



Système de Protection Logiciel



Manuel pour les Développeur

Applicable à Microsoft Windows XP à 10 (tous 32/64-Bit), CE,
Linux X86/A64/ARM et Mac OS X



Système de Protection Logiciel

Manuel pour les Développeur

Applicable à Microsoft Windows XP à 10 (tous 32/64-Bit), CE,
Linux X86/A64/ARM et Mac OS X

Mise à jour : Février 2016

SG Intec Ltd & Co. KG, Schauenburgerstr. 116, D-24118 Kiel, Germany

Fon ++49 431 97993-00 Fax ++49 431 97993-50

web : www.sg-lock.com/fr courriel : info@sg-intec.de

SG-Lock respecte la directive de l'Union Européenne sur les équipements électriques etélectro-niques. Les taxes d'élimination des déchets ont été payées. WEEE-ID : DE 39431724

Toutes les marques déposées utilisées ou représentées dans ce manuel sont la propriété exclusive de leurs titulaires.

Sous réserve de modifications techniques. La copie de ce manuel – même en extraits – n'est permise qu'avec l'autorisation écrite de SG-Intec Ltd & Co KG.

Table des matières

1	Introduction	1
2	Installation et programmes auxiliaires	3
2.1	Windows XP à 10	3
2.2	Linux	3
2.3	Mac OS X	3
2.4	Windows CE 4, 5 and 6	5
2.5	Désinstallation pour tous les systèmes	5
2.6	Modifier SG-Lock avec le manager SG-Lock	6
3	Protéger des logiciels avec SG-Lock	11
3.1	Généralités	11
3.2	Stratégies de protection	11
3.3	L'identifiant du produit – à quoi sert-il?	12
3.4	Le Chiffrement et l'authentification stimulationréponse	13
3.5	L'adaptation à différentes langues de programmation	16
4	L'API SG-Lock	17
4.1	Plan des fonctions	17
4.2	Fonctions de base	19
4.2.1	Fonction: SglAuthent	19
4.2.2	Fonction: SglSearchLock	21
4.2.3	Fonction: SglReadSerialNumber	22
4.3	Extended Fonction	23
4.3.1	Fonction: SglReadData	23
4.3.2	Fonction: SglWriteData	25
4.3.3	Fonction: SglReadCounter	26
4.3.4	Fonction: SglWriteCounter	27
4.4	Fonctions de chiffrement et de signature	28
4.4.1	Fonction: SglCryptLock	28
4.4.2	Fonction: SglSignData	30

4.5	Administrative Fonctions	33
4.5.1	Fonction: SglReadProductId	33
4.5.2	Fonction: SglWriteProductId	34
4.5.3	Fonction: SglWriteKey	35
4.5.4	Fonction: SglReadConfig	36
4.6	Valeurs de retour	38
5	Chiffrement, signature et administration de clés	39
6	Exemples de programmation	41
6.1	Fonction SglAuthent	41
6.1.1	C/C++	41
6.1.2	Delphi	41
6.1.3	Visual Basic	42
6.2	Fonction SglSearchLock	44
6.2.1	C/C++	44
6.2.2	Delphi	44
6.2.3	Visual Basic	45
6.3	Fonction SglReadSerialNumber	46
6.3.1	C/C++	46
6.3.2	Delphi	46
6.3.3	Visual Basic	47
6.4	Fonction SglReadData	48
6.4.1	C/C++	48
6.4.2	Delphi	48
6.4.3	Visual Basic	49
6.5	Fonction SglWriteData	51
6.5.1	C/C++	51
6.5.2	Delphi	51
6.5.3	Visual Basic	52
6.6	Authentification stimulation-réponse d'un SG-Lock	54
6.6.1	C/C++	54
7	Données techniques	57
7.1	SG-Lock U2/U3/U4	57

1 Introduction

Moderne et flexible, SG-Lock est un système de protection cryptographique à base matérielle qui peut être utilisé avec tous les systèmes d'exploitation Windows à 32 et 64 bits, Linux et Mac OS X.

Caractéristiques particulières :

- Chaque SG-Lock a un numéro de série individuel.
- Mémoire interne et librement utilisable jusqu'à 1024 octets
- Chiffrement 128 bits avec 16 clés librement programmables
- Jusqu'à 64 cellules compteur librement programmables pour l'enregistrement simple d'événements nombrables
- Les SG-Locks USB peuvent être installés dans le système d'arrivée sans installation d'un pilote et sans droits d'administrateur (XP à 10).

Points forts de la sécurité :

- Toute la mémoire interne est transparente pour l'utilisateur, chiffrée avec une clé 128 bits unique pour chaque module et signée en plus. Des attaques au matériel ainsi que la manipulation de valeurs ou l'échange des cellules de mémoire sont reconnus et rejetés par le processeur du module.
- Un mécanisme d'authentification simple et efficace entre l'application protégée et l'API du SG-Lock. L'API du SG-Lock se distingue par le fait qu'après son implémentation il ne peut pas être utilisé d'emblée dans sa fonctionnalité intégrale à partir de toutes les applications. En principe, chaque application doit s'authentifier envers l'API du SG-Lock afin d'avoir accès aux modules SG-Lock. Ceci empêche que des programmes non autorisés attaquent les modules de protection SG-Lock par l'interface de l'API. En plus, le programme protégé lui-même peut vérifier l'API du

SG-Lock, reconnaître et rejeter une bibliothèque truquée. Le mécanisme d'authentification complet est exécuté par l'appel d'une seule fonction à un seul paramètre.

- L'API du SG-Lock se sert d'un engin de chiffrement TEA (Tiny Encryption Algorithm) interne par rapport au module ainsi qu'à l'application. Cet algorithme de chiffrement symétrique (les clés pour le chiffrement et le déchiffrement sont identiques) et dont le haut degré de sécurité est généralement reconnu constitue la base pour l'implémentation de diverses stratégies de protection et d'authentification.

2 Installation et programmes auxiliaires

L'installation de SG-Lock s'effectue de manière simple et transparente afin de faciliter l'intégration dans l'installation de l'application protégée. Le procédé varie selon que SG-Lock doit être utilisé uniquement dans sa version USB ou la version LPT ou bien dans la combinaison des modules.

2.1 Windows XP à 10

L'installation des modules SG-Lock USB se déroule en deux étapes.

1. Copiez la bibliothèque SGLW32.DLL appartenant à SG-Lock dans le dossier d'installation de l'application protégée ou dans le dossier système (par ex. C:\WINDOWS\SYSTEM32 pour Windows XP à 10 - il faut tenir compte des droits d'inscription !).
2. Connectez le SG-Lock USB au port USB. Sous Windows XP à 10 l'identification du matériel s'effectue et s'affiche automatiquement. Après cela, cette installation est également complète, et le SG-Lock est prêt à fonctionner.

2.2 Linux

Les instructions d'installation pour Linux peuvent être trouvées sur la CD-ROM de SG-Lock.

2.3 Mac OS X

Les instructions d'installation pour MAC OS X peuvent être trouvées sur la CDROM de SG-Lock.

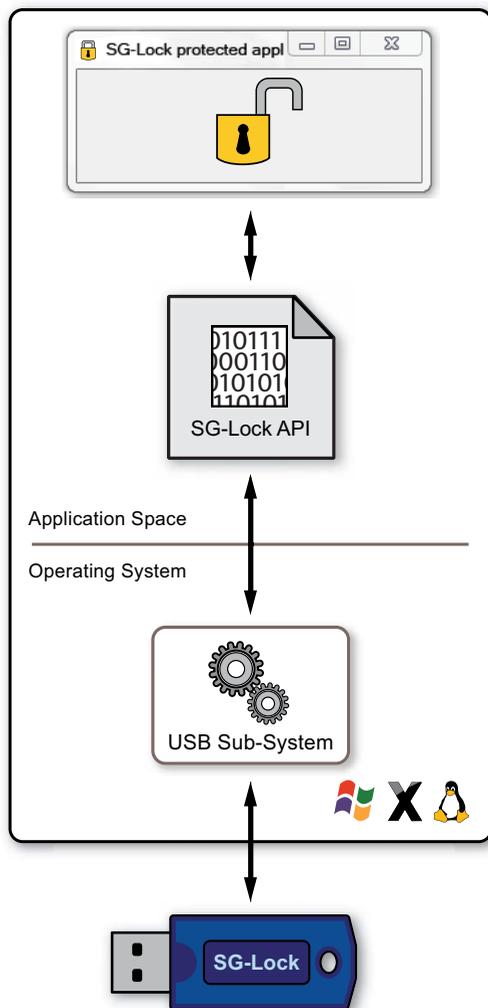


FIGURE 2.1 : L'API de SG-Lock (le fichier SGLW32.DLL) établit la connexion entre l'application protégée et le matériel SG -Lock.

