Cible de Sécurité CSPN

Seald-SDK

Référence du document	Cible de Sécurité CSPN
Version du document	1.1
Date de révision	17/07/2020

1. Sommaire

2. Introduction

Identification du produit

Organisation éditrice	Seald SAS (RCS Versailles 821 438 462)
Lien vers l'organisation	https://www.seald.io/
Nom commercial du produit	Seald-SDK
Numéro de la version évaluée	2.1
Catégorie de produit	Communication sécurisée

Structure du document

Ce document est constitué de 5 parties (à l'exception de cette introduction) décrivant :

- la cible d'évaluation, *Target of Evaluation* (TOE) ;
- l'environnement d'évaluation ;
- les biens sensibles à protéger par le produit ;
- les menaces qui pèsent sur ces biens sensibles ainsi que les agents menaçant identifiés ;
- les fonctions de sécurité couvrant les menaces précédemment décrites.

Références

Identifiant	Titre	Référence	Version	Classification
CSPN	Certification de sécurité de premier niveau	ANSSI-CSPN-CER-P-01	1.1	Public
RGS_B_1	Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques	RGS_v-2-0_B1	2.0	Public

3. Description du produit

Description générale du produit

Seald est une solution permettant de chiffrer n'importe quel fichier ou email à destination de personnes possédant la solution Seald ou non. Une fois sécurisé par Seald, le fichier ou email peut être transféré par n'importe quel moyen (email, FTP, SSH, plateforme de téléchargement, P2P, clé USB, …) sans aucune crainte qu'un quelconque intermédiaire ne soit en mesure de le lire.

Une fois sécurisé, le fichier ou email transitera dans un format propriétaire, journalisant toutes les actions effectuées sur celui-ci (tentatives d'ouverture, ouvertures avec succès, échecs d'ouverture, …), que le destinataire possède la solution Seald, ou non.

La particularité de notre approche réside dans le fait de ne pas effectuer le transfert du document. Seald s'appuie sur les outils déjà utilisés par les collaborateurs, ne changeant ainsi pas leurs habitudes.

De plus en dissociant la solution de sécurisation de celle de transfert et/ou stockage, une compromission du stockage n'entrainera d'aucune façon la compromission de l'intégrité et la confidentialité des fichiers, contrairement à d'autres modèles où la confiance cryptographique est implémentée dans le même service que celui de stockage.

Description de la manière d'utiliser le produit

Dans le cadre de cette évaluation, Seald prend la forme d'une bibliothèque *NodeJS* pouvant être intégrée dans un projet afin d'échanger des fichiers et des messages de manière sécurisée.

Cette bibliothèque est nommée *Seald-SDK* et contient les ressources nécessaires à la création des clés, à leur gestion et aux déchiffrements et chiffrements de fichiers.

L'utilisation du produit requiert la création d'un client *NodeJS* appelant les API de *Seald-SDK* pour réaliser ces diverses opérations de chiffrement.

Description technique du produit

Le produit est architecturé dans un mode client-serveur.

La bibliothèque *Seald-SDK*

Sous la forme d'une bibliothèque, cette entité est chargée :

- de la génération des clés privées : une clé privée de signature et une clé privée de chiffrement ;
- de leur renouvellement ;
- du chiffrement et du déchiffrement de fichiers ;
- de maintenir une base de données locale contenant les différentes chaînes de signatures et les paires de clés publiques des destinataires connus.

Seule cette partie figure dans le périmètre de l'évaluation de CSPN.

Le serveur *Beard*

Le serveur *Beard* exposant une API REST qui centralise et maintient :

- l'annuaire des utilisateurs ;
- les différentes chaînes de signatures (*sigchain*), voir plus bas ;
- les clés symétriques des messages, chiffrées par les clés publiques des destinataires.

Il est à noter que le périmètre d'évaluation prend en compte la possibilité d'un serveur malveillant ou compromis.

Le serveur *Dashboard*

Le serveur *Dashboard* est une application web permettant aux administrateurs Seald de :

- gérer les équipes d'utilisateurs ;
- révoquer les utilisateurs, clés et fichiers ;
- consulter des événements comme la lecture d'un message ;

Ce serveur interagit avec le serveur *Beard* pour transmettre aux clients utilisant *Seald-SDK* les informations d'équipe, de révocation ou de clés administrateur.

