouvoir authentifier le terminal de l'usager lors de l'accès au service permettant de récupérer la liste des demandes d'authentification en cours, une authentification basée sur un jeton HMAC a été mis en place. Le jeton, généré lors de la phase d'activation du moyen d'authentification ou lors de chaque changement de code confidentiel est utilisé pour générer un digest propre au terminal de l'utilisateur qui se retrouve dans l'entête de la requête. La valeur est vérifiée côté serveur pour authentifier le terminal et pour s'assurer qu'il est bien autorisé à accéder au service.
Besoin de sécurité	Intégrité, Confidentialité

4.16. Projet distribué du "MobileID Authenticator SDK"

Identifiant	**SA_FWK_SDK**
Description	Le "MobileID Authenticator SDK" est livré sous la forme de fichier framework iOS aux clients pour être intégré au sein d'une application mobile utilisée comme moyen d'authentification.
Besoin de sécurité	Intégrité, Authenticité

4.17. Empreinte du certificat SSL/TLS du serveur

Identifiant	**SA_SHA256_CERTIFICATE_SERVER**
Description	L'empreinte sha-256 du certificat SSL/TLS du serveur est fournie par l'application cliente au "MobileID Authenticator SDK". Lors de l'établissement des connexions SSL/TLS, cette valeur est systématiquement comparée à celle du certificat transmis par le serveur.
Besoin de sécurité	Intégrité

4.18. Clé publique associée à l'activation usager générée par le serveur

Identifiant	**SA_ACTV_SERVER_PUBLIC_KEY**
Description	La clé publique générée par le "MobileID Authenticator Server" lors de la phase de création du moyen d'authentification est utilisée dans l'échange de clés ECDH par le SDK pour générer un secret partagé la SA_KEY_MASTER_SECRET.
Besoin de sécurité	Intégrité, Confidentialité

4.19. La demande d'authentification

Identifiant	**SA_AUTH_REQUEST**

Description	La demande d'authentification est créée par le "MobileID Authenticator Server". Elle intègre des informations relatives à l'authentification en cours : • l'identifiant de la demande ; • la date de création ; • la date d'expiration ; • le nom du service auquel l'usager essaie de se connecter ; • l'état de la demande. L'identifiant de la demande est un élément qui sera intégré dans les données à signer.
Besoin de sécurité	Intégrité, Confidentialité

4.20. La signature de la demande d'authentification

Identifiant	**SA_AUTH_SIGNATURE**
Description	La signature est calculée par le "MobileID Authenticator SDK" et est envoyée au serveur pour authentifier l'usager. L'algorithme utilise les clés SA_KEY_POSSESSION, SA_KEY_KNOWLEDGE pour signer des données intégrant l'identifiant de la demande et le compteur SA_CTR_DATA.
Besoin de sécurité	Intégrité, Confidentialité

Chapitre 5. Description des menaces

Les menaces considérées sont les suivantes :

5.1. Vol de l'équipement d'authentification

Identifiant	**T_THEFT_DEVICE**
Description	L'attaquant dérobe l'équipement sur lequel une application intégrant le SDK a été installée et pour lequel l'ensemble des éléments nécessaires à l'authentification ont été activés par l'utilisateur.
Biens sensibles impactés	SA_ACTIVATION_ID, SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_CLIENT_PRIVATE_KEY, SA_SERVER_PUBLIC_KEY, SA_D-ENC_KEY_POSSESSION, SA_D-ENC_KEY_KNOWLEDGE, SA_D-ENC_KEY_TRANSPORT, SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_KEY_SERVER_MASTER_PUBLIC, SA_CTR_DATA, SA_HMAC_TOKEN, SA_SHA256_CERTIFICATE_SERVER

5.2. Vol du code confidentiel

Identifiant	**T_THEFT_AUTH_CODE**
Profil d'attaquant	Attaquant malveillant externe
Description	L'attaquant observe à distance l'utilisateur saisir son code confidentiel ou lui soustrait l'information par n'importe quel moyen d'ingénierie sociale ou technique.
Biens sensibles impactés	SA_AUTH_CODE, SA_KNL_ENCRYPT_KEY

5.3. Vol du code d'activation

Identifiant	**T_THEFT_ACTV_CODE**
Profil d'attaquant	Attaquant malveillant externe
Description	L'attaquant dérobe le code d'activation généré pour un utilisateur et l'utilise sur un autre terminal avant que celui-ci ne soit exploité par le titulaire de l'identifiant de connexion choisi. L'attaquant se retrouve donc en possession d'un moyen d'authentification associé à l'identifiant de connexion de l'usager.
Biens sensibles impactés	SA_ACTIVATION_CODE