2.4 Windows CE 4, 5 and 6

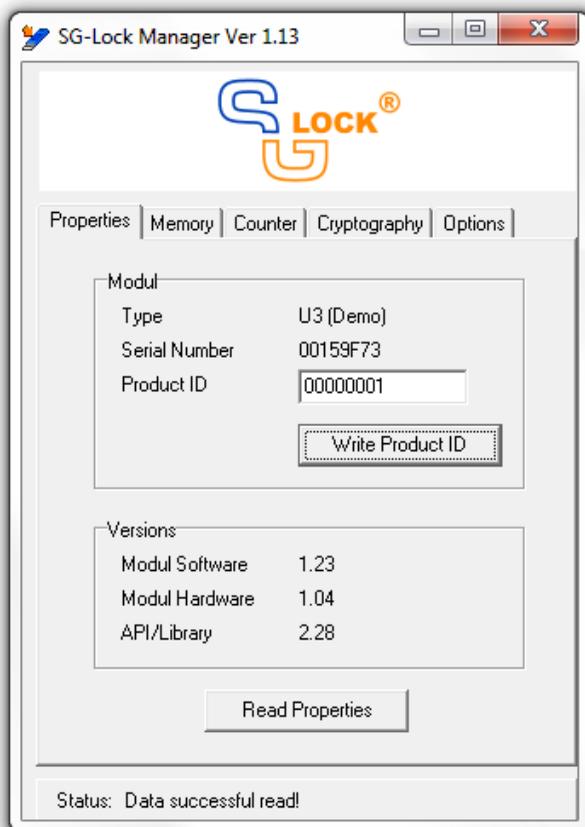
1. Copiez le fichier SGLWCE.DLL dans le dossier de votre application ou dans le dossier système (par ex. \Windows). Ceci peut se faire par un script au démarrage du système.
2. Copiez le fichier SGLUSB.DLL dans un dossier déjà existant lors du démarrage du système (par ex. \STORAGE).
3. Adaptez les deux clés au nom de DLL dans le script de registre SGLUSB.REG au chemin de SGLUSB.DLL (si par ex. votre SGLUSB.DLL se trouve dans \STORAGE, les valeurs des deux clés doivent être STORAGE\SGLUSB.DLL). N'utilisez pas de back-slash. Ne changez pas les clés PREFIX.
4. Lancez le script de registre adapté et mémorisez le registre (par ex. avec AP CONFIG MANAGER dans la fiche APSystem, icône STORGE/REGISTRY/SAVE), afin que les clés soient gardées pour les démarrages ultérieurs.

2.5 Désinstallation pour tous les systèmes

Pour désinstaller SG-Lock il suffit de supprimer les fichiers indiqués ci-dessus et le cas échéant – de supprimer les entrées indiquées dans les scripts de registre.

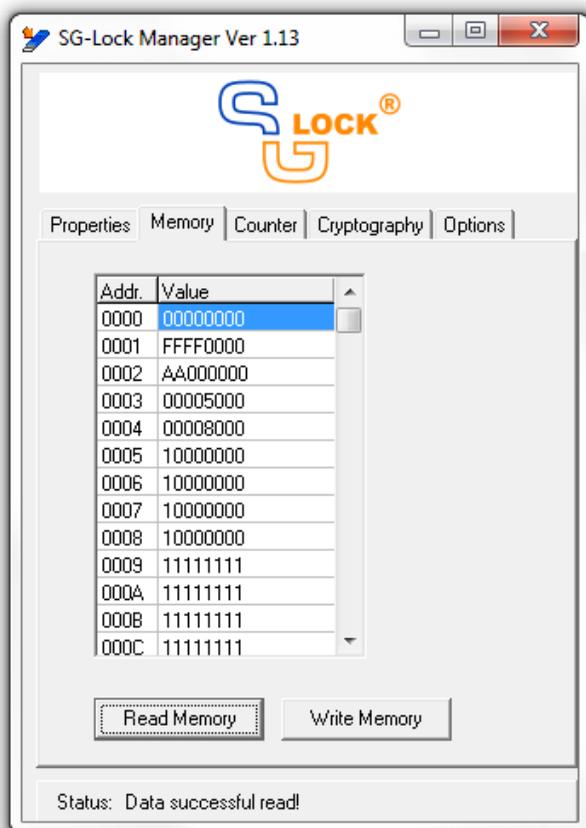
2.6 Modifier SG-Lock avec le manager SG-Lock

Le manager SG-Lock (SGLM) est un programme auxiliaire disponible sur le CDROM SG-Lock et qui sert à modifier et à contrôler tous les modules SG-Lock. Appelez le SGLM en lançant le fichier *SglMgr.Exe* dans le dossier Test. Cliquez sur l'icône *Select Language* dans la fiche *Options* pour adapter la langue. De plus, cette fiche permet d'adapter l'affichage de nombres comme nombres décimaux ou hexadécimaux. Veuillez faire attention à ce mode d'affichage si vous entrez des nombres !



Toutes les fonctions qu'offre le SGLM font partie de l'API SG-Lock et peuvent également être utilisées par toutes les applications protégées. Si vous cliquez, sur la fiche *Properties*, l'icône *Read Properties*, vous trouverez d'importantes informations telles que le type, le numéro de série, l'identifiant produit (ID) et le numéro de version du SG-Lock connecté.

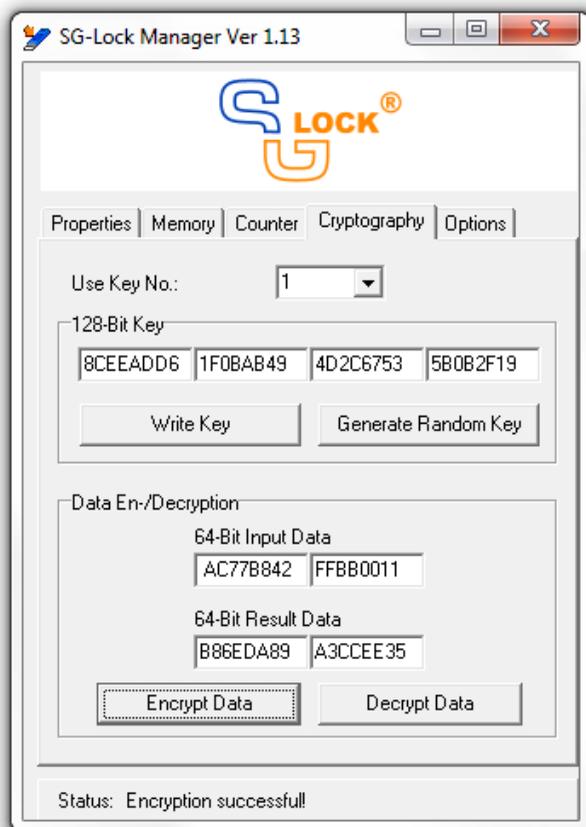
L'icône *Write Product ID* permet d'attribuer à l'identifiant produit des valeurs entre 0 et 65535 (déc.). La fonction de l'identifiant produit sera expliquée dans le chapitre 3.3.