Chiffrement d'un fichier

Le chiffrement d'un fichier suit les étapes suivantes :

1. La bibliothèque *Seald-SDK* génère deux clés symétriques. L'une sera utilisée pour chiffrer le contenu de l'archive tandis que la seconde sera utilisée pour réaliser un code d'authentification de message.
2. La bibliothèque *Seald-SDK* demande les adresses e-mails ou les identifiants Seald des destinataires à l'utilisateur.
3. La bibliothèque *Seald-SDK* envoie une requête au serveur *Beard* pour récupérer les clés publiques de chiffrement associées aux appareils des destinataires ainsi que leur chaîne de signatures.
4. La bibliothèque *Seald-SDK* vérifie l'intégrité de la chaîne de signature et valide qu'elle contient les empreintes des clés publiques reçues.
5. L'utilisateur authentifie manuellement les clés publiques reçues en validant la dernière empreinte de la chaîne de signatures (*sigchain*) pour chaque destinataire. Cette vérification n'est nécessaire qu'une seule fois.
6. La bibliothèque *Seald-SDK* chiffre les clés symétriques avec les clés publiques de chiffrement des destinataires.
7. La bibliothèque *Seald-SDK* envoie au serveur *Beard*, les clés symétriques chiffrées par chaque clé publique de chiffrement des destinataires du fichier, qui répond avec un nouvel identifiant de message.
8. Le fichier en clair est archivé au format USTAR.
9. On chiffre le contenu de l'archive avec la première clé symétrique, on calcule un code d'authentification de message sur le résultat de ce chiffrement et on concatène les deux.
10. Le résultat de cette concaténation est encapsulé dans un fichier HTML comportant en clair l'identifiant du message.

Déchiffrement d'un fichier

Le déchiffrement d'un fichier suit les étapes suivantes :

1. La bibliothèque *Seald-SDK* récupère l'identifiant du message et requête le serveur *Beard* pour obtenir les clés symétriques chiffrées avec la clé publique de chiffrement de l'appareil utilisé.
2. La bibliothèque *Seald-SDK* déchiffre la réponse du serveur et extrait les deux clés symétriques.
3. La bibliothèque *Seald-SDK* vérifie l'intégrité du message et déchiffre le contenu du message.

Chaîne de signatures *Sigchain*

Lors de la récupération des clés publiques de chiffrement (pendant une opération de chiffrement), le serveur retourne également les clés publiques de signature des appareils du destinataire ainsi qu'une chaîne de signatures. Cette chaîne de signatures trace tous les événements de création, de renouvellement et de révocation des clés d'un utilisateur.

Chaque opération dans cette chaîne est protégée en intégrité. Les opérations de création et de renouvellement sont également authentifiées par l'utilisateur, tandis que les opérations de révocation ne sont authentifiées que lorsque réalisées par l'utilisateur. Dans le cas où un administrateur révoque une paire de clés privées via le serveur *Beard*, cette opération est également ajoutée à la chaîne de signatures mais n'est pas authentifiée.

Les opérations de validation d'intégrité sont chaînées de telle sorte que seule l'empreinte du dernier bloc est à contrôler pour valider l'ensemble de la chaîne. L'authentification et la validation de l'intégrité est confirmée par une vérification manuelle réalisée par les utilisateurs. En outre, une seule vérification manuelle est nécessaire étant donné que les opérations d'ajout de clé et de renouvellement doivent être signées par une ancienne clé.

Description de l'environnement prévu pour son utilisation

L'environnement se compose donc de deux parties :

- Le client utilisant la bibliothèque *Seald-SDK* ;
- Le serveur *Beard* hébergé par Seald.

Description des dépendances

L'utilisation de la bibliothèque *Seald-SDK* nécessite l'installation d'un exécuteur JavaScript. À ce titre, les composants logiciels suivants doivent être présents sur le système :

- Node.js 12.16.3 LTS ;
- npm 6.14.4.

D'autre part, les modules suivants sont nécessaires au bon fonctionnement de la bibliothèque *Seald-SDK* :

- @seald/logger v0.2.1
- @seald/hairless v0.7.0
- @seald/follicle v1.1.1
- another-json v0.2.0
- bson v4.0.4
- emittery v0.6.0
- form-data v3.0.0
- fs-jetpack v2.2.3
- nedb v1.8.0
- node-fetch 2.6.0
- pumpify v2.0.1
- set-cookie-parser v2.4.5
- sscrypto v0.4.2
- tar-fs v2.0.1
- tar-stream v2.1.2
- tough-cookie v4.0.0

Description des utilisateurs typiques concernés

Deux profils d'utilisateur existent :

- Les utilisateurs classiques qui disposent d'un client basé sur la bibliothèque *Seald-SDK* et qui s'échangent des fichiers.
- Les administrateurs qui ont accès au *dashboard* Seald.