5.4. Attaque par force brute sur les clés et secrets

Identifiant	**T_ATK_BF_AUTH_KEYS_SECRET**
Profil d'attaquant	Attaquant malveillant ayant accès au terminal

Description	Avec l'équipement d'authentification, l'attaquant effectue une attaque par force brute pour trouver les clés utilisées pour signer le challenge d'authentification, ou le code confidentiel choisi par l'usager.
Biens sensibles impactés	SA_D-ENC_KEY_KNOWLEDGE, SA_D-ENC_KEY_POSSESSION, SA_D-ENC_KEY_TRANSPORT, SA_KEY_MASTER_SECRET, SA_KEY_ENCRYPTION_VAULT, SA_D-ENC_CLIENT_PRIVATE_KEY

5.5. Clonage de l'application, des clés et secrets

Identifiant	**T_CLONE_APP**
Profil d'attaquant	Attaquant malveillant ayant accès au terminal
Description	L'attaquant ayant accès à l'équipement de l'utilisateur, pour lequel l'ensemble des éléments nécessaires à l'authentification ont été activés, clone l'application intégrant le SDK et l'espace de stockage des données.
Biens sensibles impactés	SA_ACTIVATION_ID, SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_D-ENC_CLIENT_PRIVATE_KEY, SA_SERVER_PUBLIC_KEY, SA_D-ENC_KEY_POSSESSION, SA_D-ENC_KEY_KNOWLEDGE, SA_D-ENC_KEY_TRANSPORT, SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_KEY_SERVER_MASTER_PUBLIC, SA_CTR_DATA, SA_HMAC_TOKEN, SA_SHA256_CERIFICATE_SERVER

5.6. Accès aux informations confidentielles et aux clés en clair pendant les opérations cryptographiques

Identifiant	**T_IN_MEMORY_ACCESS**
Profil d'attaquant	Attaquant malveillant ayant accès à distance au terminal
Description	Un attaquant, ayant accès à l'équipement d'authentification, extrait les informations confidentielles et les clés de signature en mémoire lors de la réalisation d'une authentification par l'utilisateur.
Biens sensibles impactés	SA_ACTIVATION_ID, SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_AUTH_CODE, SA_KNL_ENCRYPT_KEY, SA_KEY_POSSESSION, SA_ENC_KEY_POSSESSION, SA_KEY_KNOWLEDGE, SA_ENC_KEY_KNOWLEDGE, SA_APP_API_KEY, SA_CTR_DATA, SA_HMAC_TOKEN, SA_AUTH_REQUEST, SA_AUTH_SIGNATURE

5.7. Demande d'authentification trompeuse

Identifiant	**T_MALICIOUS_AUTHREQ**
Profil d'attaquant	Attaquant malveillant externe
Description	Un attaquant accédant à un service tiers utilise l'identifiant de connexion de l'utilisateur qui reçoit une demande d'authentification sur son équipement. S'il saisit son code confidentiel, l'attaquant est authentifié sur le service tiers.

Biens sensibles impactés	Aucun bien sensible de la TOE, cependant une telle attaque peut permettre à un individu malveillant d'accéder à des services en lieu et place de l'usager possédant l'identifiant de connexion.

5.8. Application malveillante intégrant le "MobileID Authenticator SDK"

Identifiant	**T_MALICIOUS_APP**
Profil d'attaquant	Attaquant malveillant externe
Description	Une application malveillante intégrant le "MobileID Authenticator SDK", se fait passer pour une application existante.
Biens sensibles impactés	SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_AUTH_CODE, SA_AUTH_DEVICE_ID, SA_AUTH_MEAN_ID, SA_AUTH_REQUEST, SA_AUTH_SIGNATURE

5.9. Compromission des données échangées entre la "MobileID Authenticator SDK" et le "MobileID Authenticator Server" et rejeu applicatif

Identifiant	**T_COMPR_EXCH**
Profil d'attaquant	Attaquant malveillant externe avec accès actif au réseau
Description	Un attaquant consulte les biens sensibles de l'application lorsqu'ils sont échangés entre le "MobileID Authenticator SDK" et le "MobileID Authenticator Server". Il peut dès lors : • déclencher des demandes d'authentification au nom de l'usager et tenter de forger des signatures avec les informations collectées ; • modifier les informations reçues par l'utilisateur afin de lui faire signer des demandes d'authentification illégitimes.
Biens sensibles impactés	SA_ACTIVATION_CODE, SA_ACTIVATION_ID, SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_SERVER_PUBLIC_KEY, SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_HMAC_TOKEN, SA_AUTH_REQUEST, SA_AUTH_SIGNATURE

5.10. Capture, rejeu/modification des trames réseau

Identifiant	**T_RPL_NET_FRAME**
Profil d'attaquant	Attaquant malveillant externe avec accès actif au réseau
Description	Un attaquant capture les trames réseaux émises lors d'une transaction légitime (création du moyen d'authentification, authentification) et tente de les rejouer telles quelles ou après les avoir modifiées afin d'abuser le système.