La fiche *Memory* permet de lire la mémoire interne du module (si disponible)

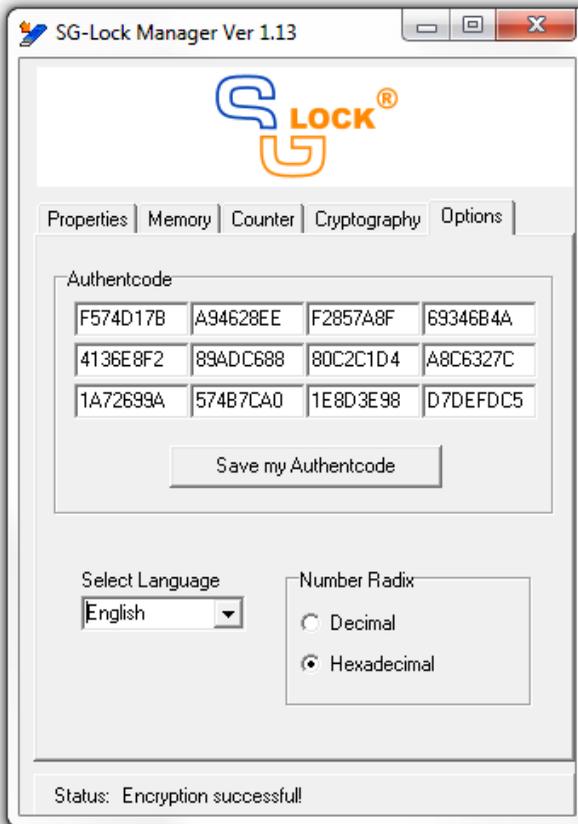
et d'y inscrire des données. Pour changer une ou plusieurs cases de mémoire inscrivez les valeurs souhaitées dans le tableau et mémorisez-les dans la mémoire interne du module en cliquant l'icône *Write Memory*.

La fiche *Counter* offre les mêmes possibilités pour les cases de mémoire du compteur. Celles-ci ne sont pas affichées dans la mémoire normale, elles utilisent une mémoire supplémentaire ce qui évite toute interférence entre la mémoire des données et celle du compteur.



La fiche *Cryptography* offre la possibilité d'effectuer les fonctions cryptographiques de SG-Lock. SG-Lock se sert d'un chiffrement par blocs symétrique

(c'est-à-dire que les clés pour le chiffrement et le déchiffrement sont identiques) 64 bits intégré au module. La longueur de clé est 128 bits. L'algorithme de chiffrement est TEA. Les SG-Lock des séries 3 et 4 disposent de plusieurs mémoires clés. D'abord on choisit, sous *Use Key No.*, la clé qui doit être modifiée ou qui doit servir au chiffrement.



L'icône *Generate Random Key* permet de produire une clé aléatoire 128 bits. Celle-ci peut être traitée par le presse-papier, par ex. à des fins de documentation. (Ceci est recommandé parce que pour des raisons de sécurité les clés ne peuvent pas être lues, mais seulement écrites.) Avec la fonction *Write Key*, la clé

peut ensuite être inscrite dans la mémoire interne du module. Afin de chiffrer des données d'évaluation il faut entrer deux valeurs 32 bits (l'équivalent d'un bloc 64 bits) dans les deux cases *Input Data*. Par l'appel des fonctions *Encrypt* or *Decrypt Data* ces données chiffrées ou déchiffrées avec la clé préselectionnée sont affichées dans les cases *Output Data*. La fiche *Options* permet de fixer la langue souhaitée et le mode d'affichage des nombres. Dans le cas de l'affichage hexadécimal, tous les nombres sauf les numéros de version sont indiqués sur la fiche *Properties*. C'est aussi en mode hexadécimal sans signes diacritiques que les nombres sont entrés.

Attention : Il est nécessaire d'entrer un code d'authentification (AC), si d'autres modules SG-Lock que les démos doivent être utilisés. Si aucun AC n'est entré, seuls les modules démos sont reconnus. Chaque développeur de logiciel employant SG-Lock reçoit une fois lors de la première livraison un AC individuel qui lui garantit l'accès exclusif à ses modules SG-Lock. Tous les modules livrés par la suite seront initialisés avec le même AC. Pour avoir accès à tous les modules retail avec le SGLM il faut entrer et mémoriser une seule fois l'AC.

Attention : L'AC est communiqué en mode hexadécimal – le cas échéant, le mode d'affichage de nombres doit être adapté lors de l'entrée de l'AC.

3 Protéger des logiciels avec SG-Lock

3.1 Généralités

Le fonctionnement de SG-Lock en tant que protection contre la copie se base sur la connexion produite par l'appel de certaines fonctions entre le logiciel facile à copier et qui doit être protégé par conséquent et le matériel SG-Lock dont la copie est pratiquement impossible.

Ce sont les fonctions de l'API (Application Programming Interface) qui sont utilisées dans ce but. Elles sont incluses dans le logiciel livré avec le matériel SG-Lock – dans la bibliothèque SGLW32.DLL.

L'API de SG-Lock offre diverses fonctions qui sont mises en oeuvre selon le genre de la protection souhaitée. Une protection efficace n'a pas besoin de la totalité de ces fonctions.

3.2 Stratégies de protection

La façon la plus fréquente de protéger un logiciel contre l'utilisation illégale consiste dans la simple protection anti-copie qui doit empêcher l'utilisation du logiciel sur un nombre d'ordinateurs excédant celui fixé dans le contrat. Le logiciel est sécurisé par la vérification répétée de l'installation d'un SG-Lock sur l'ordinateur. D'autres stratégies de protection permettent de limiter le nombre de démarrages qu'accepte un certain logiciel. Pour cela, il est nécessaire de contrôler outre la présence d'un SG-Lock un compteur ou une variable dans le SG-Lock afin de bloquer l'exécution du logiciel après un certain nombre de démarrages.

Une autre limitation de l'utilisation consiste à déterminer la date jusqu'à laquelle le logiciel peut être exécuté. Pour cela, il faut inscrire la date dans la mémoire de données et la contrôler à chaque démarrage. Dans le cas du « pay per use », l'utilisateur doit payer pour la prolongation de l'utilisation. Il faut donc enregistrer une nouvelle date, et la procédure recommence.

Une autre possibilité est de faire marcher le logiciel sans limite tout en comptant les emplois de certaines fonctions qui sont payants.

3.3 L'identifiant du produit – à quoi sert-il ?

Les fabricants offrent souvent des paquets logiciel et/ ou suppléments différents. Si plusieurs de ces applications différentes utilisent le dispositif de protection

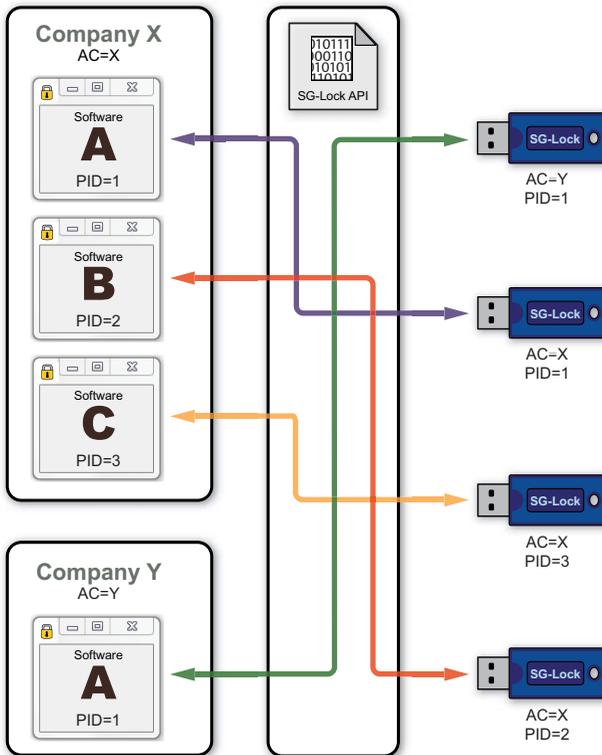


FIGURE 3.1 : La SG-Lock ProductId permet une séparation facile de différents produits d'une compagnie. L'Authentcode sépare des compagnies strictement l'un de l'autre.

anticopie d'un même éditeur, il en résulte une administration lourde dans la programmation parce qu'il faut vérifier à chaque appel que le module de protection anticopie appartient au paquet en cours ou bien qu'il s'adapte au supplément. SG-Lock évite cette charge administrative ainsi que les interférences et les erreurs qui peuvent en résulter en assignant un identifiant produit. L'API de SG-Lock vérifie au moyen de l'identifiant produit si le module de protection anticopie appartenant à l'application est branché.