Description des hypothèses sur l'environnement

H.1 (H_CLIENT_TRUSTED)

L'appareil de l'utilisateur hébergeant le client utilisant la bibliothèque *Seald-SDK* est bien administré, sain et de confiance.

H.2 (H_DEVICE_KEYS)

Les clés privées générées sur un poste sain, et de confiance, tel que décrit dans les fournitures cryptographiques. Elles sont stockées sur l'appareil de l'utilisateur qui est considéré comme sain et de confiance.

H.3 (H_SIGCHAIN_MANUAL_CHECK)

Deux utilisateurs souhaitant s'échanger des documents, vérifient manuellement l'empreinte du dernier bloc de leur *sigchain* respective via un canal auxiliaire sûr.

H.4 (H_ADMIN_KEYS_NOT_USED)

Les utilisateurs n'acceptent pas de clé d'administration, et n'acceptent pas de générer pas de clé de récupération.

H.5 (H_TRANSFER_PUBLIC_KEYS)

Lors de l'ajout d'un nouvel appareil, la clé publique de celui-ci doit être importée dans un ancien appareil afin d'être déclaré comme appareil valide. Ce transfert de clé publique est réalisé manuellement par l'utilisateur et est supposé intègre et authentifié.

Définition du périmètre de l'évaluation

Tout d'abord, les éléments suivants ne rentrent pas dans le périmètre d'évaluation :

- le serveur *Beard* ;
- le *dashboard* Seald et les mécanismes de journalisation, de révocation et *licensing* ;
- le stockage des fichiers chiffrés qui n'est pas réalisé par Seald ;
- le déchiffrement et le chiffrement de fichiers par des utilisateurs sans Seald ;
- la création d'une clé administrateur et l'utilisation des clés administrateur

L'implémentation pour Node.js de la bibliothèque *sscrypto* en version 0.4.2 ainsi que la bibliothèque *Seald-SDK* sont dans le périmètre de l'évaluation avec les arguments de *Seald-SDK* respectant les conditions suivantes :

- l'option *strict-mode* activée. Cette option ajoute des exigences supplémentaires sur le comportement de *Seald-SDK* ;
- le paramètre *keySize* à 4096. Cette option définit, conformément aux fournitures cryptographiques fournies, la taille du module des clés asymétriques à 4096 bits ;
- le paramètre *sscrypto* injectant le module *'sscrypto/node'* de la bibliothèque *sscrypto* en version 0.4.2 implémentant une partie des mécanismes cryptographiques décrits dans les fournitures cryptographiques.

Dans le cadre de cette certification, la génération, la gestion et l'utilisation des clés d'administration est hors périmètre. Le mécanisme de récupération s'appuyant sur ces clés d'administration, ce mécanisme est également hors périmètre bien que disponible en mode strict. On devra néanmoins s'assurer que la bibliothèque *Seald-SDK* n'ajoute pas des clés de récupération sans l'aval de l'utilisateur.

4. Description de l'environnement technique dans lequel le produit doit fonctionner

Prérequis matériel

Un système en mesure d'installer les dépendances décrites dans la partie Description des dépendances page 2.

Le client *Seald-Cli* sera utilisé. Il s'agit d'une interface en ligne de commande développée par Seald qui utilise la bibliothèque *Seald-SDK.*

Système d'exploitation retenu

Le système Debian 10 *buster* est retenu pour la réalisation de l'évaluation.

Configurations type

La bibliothèque *Seald-SDK* sera configurée avec les options suivantes :

- *strict-mode* activé ;
- *api-url* : URL donnée par les équipes de Seald, correspondant au serveur *Beard* ;
- *keySize* : 4096 ;
- *sscrypto :* le module *'sscrypto/node'* de la bibilothèque *sscrypto* en version 0.4.2.

5. Description des biens sensibles que le produit doit protéger

Les biens sensibles de la solution Seald, compte tenu du périmètre à évaluer.

Biens métiers

B.1 (B_DOCUMENTS)

Les documents échangés par les utilisateurs de Seald. Ces documents doivent être protégés en confidentialité et en intégrité.

Biens cryptographiques

B.2 (B_SIGNING_KEYS)

Les clés privées de signature d'un utilisateur sur ses équipements. Ces clés doivent être protégées en confidentialité.