Biens sensibles impactés	SA_ACTIVATION_CODE, SA_ACTIVATION_ID, SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_SERVER_PUBLIC_KEY, SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_HMAC_TOKEN, SA_AUTH_REQUEST, SA_AUTH_SIGNATURE

5.11. Usurpation du serveur

Identifiant	**T_USRP_SRV**
Profil d'attaquant	Attaquant malveillant externe avec accès actif au réseau
Description	Un attaquant redirige les flux de l'équipement de l'utilisateur et tente de se faire passer pour le "MobileID Authenticator Server".
Biens sensibles impactés	SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_ACTIVATION_ID, SA_ACTIVATION_CODE, SA_CLIENT_PUBLIC_KEY, SA_KEY_POSSESSION, SA_KEY_KNOWLEDGE, SA_KEY_TRANSPORT, SA_KEY_ENCRYPTION_VAULT, SA_APP_API_KEY, SA_HMAC_TOKEN

5.12. Compromission des services de notification

Identifiant	**T_COMPR_NOTIF_SRV**
Profil d'attaquant	Attaquant malveillant externe ayant accès au service de notification
Description	Un attaquant a accès et peut modifier l'ensemble des informations transitant par les services de notification. Il peut aussi forger de nouvelles notifications à destination de l'usager. Par ce biais, il peut induire l'usager en erreur sur la nature de l'opération (authentification) en cours, se faire passer pour le gestionnaire service et lui demander certaines informations confidentielles telles que son code, simuler une demande d'authentification pour le contraindre à saisir son code.
Biens sensibles impactés	Non applicable

5.13. Rétro-ingénierie du "MobileID Authenticator SDK"

Identifiant	**T_RV_ENG**
Profil d'attaquant	Attaquant malveillant externe ou ayant accès au terminal
Description	Un attaquant récupère l'application intégrant le "MobileID Authenticator SDK" et en extrait les éléments afin de les étudier pour en déterminer leur fonctionnement interne et d'en faire ressortir les principaux mécanismes. A partir de ces informations, il est capable d'en identifier les faiblesses, les éventuelles failles qu'il exploitera, monter des attaques sur le moyen d'authentification ou les services exposés par le "MobileID Authenticator Server".
Biens sensibles impactés	Non applicable

5.14. Version altérée du "MobileID Authenticator SDK" distribuée

Identifiant	**T_ALT_SDK**
Profil d'attaquant	Attaquant malveillant externe
Description	Un attaquant fournit à une société développant une application mobile intégrant le "MobileID Authenticator SDK" une version modifiée du composant, compromettant de ce fait le service d'authentification. Cela pourra lui permettre d'usurper l'identité des usagers, de récupérer leurs informations personnelles, de mystifier le service …
Biens sensibles impactés	SA_AUTH_MEAN_ID, SA_AUTH_DEVICE_ID, SA_AUTH_CODE, SA_KNL_ENCRYPT_KEY, SA_CLIENT_PUBLIC_KEY, SA_CLIENT_PRIVATE_KEY, SA_CLIENT_PUBLIC_KEY, SA_KEY_MASTER_SECRET, SA_KEY_POSSESSION, SA_ENC_KEY_POSSESSION, SA_KEY_KNOWLEDGE, SA_ENC_KEY_KNOWLEDGE, SA_KEY_TRANSPORT, SA_ENC_KEY_TRANSPORT, SA_KEY_ENCRYPTION_VAULT, SA_APP_API_KEY, SA_APP_SECRET_ECIES, SA_KEY_SERVER_MASTER_PUBLIC, SA_ACTIVATION_ID, SA_ACTIVATION_CODE, SA_CTR_DATA, SA_HMAC_TOKEN, SA_PROJECT_SDK, SA_SHA256_CERTIFICATE_SERVER, SA_ACTV_SERVER_PUBLIC_KEY, SA_AUTH_REQUEST, SA_AUTH_SIGNATURE