Un exemple : L'entreprise X propose les trois paquets logiciel A, B et C. Au produit A est assigné l'identifiant produit 1, au produit B l'identifiant 2 et à C l'identifiant 3. L'utilisateur lance le logiciel B, les trois modules SG-Lock pour les trois paquets offerts étant branchés sur l'ordinateur.

S'il n'était pas possible de recourir à l'identifiant produit, il serait nécessaire d'appeler les trois modules l'un après l'autre lors de l'appel d'une fonction de l'API et de vérifier s'il s'agit du module anti-copie appartenant au logiciel. Ce n'est qu'après que le numéro de série peut être lu par exemple. L'utilisation de l'identifiant produit facilite le procédé de vérification et évite des erreurs. Vérifiant la présence du module anti-copie appartenant au logiciel B, l'API de SG-Lock prend l'identifiant produit comme paramètre et reçoit la valeur 2. Résultat : Le module de l'application B est trouvé, les deux autres sont exclus par l'API de SGLock. L'application B fonctionne virtuellement sur un ordinateur auquel il n'est branché qu'un module de protection anti-copie au maximum.

3.4 Le Chiffrement et l'authentification stimulationréponse

Le chiffrement TEA (Tiny Encryption Algorithm) qu'utilise SG-Lock est un excellent moyen de chiffrer des données importantes – mais ce n'est pas tout : Tous les dispositifs de protection logiciel proposés par SG-Lock – même les plus simples – peuvent encore être améliorés par l'application du chiffrement TEA intégré. Le concept de sécurité sur lequel se base notre système consiste à authentifier le module de protection anti-copie trouvé au moyen du chiffrement correct d'un nombre aléatoire et d'une clé secrète seulement connue des deux côtés.

Ce procédé nommé authentification stimulation-réponse (Challenge-Response- Authentification) est très répandu dans la cryptographie. La clé 128 bits sert de mot de passe qui – inscrit dans la mémoire clé – doit être connu

du SG-Lock à authentifier. Cependant, le mot de passe n'est pas transmis lors du processus d'authentification parce qu'il pourrait être intercepté et ne serait plus secret alors. Ce qui est effectivement vérifié c'est la présupposition d'un mot de passe ainsi que l'existence de la clé correcte.

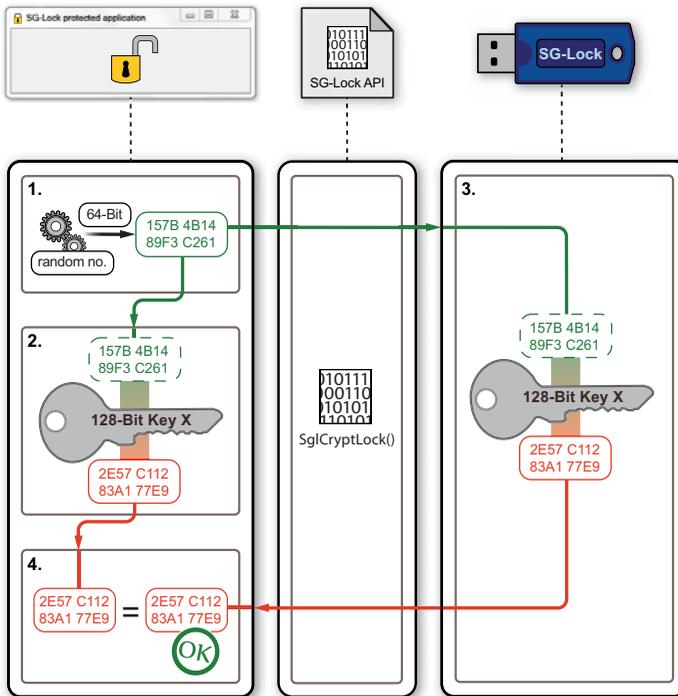


FIGURE 3.2 : La Challenge-Response-Authentication de SG-Lock fournit une connexion sûre de l'application protégée (l'EXE-fichier) par le système d'exploration entier sur le bus l'USB au SG-Lock lui-même.

Le processus se déroule comme suit (pour les exemples de programmation voir paragraphe 6.6.) :

1. Vous générez – par ex. avec le manager SG-Lock – une clé 128 bits aléatoire, vous l’intégrez dans la programmation du module SG-Lock (cette fonction de l’API s’appelle `SglWriteKey`) et vous la déclarez en plus comme constante dans votre application à protéger. C’est ainsi que la clé est connue des deux côtés (du programme protégé et du module SG-Lock). Attention : Cette étape n’intervient pas pour la série 2 puisque les clés de cette série programmées par l’éditeur ne peuvent pas être modifiées. Utilisez la clé programmée par l’éditeur, elle vous sera signalée séparément lors de la première livraison. Les modules de démonstration ont leurs propres clés comme l’explique le chapitre 5.
2. Générez dans le programme à protéger un nombre aléatoire 64 bits (deux nombres aléatoires 32 bits en pratique) avec la fonction qu’offre votre langue de programmation pour la génération de nombres aléatoires. Utilisez – si c’est possible – une autre fonction garantissant qu’après chaque démarrage du programme d’autres nombres sont choisis (« seed »). Sinon, il y aurait toujours les mêmes séquences de nombres aléatoires ce qui affaiblirait la protection.
3. Chiffrez ensuite le nombre aléatoire 64 bits au moyen de la fonction `SglCryptLock` que contient l’API de SG-Lock et mémorisez le résultat. Ce chiffrement s’effectue dans le module anti-copie SG-Lock.
4. Chiffrez maintenant le même nombre aléatoire 64 bits (et non pas le dernier résultat) au moyen de la fonction `SglTeaEncipher` contenue dans le fichier `SglW32-Include` en utilisant la même clé 128 bits générée auparavant. Mémorisez également ce résultat afin de pouvoir le comparer dans l’étape suivante. Ce chiffrement s’effectue dans l’application protégée.
5. Vérifiez par comparaison si le résultat du chiffrement effectué par le module correspond au résultat du chiffrement effectué dans l’application. L’authenticité n’est garantie que dans le cas de l’identité des deux résultats puisqu’ apparemment les deux chiffrements ont été effectués au moyen de la clé 128 bits correcte qui en dehors de l’application n’existe que dans le module SG-Lock recherché.

La comparaison des résultats de chiffrement peut être répartie sur plusieurs parties du programme si vous comparez ou le résultat entier ou seulement des parties (la première séquence 16 bits par ex.) à un certain moment du programme et que vous vérifiez ultérieurement le résultat entier ou les autres parties. C'est ainsi qu'il devient plus difficile de supprimer la comparaison et par là la protection anti-copie dans la langue machine de l'application protégée. Une autre possibilité consiste à chiffrer d'un coup plusieurs blocs 64 bits pour les comparer séparément.

3.5 L'adaptation à différentes langues de programmation

Les fonctions de l'API SG-Lock peuvent être activées directement dans le programme protégé. Pour cela il est nécessaire d'intégrer – suivant la langue de programmation – les fichiers Include enregistrés sur le CD-Rom SG-Lock dans le texte du programme. C#, Visual Basic, Delphi et quelques autres signalent à l'éditeur de liens (linker) que les fonctions SG-Lock se trouvent dans la bibliothèque externe SGLW32.Dll (appelée parfois « third party DLL »).