B.3 (B_ENCRYPTION_KEYS)

Les clés privées de chiffrement d'un utilisateur sur ses équipements. Ces clés doivent être protégées en confidentialité.

B.4 (B_SYMMETRIC_KEYS)

Les clés symétriques utilisées pour le chiffrement et le code d'authentification des documents. Ces clés doivent être protégées en confidentialité.

B.5 (B_SIGCHAIN)

La chaîne de signature d'un utilisateur qui permet d'authentifier ses clés publiques de signature et de chiffrement. La chaîne de signature doit être protégée en intégrité et en authenticité pour les blocs de renouvellement, et de création de clés.

B.6 (B_NEW_DEVICE_PUB_KEYS)

Les clés publiques (de chiffrement et de signature) d'un nouvel appareil non encore enregistré dans la chaine de signatures d'un utilisateur. Ces clés doivent être protégées en intégrité et en authenticité.

Le tableau suivant résume les besoins de protection pour les différents biens sensibles identifiés :

	Confidentialité	Intégrité	Authenticité	Disponibilité
B.1 B_DOCUMENTS	X	X		
B.2 B_SIGNING_KEYS	X			
B.3 B_ENCRYPTION_KEYS	X			
B.4 B_SYMMETRIC_KEYS	X			
B.5 B_SIGCHAIN		X	X	
B.6 B_NEW_DEVICE_PUB_KEYS		X	X	

Il est à noter que les métadonnées associées à un échange de documents ne sont pas considérées comme des biens sensibles. Ces métadonnées comprennent :

- expéditeur ;
- destinataire(s) ;
- nom du fichier ;

- état du document (lu, révoqué).

Enfin, les biens *B.2 à B.6* sont des biens supports à *B.1* qui est le bien métier à protéger. En effet, si l'un de ces biens venait à être compromis, le bien *B.1* le serait également.

6. Description des menaces

Agents menaçants

Les agents menaçants comprennent toute personne souhaitant déchiffrer un message sans en être le destinataire.

Plusieurs postures d'un tel attaquant sont considérées :

AM.1 (AM_SENDER_MITM)

Un attaquant en position de *Man-In-The-Middle* entre l'expéditeur et le serveur Seald, et en possession du document chiffré au format *SHTML.*

AM.2 (AM_RECEIVER_MITM)

Un attaquant en position de *Man-In-The-Middle* entre le serveur Seald et le destinataire, et en possession du document chiffré au format *SHTML.*

AM.3 (AM_MALICIOUS_SERVER)

Un attaquant ayant compromis le serveur Seald et maîtrisant tous les échanges entre l'expéditeur et les destinataires.

Il est à noter qu'en pratique **AM.3** englobe les deux premiers agents menaçants. En outre, on considère que les agents menaçants possèdent les documents chiffrés au format *SHTML* qu'ils cherchent à déchiffrer.

Menaces

M.1 (M_DECRYPTION)

Un attaquant qui n'est pas le destinataire d'un document est en mesure de le déchiffrer.

Cette menace peut se réaliser par la compromission de différents biens supports :

- Compromission de la *sigchain* : dans ce cas, il est possible de fournir une clé publique arbitraire à l'expéditeur, qui chiffrera la clé symétrique avec cette clé publique (matérialisée par **M.6 M_SIGCHAIN_COMPROMISE**).
- Compromission de la clé privée de signature d'un destinataire qui permet de compromettre sa *sigchain* (matérialisée par **M.3 M_SIGNING_KEY_COMPROMISE**).
- Compromission de la clé privée de chiffrement d'un destinataire ou de l'expéditeur pour déchiffrer la clé symétrique servant à chiffrer le document (matérialisée par **M.4 M_ENCRYPTION_KEY_COMPROMISE**).
- Obtention de la clé symétrique de chiffrement qui permet de déchiffrer directement le document (matérialisée par **M.5 M_SYMMETRIC_KEYS_COMPROMISE)**.

M.2 (M_ALTERATION)

Un attaquant qui n'est pas le destinataire d'un document est en mesure de l'altérer, sans modifier l'identifiant du message.

Cette menace peut se réaliser par la compromission de différents biens supports :

- Obtention de la clé symétrique de chiffrement et code d'authentification qui permet de déchiffrer et modifier directement le document (matérialisée par **M.5 M_SYMMETRIC_KEYS_COMPROMISE**).
- Compromission de la clé privée de chiffrement d'un destinataire ou de l'expéditeur pour déchiffrer la clé symétrique servant à prouver l'intégrité du document (**M.4 M_ENCRYPTION_KEY_COMPROMISE**).