Chapitre 6. Description des fonctions de sécurité

6.1. Protection du code confidentiel

Identifiant	**FSEC_PROTECT_AUTH_CODE**
Description	Le code confidentiel de l'usager SA_AUTH_CODE est protégé du vol au moyen de 3 mesures complémentaires : • La génération d'une suite de chiffres allant de 0 à 9 dans un ordre aléatoire et qui sera utilisée par l'application pour peupler un clavier virtuel ; • La fonction de renouvellement du code confidentiel (l'ancien code doit être connu de l'usager pour en définir un nouveau) ; • L'absence de stockage du code confidentiel sur le terminal, le code étant uniquement utilisé pour générer la clé SA_KNL_ENCRYPT_KEY servant à chiffrer la clé de signature associée au facteur de connaissance sur le terminal (SA_KEY_KNOWLEDGE).

6.2. Protection des données échangées

Identifiant	**FSEC_PROTECT_STREAMS**
Description	L'intégralité des flux entre les différents composants constituant la solution "MobileID Authenticator SDK et Server" sont protégés en confidentialité, intégrité, authenticité et ce dès la phase d'activation. Un canal sécurisé TLS (à minima en version 1.2) avec des suites cryptographiques robustes est mis en place. Le certificat présenté par le serveur doit être signé par une autorité de certification reconnue. De plus, l'empreinte du certicat présenté par le serveur doit correspondre à celle attendue par l'application cliente. Par ailleurs, un mécanisme de chiffrement ECIES est mis en oeuvre pour l'échange de certains biens sensibles (notamment les clés publiques échangées utilisées pour générer le secret partagé via ECDH). D'autre part, des mesures visant à authentifier l'application mobile intégrant le "MobileID Authenticator SDK" sont également mises en places notamment lors de l'accès aux services sensibles exposés par le "MobileID Authenticator Server". Lors des appels nécessaires au processus d'activation ou lors d'une authentification, l'application mobile, positionne systématiquement une clé d'application dans les requêtes émises (SA_APP_API_KEY).

6.3. Protection des clés et secrets

Identifiant	**FSEC_PROTECT_KEYS_SECRETS**

Description	L'ensemble des clés cryptographiques, secrets et jetons nécessaires au bon fonctionnement de "MobileID Authenticator SDK" sont protégés en confidentialité. Les clés cryptographiques persistées sont systématiquement chiffrées et stockées dans le keychain iOS à la demande du "MobileID Authenticator SDK". Toutes les données insérées au sein de ce composant sont chiffrées par la clé, SA_KEY_CONTAINER générée et gérée par iOS, et stockée dans le SE du terminal. Il y a donc deux niveaux de chiffrement pour les clés. Cela concerne : • La clé privée SA_CLIENT_PRIVATE_KEY de la biclé générée lors de l'activation sur l'équipement ; • la clé de signature associée au facteur de possession lors de l'authentification SA_KEY_POSSESSION, • la clé de signature associée au facteur de connaissance lors de l'authentification SA_KEY_KNOWLEDGE, • la clé de transport utilisée pour déchiffrer les informations sur le statut d'activation SA_KEY_TRANSPORT. Certaines clés utilisées lors la phase de création du moyen d'authentification et n'étant plus utilisées ultérieurement ne sont pas persistées. Il s'agit : • la clé publique SA_CLIENT_PUBLIC_KEY de la biclé générée lors de l'activation sur l'équipement ; • la clé maîtresse SA_KEY_MASTER_SECRET générée à partir de la clé privée de l'équipement et de la clé publique associée à l'activation côté serveur ; • la clé de stockage utilisée pour chiffrer/déchiffrer des informations secrètes sur l'équipement SA_KEY_ENCRYPTION_VAULT. Le jeton HMAC est lui aussi stocké dans le keychain iOS à la demande du "MobileID Authenticator SDK". La clé de l'application SA_APP_API_KEY et le secret SA_APP_SECRET_ECIES sont stockés chiffrés au sein d'un fichier de configuration de l'application. Les opérations cryptographiques réalisées par le "MobileID Authenticator SDK" sont implémentées au sein d'un composant isolé, rapide en exécution ce qui complexifie les attaques en mémoire. Par ailleurs, les clés et les résultats de calculs intermédiaires associés sont immédiatement effacés de la mémoire dès qu'ils ne sont plus nécessaires.