Sous C/C++, pour la connexion statique usuelle il est nécessaire de signaler au moyen d'une bibliothèque importée à l'éditeur de liens que les fonctions de l'API SG-Lock se trouvent dans une bibliothèque externe, à savoir SGLW32.Dll. Pour ce faire, il faut ajouter la bibliothèque importée au projet dans la liste des bibliothèques que l'éditeur de liens doit traiter. Autrement, l'éditeur de liens va terminer le processus de l'édition de liens, et un message d'erreur va signaler qu'il n'a pas trouvé les fonctions SG-Lock utilisées. Malheureusement, le format de bibliothèques importées varie d'un compilateur à l'autre. Les bibliothèques importées des compilateurs les plus usuels se trouvent sur le CD-Rom SG-Lock.

Dans le cas où il n'y a pas de bibliothèque convenant au compilateur, on se servira de l'outil joint au compilateur pour la générer. Veuillez consulter la documentation du compilateur (mot-clé : génération d'une bibliothèque importée). Une autre possibilité consiste à attribuer un lien dynamique à la bibliothèque SG-Lock en employant les fonctions *LoadLibrary()* et *GetProcAddress()* qu'offre Win32. Pour en savoir plus veuillez consulter la documentation Microsoft Windows SDK.

4 L'API SG-Lock

4.1 Plan des fonctions

Les fonctions de l'API SG-Lock se divisent en quatre groupes : fonctions de base, fonctions élargies, fonctions cryptographiques et fonctions administratives. Effectuant des opérations fondamentales comme la vérification de la connexion d'un SG-Lock, les fonctions de base interviennent pratiquement dans toutes les formes de protection du logiciel. Les fonctions élargies offrent des fonctionnalités répondant à des objectifs opérationnels particuliers, comme par ex. des mémoires ou des compteurs qui permettent d'enregistrer des chaînes de caractères (strings) ou de compter et de limiter le nombre de démarrages d'un programme. Les fonctions cryptographiques servent à chiffrer et à signer n'importe quelles données telles que des textes, des images, des courriels, des films etc. C'est des stratégies de protection et de commercialisation que dépend l'intégration de ces dernières fonctions dans le code du programme.

Le groupe des fonctions administratives est un supplément destiné à préparer les modules SG-Lock pour la livraison avec le logiciel protégé. Normalement, il n'est pas intégré dans le code de l'application protégée. En effet, il permet de créer rapidement et de manière individuelle des programmes d'initialisation simples.

Nom de fonction	Description
Fonctions de base	
SglAuthent	Authentification de la bibliothèque SG-Lock.
SglSearchLock	Cherche un périphérique SG-Lock.
SglReadSerialNumber	Lit le numéro de série d'un périphérique SG-Lock.
Fonctions élargies	
SglReadData	Lit des données dans la mémoire d'un périphérique SG-Lock.
SglWriteData	Inscrit des données dans la mémoire d'un périphérique SG-Lock.
SglReadCounter	Lit une valeur de comptage dans un périphérique SG-Lock.
SglWriteCounter	Inscrit une valeur de comptage dans un périphérique SG-Lock.
Fonctions cryptographiques	
SglCryptLock	Chiffre et déchiffre un ou plusieurs blocs de données au moyen d'un périphérique SG-Lock et d'une clé 128 bits intégrée au périphérique.
SglSignData	Signe des données par combinaison de l'ordinateur et du périphérique SG-Lock.
Fonctions administratives	
SglReadProductId	Lit l'identifiant produit dans un périphérique SG-Lock.
SglWriteProductId	Inscrit l'identifiant produit dans un périphérique SG-Lock.
SglWriteKey	Inscrit une clé 128 bits dans un périphérique SG-Lock.
SglReadConfig	Lit des données de configuration dans les environnements du SG-Lock ou dans un périphérique SG-Lock (par ex. le type du périphérique SG-Lock)

4.2 Fonctions de base

4.2.1 Fonction : SglAuthent

Description

Authentification de la bibliothèque SG-Lock envers l'application protégée et vice versa.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglAuthent(
    ULONG *AuthentCode );
```

Paramètre

AuthentCode	Suite de 48 octets (12 DWORDs) assignée de manière individuelle à chaque utilisateur de SG-Lock.
-------------	--

Valeurs de retour

SGL_SUCCESS	Exécution correcte
SGL_AUTHENTICATION_FAILED	L'authentification avec <i>SglAuthent</i> a échoué

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Cette fonction de l'API SG-Lock doit être la première à être activée une fois avec succès, parce que toutes les autres fonctions de l'API SG-Lock ne sont disponibles qu'après. Dans le cas de liens dynamiques, ceci doit être effectué pour l'édition de chaque lien (appel *LoadLibrary()*). Les kits de démonstration ont leur propre code d'authentification indiqué dans les exemples de programmation.

Attention : Cette fonction ne peut pas vérifier si un module anti-copie SG-Lock est branché.

4.2.2 Fonction : SglSearchLock

Description

Cherche un SG-Lock.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglSearchLock(  
    ULONG ProductId );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
-----------	---

Valeurs de retour

SGL_SUCCESS	SG-Lock trouvé.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Fonction standard qui vérifie si un module SG-Lock est branché.

4.2.3 Fonction : SglReadSerialNumber

Description

Lit le numéro de série assigné de manière individuelle à chaque SG-Lock.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglReadSerialNumber (
    ULONG ProductId ,
    ULONG *SerialNumber );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
SerialNumber	Aiguille pointée sur la variable par laquelle le numéro de série du SG-Lock est rendu.

Valeurs de retour

SGL_SUCCESS	Numéro de série SG-Lock lu avec succès.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Indépendamment du type d'interface USB ou LPT, chaque module SG-Lock possède un numéro de série individuel (ceci ne vaut pas pour les modules de démonstration).

Ce numéro de série individuel ne sert pas seulement à l'identification d'un module SG-Lock, mais il sert également de valeur principale (Mastervalue) dérivant de façon sécurisée des numéros, des clés, des codes individuels au moyen de fonctions de conversion appropriées.

4.3 Extended Fonction

4.3.1 Fonction : SglReadData

Description

Lit des valeurs de donnée 32 bits dans la mémoire SG-Lock.

Modèles

U2 : X U3 : ✓ U4 : ✓

Declaration

```
ULONG SglReadData (
    ULONG ProductId ,
    ULONG Address ,
    ULONG Count ,
    ULONG *Data );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
Address	Adresse de départ du bloc composé de valeurs de donnée. 0 jusqu'à 63 – SG-Lock U3 0 jusqu'à 255 – SG-Lock U4
Count	Nombre des valeurs de donnée.
Data	Aiguille pointée sur la zone de données dans laquelle les valeurs de donnée doivent être copiées (le développeur doit vérifier la taille suffisante de la zone de données).

Valeurs de retour

SGL_SUCCESS	Valeurs de donnée lues avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

La mémoire de données intégrée au module peut être utilisée pour stocker de façon sécurisée n'importe quelles données comme des codes, des numéros, des clés, des mots de passe, des licences, etc.

4.3.2 Fonction : SglWriteData

Description

Inscrit des valeurs de donnée 32 bits dans la mémoire SG-Lock.

Modèles

U2 : X U3 : ✓ U4 : ✓

Declaration

```
ULONG SglWriteData (
    ULONG ProductId ,
    ULONG Address ,
    ULONG Count ,
    ULONG *Data );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
Address	Adresse de départ du bloc composé de valeurs de donnée. 0 jusqu'à 63 – SG-Lock U3 0 jusqu'à 255 – SG-Lock U4
Count	Nombre des valeurs de donnée.
Data	Aiguille pointée sur la zone de données dans laquelle les valeurs de donnée doivent être prises.

Valeurs de retour

SGL_SUCCESS	Valeurs de donnée inscrites avec succès dans la mémoire SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Voir *SglReadData*.

4.3.3 Fonction : SglReadCounter

Description

Lit une valeur de comptage 32 bits dans la mémoire SG-Lock.

Modèles

U2 : X U3 : ✓ U4 : ✓

Declaration

```
ULONG SglReadCounter (
    ULONG ProductId ,
    ULONG CntNum ,
    ULONG *Data );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
CntNum	Nombre des valeurs de donnée. 0 jusqu'à 15 – SG-Lock U3 0 jusqu'à 63 – SG-Lock U4
Data	Data aiguille pointée sur la variable recevant la valeur de comptage.