M.3 (M_SIGNING_KEYS_COMPROMISE)

Un attaquant parvient à compromettre la clé privée de signature d'un utilisateur.

Cette compromission lui permet d'altérer la *sigchain* et d'ajouter un nouveau couple de clés privées afin de pouvoir déchiffrer des documents.

M.4 (M_ENCRYPTION_KEYS_COMPROMISE)

Un attaquant parvient à compromettre la clé privée de chiffrement d'un utilisateur.

Cette compromission lui permet de déchiffrer les clés symétriques retournées par le serveur qui servent à chiffrer un document.

M.5 (M_SYMMETRIC_KEYS_COMPROMISE)

Un attaquant parvient à obtenir les clés symétriques de chiffrement d'un document.

Cette compromission lui permet de déchiffrer directement le document.

M.6 (M_SIGCHAIN_COMPROMISE)

Un attaquant parvient à ajouter un bloc d'ajout ou de renouvellement de clés arbitraire dans la *sigchain* d'un utilisateur, lui permettant d'ajouter une paire de clés publiques arbitraires dont il possède les clés privées.

Cette compromission lui permet ensuite de déchiffrer un document chiffré après la compromission de la *sigchain.*

M.7 (M_ADMIN_KEYS_SILENTLY_USED)

Un attaquant parvient simultanément à :

- Injecter une clé administrateur à un utilisateur ;
- Faire accepter silencieusement cette clé par l'utilisateur afin de générer une clé de récupération.

La connaissance de ces clés permet ensuite de déchiffrer n'importe quel document.

M.8 (M_DEVICE_COMPROMISE)

Un attaquant parvient à compromettre un appareil utilisateur contenant les clés privées de signature et de chiffrements.

M.9 (M_NEW_DEVICE_PUBLIC_KEYS_INTERCEPTION)

Un attaquant parvient à compromettre les clés publiques d'un nouvel appareil lors du transfert de celle-ci vers un ancien appareil. Cette compromission lui permet d'ajouter ces clés publiques dans la chaine de signatures de l'utilisateur et que les expéditeurs chiffrent les fichiers avec nouvelle cette clé publique.

Menaces planant sur les biens sensibles	
M.1 M_DECRYPTION	**B.1 B_DOCUMENTS**
M.2 M_ALTERATION	**B.1 B_DOCUMENTS**
M.3 M_SIGNING_KEYS_COMPROMISE	B.2 B_SIGNING_KEYS (directement) par extension : B5. B_SIGCHAIN B.4 B_SYMMETRIC_KEYS par conséquent : B.1 B_DOCUMENTS
M.4 M_ENCRYPTION_KEYS_COMPROMISE	**B.3 B_ENCRYPTION_KEYS** (directement) par extension : **B.4 B_SYMMETRIC_KEYS** par conséquent : **B.1 B_DOCUMENTS**
M.5 M_SYMMETRIC_KEYS_COMPROMISE	**B.4 B_SYMMETRIC_KEYS** (directement) par conséquent : **B.1 B_DOCUMENTS**
M.6 M_SIGCHAIN_COMPROMISE	**B.5 B_SIGCHAIN** (directement) par extension : **B.4 B_SYMMETRIC_KEYS** par conséquent : **B.1 B_DOCUMENTS**
M.7 M_ADMIN_KEYS_SILENTLY_USED	**B.2 B_SIGNING_KEYS** **B.3 B_ENCRYPTION_KEY** par conséquent : **B.4 B_SYMMETRIC_KEYS** **B.5 B_SIGCHAIN** **B.1 B_DOCUMENTS**
M.8 M_DEVICE_COMPROMISE	**B.2 B_SIGNING_KEYS** **B.3 B_ENCRYPTION_KEYS** (directement) par extension : **B5. B_SIGCHAIN** **B.4 B_SYMMETRIC_KEYS** par conséquent : **B.1 B_DOCUMENTS**
M.9 M_NEW_DEVICE_PUB_KEYS_TAMPERING	**B.2 B_SIGNING_KEYS** **B.3 B_ENCRYPTION_KEYS** (directement) par extension : **B5. B_SIGCHAIN** **B.4 B_SYMMETRIC_KEYS** par conséquent : **B.1 B_DOCUMENTS**