6.4. Protection contre les demandes d'authentification illégitimes

Identifiant	**FSEC_PROTECT_UNAUTH**

Description	Les demandes d'authentification illégitimes sont contrées par : • Le fait que le "MobileID Authenticator SDK" génère une signature quelle que soit la valeur du code confidentiel saisi par l'usager, valide ou non. Si le nombre de signatures reçues par le système atteint le seuil défini, le moyen d'authentification est bloqué côté serveur ; • La transmission à l'usager via le SDK d'informations associées à une demande d'authentification effectuée en son nom : date de la demande, nom du service à l'origine de la demande et temps restant. Par ailleurs, la notification transitant par le service de notification indiquant à l'application qu'une demande d'authentification est en cours, n'intègre qu'un texte court ne contenant aucune information sensible. Si un attaquant parvenait à émettre ce type de notification, la seule conséquence pour l'usager d'un clique sur cette dernière serait l'ouverture de l'application.

6.5. Protection contre les signatures illégitimes de demandes d'authentification

Identifiant	**FSEC_PROTECT_SIGN_UNAUTH**
Description	Les signatures illégitimes de demandes d'authentification illégitimes sont contrées par l'intégration d'un compteur SA_CTR_DATA dans le calcul de la signature. Ce compteur est maintenu côté client et serveur et incrémenté dès qu'une signature valide est transmise par le client au serveur. Cette valeur est initiée lors de l'activation. En cas de tentative de rejeu d'une authentification réussie, la valeur du compteur sera désynchronisée entre ce qu'envoie l'attaquant et le serveur.

6.6. Vérification locale du certificat présenté par le serveur de gestion d'identité

Identifiant	**FSEC_CERT_PINNING**
Description	Pour tous les échanges avec le "MobileID Authenticator Server", la TOE vérifie la clé publique du certificat présentée par le serveur avec l'empreinte de la clé attendue embarquée dans le code de l'application (méthode du Public Key Pinning).

Chapitre 7. Description des fonctions de support

7.1. Vérification de l'environnement d'exécution de l'application mobile

Identifiant	**FSUP_VERIF_ENV**
Description	Le "MobileID Authenticator SDK" offre à l'application l'intégrant des fonctionnalités permettant de vérifier qu'elle est bien installée sur un OS compatible avec les prérequis techniques imposés par ARIADNEXT. Le SDK effectue des tests de vérification concernant les bibliothèques présentes sur le téléphone et les capacités du matériel. Ces tests visent à interdire l'utilisation d'un OS non intègre (le terme « jailbreaké » est généralement employé) ou n'intégrant pas des patchs de sécurité jugés critiques.

Chapitre 8. Couverture des menaces

La couverture des menaces identifiées Chapitre 5 par les fonctions de sécurité décrites Chapitre 6, les fonctions de support décrites Chapitre 7 et dans le cadre des hypothèses listées Paragraphe 3.4 est donnée dans le tableau suivant :

Menaces	**Fonctions de sécurité**
T_THEFT_DEVICE	FSEC_PROTECT_AUTH_CODE, FSEC_PROTECT_UNAUTH
T_THEFT_AUTH_CODE	H_USR_SEC, FSEC_PROTECT_AUTH_CODE
T_THEFT_ACTV_CODE	H_APP_SEC, FSEC_PROTECT_STREAMS
T_ATK_BF_AUTH_KEYS_SECRET	FSEC_PROTECT_KEYS_SECRETS
T_CLONE_APP	H_DEVICE_SEC, FSEC_PROTECT_KEYS_SECRETS
T_IN_MEMORY_ACCESS	H_DEVICE_SEC, FSEC_PROTECT_KEYS_SECRETS, FSEC_VERIF_ENV, FSUP_OBFUSC
T_MALICIOUS_AUTHREQ	FSEC_PROTECT_UNAUTH
T_MALICIOUS_APP	H_APP_SEC, FSEC_PROTECT_STREAMS
T_COMPR_EXCH	FSEC_VERIF_ENV, FSEC_PROTECT_STREAMS, FSEC_CERT_PINNING,FSEC_PROTECT_SIGN_UNAUTH
T_RPL_NET_FRAME	FSEC_VERIF_ENV, FSEC_PROTECT_STREAMS, FSEC_PROTECT_UNAUTH, FSEC_PROTECT_SIGN_UNAUTH
T_USRP_SRV	FSEC_CERT_PINNING, H_DEVICE_SEC, FSEC_PROTECT_STREAMS
T_COMPR_NOTIF_SRV	FSEC_PROTECT_UNAUTH
T_RV_ENG	La compilation standard du code Swift en mode release réalise déjà, par optimisation et suppression des symbôles, une grande partie de ce que peut réaliser, une solution d'obfuscation.
T_ALT_SDK	H_DISTRIB_SEC