Valeurs de retour

SGL_SUCCESS	Valeur de comptage lue avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Les compteurs (Counter) sont des zones de données 32 bits dans la mémoire SGLock qui ne servent pas seulement au comptage mais aussi à toutes les applications acceptant des variables 32 bits. La mémoire du module peut encore être élargie de cette manière.

4.3.4 Fonction : SglWriteCounter

Description

Inscrit une valeur de comptage 32 bits dans la mémoire SG-Lock.

Modèles

U2 : X U3 : ✓ U4 : ✓

Declaration

```
ULONG SglWriteCounter (
    ULONG ProductId ,
    ULONG CntNum ,
    ULONG Data );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
CntNum	Nombre des valeurs de donnée. 0 jusqu'à 15 – SG-Lock U3 0 jusqu'à 63 – SG-Lock U4
Data	Valeur de comptage à écrire.

Valeurs de retour

SGL_SUCCESS	Valeur de comptage inscrite avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Voir *SglReadCounter*.

4.4 Fonctions de chiffrement et de signature

4.4.1 Fonction : SglCryptLock

Description

Chiffre ou déchiffre un ou plusieurs blocs de données 64 bits avec une clé 128 bits. L'algorithme est TEA.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglCryptLock (
    ULONG ProductId ,
    ULONG KeyNum ,
    ULONG CryptMode ,
    ULONG BlockCnt ,
    ULONG *Data );
```

Paramètre

ProductId	Identifiant produit, affiche l' identifiant produit du SG-Lock cherché.
KeyNum	Numéro de la clé à utiliser : 0 jusqu'à 1 – SG-Lock U3 0 jusqu'à 15 – SG-Lock U4
CryptMode	Mode de travail : 0 – Chiffrer 1 – Déchiffrer
BlockCnt	Nombre des blocs de données 64 bits à traiter
Data	Aiguille pointée sur la zone de données où se trouvent les blocs de données à traiter (le développeur doit vérifier la taille suffisante par rapport au paramètre BlockCnt).

Valeurs de retour

SGL_SUCCESS	Chiffrement accompli.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

La fonction modifie les valeurs d'entrée transmises par le paramètre Data. Si celles-ci doivent être traitées ultérieurement, elles doivent être sauvegardées avant l'appel de la fonction.

4.4.2 Fonction : SglSignData

Description

Description : signe ou vérifie la signature d'un champ de données. Le traitement est effectué par l'application (CPU de l'ordinateur) ainsi que par le SG-Lock connecté. Cette fonction combine les avantages des deux fonctions précédentes.

Condition importante : les deux clés (celle de l'application et celle intégrée au module SG-Lock) doivent être différentes ! La signature a une longueur de 64 bits.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglSignData(  
    ULONG ProductId ,  
    ULONG *AppSignKey ,  
    ULONG LockSignKeyNum ,  
    ULONG Mode ,  
    ULONG LockSignInterval ,  
    ULONG DataLen ,  
    ULONG *Data ,  
    ULONG *Signature );
```

Paramètre

ProductId	Identifiant produit, affiche l'identifiant produit du SG-Lock à utiliser.
AppSignKey	Clé 128 bits de l'application à utiliser lors de la génération ou la vérification.
LockSignKeyNum	Le numéro de la clé 128 bits à utiliser lors de la génération ou la vérification. 0 – SG-Lock U2 0 jusqu'à 1 – SG-Lock U3 0 jusqu'à 15 – SG-Lock U4
Mode	Mode de travail : 0 – Générer la signature 1 – Vérifier la signature
LockSignInterval	Affiche la manière dont la capacité de calcul destinée à la signature se partage entre l'application et le SG-Lock. Puissance de 2, la valeur détermine le bloc de données qui est traité par le SG-Lock. Exemple : valeur= 8, 2 puissance 8= 256, c'est-à-dire qu'à chaque 256 ^e valeur de données 32 bits, SG-Lock traite les données, toutes les autres sont traitées par l'application (CPU de l'ordinateur).
DataLen	Nombre de valeurs 32 bits du champ de données
Data	Aiguille pointée sur la zone de données où se trouvent les blocs de données à traiter.
Signature	Aiguille pointée sur la zone de données dans laquelle soit la signature générée est rendue soit la signature est transmise pour être vérifiée. Aiguille pointée sur un champ de données avec deux valeurs 32 bits (le développeur doit vérifier la taille suffisante par rapport au paramètre DataLen)

Valeurs de retour

SGL_SUCCESS	Signature générée avec succès ou signature valable.
SGL_SIGNATURE_INVALID	Signature non valable.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Comme et la clé de l'application protégée et celle du SG-Lock utilisé sont absolument nécessaires à la génération de la signature, la grande capacité de calcul de l'application (CPU de l'ordinateur) est combinée avec la haute sécurité du SGLock. Le SG-Lock traite toujours le premier bloc de données, et tous les blocs suivants en dépendent à cause de l'enchaînement mis en place lors de la génération de la signature.

Attention : Les deux clés 128 bits utilisées ici (celle de l'application et celle intégrée au module SG-Lock) doivent être différentes et aussi dissemblables que possible afin que les avantages de la fonction soient exploitées au plus haut degré.

4.5 Administrative Fonctions

4.5.1 Fonction : SglReadProductId

Description

Lit l'identifiant produit à 16 bits dans le SG-Lock.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglReadProductId (
    ULONG* ProductId);
```

Paramètre

ProductId	Identifiant produit, aiguille pointée sur une variable 32 bits recevant l'identifiant produit.
-----------	--

Valeurs de retour

SGL_SUCCESS	Identifiant produit lu avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

L'identifiant produit sert à faciliter l'administration de plusieurs applications protégées ou de plusieurs suppléments d'une application.

Attention : Seuls les 16 bits inférieurs ou les identifiants produit de 0 jusqu'à 65535 peuvent être utilisés. Une description plus détaillée de l'emploi de l'identifiant produit se trouve dans le chapitre 3.3.

Les compteurs (Counter) sont des zones de données 32 bits dans la mémoire SG-Lock qui ne servent pas seulement au comptage mais aussi à toutes les applications acceptant des variables 32 bits.

La mémoire du module peut encore être élargie de cette manière.

4.5.2 Fonction : SglWriteProductId

Description

Inscrit un nouvel identifiant produit 16 bits dans le SG-Lock.

Modèles

U2: ✓ U3: ✓ U4: ✓

Declaration

```
ULONG SglWriteProductId (
    ULONG OldProductId ,
    ULONG NewProductId );
```

Paramètre

OldProductId	OldProductId, affiche l'identifiant produit actuel du SG-Lock.
NewProductId	Affiche le nouvel identifiant produit du SG-Lock.

Valeurs de retour

SGL_SUCCESS	Identifiant produit inscrit avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Voir *SglReadProductId*.

4.5.3 Fonction : SglWriteKey

Description

Inscrit une clé 128 bits dans la mémoire clé du SG-Lock.

Modèles

U2 : X U3 : ✓ U4 : ✓

Declaration

```
ULONG SglWriteKey (
    ULONG ProductId ,
    ULONG KeyNum ,
    ULONG *Key );
```

Paramètre

ProductId	Identifiant produit, affiche l'identifiant produit du SGLock à utiliser.
KeyNum	Numéro de la clé à écrire. 0 jusqu'à 1 – SG-Lock U3 0 jusqu'à 15 – SG-Lock U4
Key	Clé 128 bits à écrire. Aiguille pointée sur un champ de données composé de quatre valeurs 32 bits constituant.

Valeurs de retour

SGL_SUCCESS	Clé inscrite avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

4.5.4 Fonction : SglReadConfig

Description

Lit des configurations et des informations se référant au SG-Lock.