	B.1	B.2	B.3	B.4	B.5
M.1 M_DECRYPTION	**X**				
M.2 M_ALTERATION	**X**				
M.3 M_SIGNING_KEYS_COMPROMISE	I	**X**		I	I
M.4 M_ENCRYPTION_KEYS_COMPROMISE	I		**X**	I	

M.5 M_SYMMETRIC_KEYS_COMPROMISE	I			**X**	
M.6 M_SIGCHAIN_COMPROMISE	I			I	**X**
M.7 M_ADMIN_KEYS_SILENTLY_USED	I	**X**	**X**	I	I
M.8 M_DEVICE_COMPROMISE	I	**X**	**X**	I	I
M.9 M_NEW_DEVICE_PUB_KEYS_TAMPERING	I	**X**	**X**		

Les X correspondent aux menaces directes.

Les I correspondent aux menaces indirectes.

7. Description des fonctions de sécurité

Le produit Seald implémente les fonctions de sécurité suivantes :

FS.1 (FS_ENCRYPTED_DOCUMENTS)

Les documents sont chiffrés symétriquement avec l'algorithme AES 256 en mode CBC. Un contrôle d'intégrité est également réalisé au moyen d'un HMAC SHA-256. Deux clés différentes sont utilisées pour ces deux opérations.

Les documents sont par conséquent protégés en confidentialité et en intégrité.

Le mécanisme de chiffrement est spécifié dans les fournitures cryptographiques au paragraphe "Cryptographie symétrique".

FS.2 (FS_ENCRYPTED_SYMMETRIC_KEYS)

Les clés symétriques de chiffrement et d'intégrité sont chiffrées par les clés publiques de chiffrement des destinataires avec l'algorithme RSA et le schéma PKCS1 OAEP.

Le mécanisme de chiffrement est spécifié dans les fournitures cryptographiques au paragraphe "Chiffrement & déchiffrement asymétriques".

FS.3 (FS_SIGCHAIN)

Les opérations sur les clés des utilisateurs sont ajoutées dans une chaîne de blocs. Les opérations possibles sont les suivantes :

- création ;
- révocation ;
- renouvellement.

Les ajouts de création et de renouvellement sont authentifiés par l'utilisateur réalisant l'ajout et assure l'intégrité du bloc précédent. L'opération de révocation n'est authentifiée que lorsque réalisée par l'utilisateur, si cette opération est réalisée par le serveur *Beard* l'ajout du bloc n'est pas authentifié. Cependant, ce mécanisme n'affecte pas le rôle de cette fonction qui permet de protéger contre la création ou le renouvellement de clés arbitraires.

L'exploitation de l'absence d'authentification sur les ajouts de bloc de révocation permet seulement d'affecter la disponibilité de Seald et empêchera le déchiffrement d'un document.

Les opérations d'ajout et de vérification sont spécifiées dans les fournitures cryptographiques au paragraphe "Chaîne de signatures du trousseau d'un utilisateur".

Couverture des menaces par les fonctions de sécurité et hypothèses	
FS.1 FS_ENCRYPTED_DOCUMENTS	M.1 M_DECRYPTION M.2 M_ALTERATION
FS.2 FS_ENCRYPTED_SYMMETRIC_KEYS	M.5 M_SYMMETRIC_KEYS_COMPROMISE
FS.3 FS_SIGCHAIN H.3 H_SIGCHAIN_MANUAL_CHECK	M.6 M_SIGCHAIN
H.2 H_DEVICE_KEYS	M.3 M_SIGNING_KEY_COMPROMISE M.4 M_ENCRYPTION_KEY_COMPROMISE
H.4 H_ADMIN_KEYS_NOT_USED	M.7 M_ADMIN_KEYS_SILENTLY_USED

H.5 H_TRANSFER_PUBLIC_KEYS	M.9 M_NEW_DEVICE_PUB_KEYS_TAMPERING

	M.1	M.2	M.3	M.4	M.5	M.6	M.7	M.8
FS.1 FS_ENCRYPTED_DOCUMENTS	X	X						
FS.2 FS_ENCRYPTED_SYMMETRIC_KEYS					X			
FS.3 FS_SIGCHAIN						X		
H.1 H_CLIENT_TRUSTED								X
H.2 H_DEVICE_KEYS			X	X				
H.3 H_SIGCHAIN_MANUAL_CHECK						X		
H.4 H_ADMIN_KEYS_NOT_USED							X	
H.5 H_TRANSFER_PUBLIC_KEYS								X