Modèles

U2 : ✓ U3 : ✓ U4 : ✓

Declaration

```
ULONG SglReadConfig(
    ULONG ProductId ,
    ULONG Category ,
    ULONG *Data );
```

Paramètre

ProductId	Identifiant produit, affiche l'identifiant produit du SG-Lock à utiliser
Category	0 : information sur le module SG-Lock.
Data	Aiguille pointée sur un champ de données composé de 8 valeurs de nombre entier ayant une largeur de 32 bits. Voilà les significations des valeurs : Index 0 : type Index 1 : interface Index 2 : version logiciel Index 3 : version matériel Index 4 : numéro de série Index 5 : volume de mémoire en DWORDs Index 6 : nombre de compteurs Index 7 : nombre de clés 128 bits

Valeurs de retour

SGL_SUCCESS	Configuration lu avec succès dans le SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock n'a pas été trouvé.

La liste complète des valeurs de retour se trouve dans le chapitre 4.6.

Commentaire

Les fichiers Include et Header de l'API SG-Lock offrent des informations supplémentaires sur les différentes valeurs.

4.6 Valeurs de retour

Chaque fonction de l'API SG-Lock rend une valeur au moyen de laquelle l'exécution correcte de la fonction peut être vérifiée. S'il y a une erreur au cours de l'exécution, une valeur autre que 0 est rendue. La valeur rendue précise l'erreur.

Valeur	rendue	Description
SGL_SUCCESS	0	Exécution correcte.
SGL_DGL_NOT_FOUND	1	SG-Lock n'a pas été trouvé.
SGL_AUTHENTICATION- _REQUIRED	5	L'authentification avec <i>SglAuthent</i> n'a pas été effectuée ou elle n'a pas été effectuée correctement.
SGL_AUTHENTICATION- _FAILED	6	L'authentification avec <i>SglAuthent</i> a échoué.
SGL_FUNCTION_NOT- _SUPPORTED	7	La fonction activée n'est pas soutenue par le module SG-Lock reconnu (par ex. <i>SglReadData</i> dans le cas du module U2)
SGL_PARAMETERS- _INVALID	8	Un paramètre de la fonction activée n'est pas dans la zone des valeurs acceptées (par ex. Adresse 5000 pour la fonction <i>SglReadData</i>).
SGL_SIGNATURE_INVALID	9	Signature non valable.
SGL_USB_BUSY	10	SG-Lock est occupé

TABLE 4.1 : valeurs rendues de l'API SG-Lock et leurs significations.

5 Chiffrement, signature et administration de clés

SG-Lock dispose d'une fonction de chiffrement symétrique (c'est-à-dire que la clé pour le chiffrement et celle pour le déchiffrement sont identiques) par blocs intégrée au module. La taille des blocs de données est de 64 bits soit 2 mots doubles. La clé a une largeur de 128 bits soit 4 mots doubles.

La fonction de chiffrement peut être utilisée pour chiffrer ou déchiffrer et pour signer n'importe quelles données comme par ex. des informations de configuration, des données confidentielles, etc. Le chiffrement peut être effectué directement dans le matériel SG-Lock ce qui garantit une haute sécurité parce que la clé n'apparaît pas en dehors du SG-Lock et ne peut pas être interceptée par conséquent. La haute sécurité que fournit ce fonctionnement s'accompagne d'une capacité de chiffrement réduite, à savoir 100 blocs par seconde, c'est-à-dire environ 0,8 kb/sec, restriction qui tient à la capacité de calcul du matériel SG-Lock. En pratique, cette fonction est limitée à de petites quantités de données si le chiffrement s'effectue entièrement dans le SG-Lock. Le chiffrement utilisant la CPU de l'ordinateur offre moins de sécurité parce que la clé est mémorisée quelque part dans le système et qu'elle pourrait donc être trouvée avec certains outils. L'avantage de cette variante est une haute capacité de chiffrement excédant 10 MB/sec.

Il est également possible de combiner le chiffrement interne du SG-Lock avec le chiffrement interne de l'ordinateur pour unir dans un procédé leurs avantages – la haute capacité de chiffrement de la CPU de l'ordinateur et la haute sécurité du chiffrement interne du SG-Lock. Cette combinaison atteint pratiquement la même capacité de chiffrement que celle du chiffrement interne de l'ordinateur.

Pour cette solution du chiffrement, tous les blocs de données 64 bits sont chiffrés séparément et liés l'un à l'autre. Le premier bloc et quelques autres par intervalle sont chiffrés par le matériel SG-Lock, tous les autres sont chiffrés par la CPU de l'ordinateur. La sécurité de ce procédé dépend principalement du fait que les deux clés 128 bits (celle du SG-Lock et celle de l'ordinateur) sont *différentes*. Car en principe, la clé de l'ordinateur peut toujours être interceptée,

mais grâce au chiffrement effectué à l'intérieur du SG-Lock et à l'association des blocs de données, les données sont déjà chiffrées avec une *autre* clé 128 bits inconnue.

Le fabricant initialise toutes les mémoires clé (1, 2 ou 16 unités selon le type du module) avec leurs clés avant la livraison. Chaque utilisateur de SG-Lock reçoit ses propres clés secrètes qu'il peut modifier en cas de nécessité avec des clés générées par lui-même (les clés des types U2/L2 ne peuvent pas être modifiées). Tous les modules SG-Lock d'un utilisateur contiennent un lot identique de clés. Lors de la première commande, ce lot de clés est communiqué à l'utilisateur. Tous les modules de démonstration (USB et LPT) ont leur propre lot de clés individuel indiqué ci-dessous :

N°	Type	Clé 128 bits (hexadécimal)			
0	2,3,4	D94B6C2B	17E88CEF	DADBCF1D	202161A2
1	3,4	2181588C	3798A2BB	36CAB86B	051040C1
2	4	BBDBF022	D0D85396	9B6EFB5F	41354633
3	4	97CCAFDC	1EB606E7	5CB83119	9F7F457C
4	4	F8BA5A4D	1C1BCBD0	61140A39	49507A3F
5	4	326FD7E8	E6C39F3A	CBA04A4B	37804850
6	4	554E5BA7	81665744	8F747F62	E0EE72F9
7	4	BAD58985	238BF49B	C97B1173	D3A28313
8	4	98940499	D20EDC71	68388EB6	B5DF3D1C
9	4	0FC6EC5F	EBD20065	093984EF	F52F415F
10	4	8DC071AA	668477BE	095C0CBE	3545E855
11	4	CBC15944	155BF5E3	88D9C8D3	E7142A18
12	4	F0D76719	43A48195	7AA26332	D3B2E83C
13	4	8A467F11	789CD8E2	030FE272	A4750E6B
14	4	18FD8C08	B29157D7	F160F6A2	9E2FA426
15	4	90EC452F	04C30099	4B5102A9	4D942D78

TABLE 5.1 : les clés 128 bits de modules de démonstration programmées par le fabricant

6 Exemples de programmation

6.1 Fonction SglAuthent

6.1.1 C/C++

```
#include "SGLW32.h"

unsigned int ReturnCode;

// This is the DEMO authentication code,
// every regular SG-Lock user gets its
// own unique authentication code.
unsigned int MyAuthentCode [12] = {
    0xF574D17B, 0xA94628EE,
    0xF2857A8F, 0x69346B4A,
    0x4136E8F2, 0x89ADC688,
    0x80C2C1D4, 0xA8C6327C,
    0x1A72699A, 0x574B7CA0,
    0x1E8D3E98, 0xD7DEFDC5 };

// do authentication of SGLW32.Dll
ReturnCode = SglAuthent( MyAuthentCode );
if( ReturnCode != SGL_SUCCESS ) {
    // authentication failed!!
    printf( "SglAuthent: Error! (code: %0x%X)\n",
        ReturnCode );
}

// authentication succeeded... do the next regular thing...
```

6.1.2 Delphi

```
interface
uses
{$INCLUDE 'SGLW32IF.PAS'}
implementation
{$INCLUDE 'SGLW32IP.PAS'}
```

```

{ This is the DEMO authentication code, every regular SG-Lock
  user gets its own unique authentication code.}
MyAuthentCode: Array [0..11] of LongWord= (
  $F574D17B, $A94628EE, $F2857A8F, $69346B4A,
  $4136E8F2, $89ADC688, $80C2C1D4, $A8C6327C,
  $1A72699A, $574B7CA0, $1E8D3E98, $D7DEFDC5 );

procedure TForm1.Button1Click(Sender: TObject);
  var ReturnCode: LongWord;

  { do authentication of SGLW32.Dll }
  ReturnCode:= SglAuthent( MyAuthentCode );
  if( ReturnCode <> SGL_SUCCESS ) then
    begin
      { authentication failed !! }
      Memo1.Text:= 'SglAuthent: _Error!_' + char($0D) + char($0A);
    end;

  { authentication succeeded .. do the next regular thing ... }

end;

```

6.1.3 Visual Basic

```

' The file SGLW32.BAS has to be included in the project
' to ensure that all SG-Lock functions and constants
' are declared !!!

' This is the DEMO authentication code, every regular
' SG-Lock user gets its own unique authentication code.
Dim MyAuthentCode(0 To 11) As Long

  MyAuthentCode(0) = &HF574D17B
  MyAuthentCode(1) = &HA94628EE
  MyAuthentCode(2) = &HF2857A8F
  MyAuthentCode(3) = &H69346B4A
  MyAuthentCode(4) = &H4136E8F2
  MyAuthentCode(5) = &H89ADC688
  MyAuthentCode(6) = &H80C2C1D4
  MyAuthentCode(7) = &HA8C6327C
  MyAuthentCode(8) = &H1A72699A
  MyAuthentCode(9) = &H574B7CA0
  MyAuthentCode(10) = &H1E8D3E98
  MyAuthentCode(11) = &HD7DEFDC5

Private Sub ButtonSearchSGLock_Click()

```

```
Dim Rc As Long      ' ReturnCode

' do authentication of SGLW32.Dll
Rc = SglAuthent( AuthentCode() )

If Rc = SGL_SUCCESS Then
    Text1.Caption = "SglAuthent_ succeeded_!"
Else
    Text1.Caption = "SglAuthent_ failed_!"
    Exit Sub
End If

' SG-Lock found .. do the next regular thing ...

End Sub
```

6.2 Fonction SglSearchLock

6.2.1 C/C++

```
#include "SGLW32.h"
// In the case a SG-Lock user protects more than 1
// application/product, he should give each of it a unique
// product ID. Then its very easy to distinguish the SG-Locks
// for each product
#define MY_PRODUCT_ABC_ID 1
#define MY_PRODUCT_XYZ_ID 2

unsigned int ReturnCode ;

// Search SG-Lock with product ABC
ReturnCode = SglSearchLock( MY_PRODUCT_ABC_ID );
if( ReturnCode != SGL_SUCCESS ) {
    // no SG-Lock found!!
    printf( "SglSearchLock: Error!(code: %0x%X)\n",
           ReturnCode );
}

// SG-Lock found! ...do the next regular thing ...
```

6.2.2 Delphi

```
interface
uses
{$INCLUDE 'SGLW32IF.PAS'}
implementation
{$INCLUDE 'SGLW32IP.PAS'}

{In the case a SG-Lock user protects more than 1 application/
 product, he should give each of it a unique product ID. Then
 its very easy to distinguish the SG-Locks for each product}
const MY_PRODUCT_ABC_ID = 1;
        MY_PRODUCT_XYZ_ID = 2;

procedure TForm1.Button1Click (Sender: TObject);
var ReturnCode : LongWord;

    { Search SG-Lock for product ABC }
    ReturnCode := SglSearchLock ( MY_PRODUCT_ABC_ID );
    if( ReturnCode <> SGL_SUCCESS ) then
    begin
        { no SG-Lock found!! }
    end
```

```

    Memol.Text:= 'SglSearchLock:␣Error!␣'+char($0D)+char($0A);
end;

    { SG-Lock found .. do the next regular thing ... }

end;

```

6.2.3 Visual Basic

```

' The file SGLW32.BAS has to be included in the project
' to ensure that all SG-Lock functions and constants
' are declared !!!

' In the case a SG-Lock user protects more than 1
' application/product, he should give each of it a
' unique product ID. Then its very easy to distinguish
' the SG-Locks for each product.
Public Const MY_PRODUCT_ABC_ID As Long = 1
Public Const MY_PRODUCT_XYZ_ID As Long = 2

Private Sub ButtonSearchSGLock_Click()

Dim Rc As Long    ' ReturnCode

    ' Search SG-Lock for product ABC
    Rc = SglSearchLock( MY_PRODUCT_ABC_ID )

    Select Case Rc
        Case SGL_SUCCESS
            Text1.Caption = "SG-Lock␣found␣!"
        Case SGL_DGL_NOT_FOUND
            Text1.Caption = "SG-Lock␣not␣found␣!"
        Exit Sub
        Case Else
            Text1.Caption = "Error␣" & Rc & "␣occured␣!"
        Exit Sub
    End Select

    ' SG-Lock found .. do the next regular thing ...

End Sub

```

6.3 Fonction SglReadSerialNumber

6.3.1 C/C++

```
#include "SGLW32.h"
#define PROD_ABC_ID 1

unsigned int ReturnCode ;
unsigned int SerialNumber

// Read serial number of SG-Lock with product ABC
ReturnCode = SglReadSerialNumber( PROD_ABC_ID, &SerialNumber );
if( ReturnCode != SGL_SUCCESS ) {
    // no SG-Lock found!!
    printf("SglReadSerialNumber :_Error!_(code :_%d)\n",
        ReturnCode );
}

// SG-Lock serial number read ! ...do the next regular thing...
```

6.3.2 Delphi

```
interface
uses
{$INCLUDE 'SGLW32IF.PAS'}
implementation
{$INCLUDE 'SGLW32IP.PAS'}

procedure TForm1.Button1Click(Sender: TObject);
const PROD_ABC_ID = 1;
var ReturnCode : LongWord;
    SerialNumber : LongWord;

{ Read serial number of SG-Lock with product ABC }
ReturnCode:= SglReadSerialNumber( PROD_ABC_ID,
                                Addr(SerialNumber) );
if( ReturnCode <> SGL_SUCCESS ) then
begin
    { no SG-Lock found!! }
    Mem1.Text:= 'SglReadSerialNumber :_Error!_' +
                char($0D) + char($0A);
end;

{ SG-Lock serial number read! ..do the next regular thing.. }

end;
```

6.3.3 Visual Basic

```

' The file SGLW32.BAS has to be included in the project
' to ensure that all SG-Lock functions and constants
' are declared !!!

' In the case a SG-Lock user protects more than 1
' application/product, he should give each of it a
' unique product ID. Then its very easy to distinguish
' the SG-Locks for each product.
Public Const PROD_ABC_ID As Long = 1

Private Sub ButtonSearchSGLock_Click ()

    Dim Rc As Long    ' ReturnCode
    Dim SerialNumber As Long

    ' Read serial number of SG-Lock for product ABC
    Rc = SglReadSerialNumber( PROD_ABC_ID, SerialNumber )

    Select Case Rc
        Case SGL_SUCCESS
            Text1.Caption = SerialNumber
        Case SGL_DGL_NOT_FOUND
            Text1.Caption = "SG-Lock not found!"
            Exit Sub
        Case Else
            Text1.Caption = "Error_" & Rc & "_occured_"
            Exit Sub
    End Select

    ' SG-Lock serial number read ..do the next regular thing...

End Sub

```

6.4 Fonction SglReadData

6.4.1 C/C++

```
#include "SGLW32.h"
#define PROD_ABC_ID 1
// address where date is stored in SG-Lock:
#define RUN_DATE_ADR 10
// date stored as year/month/day (3 DWords):
#define RUN_DATE_CNT 3
unsigned int RC;
unsigned int RunDate [3];
// date storage for compare

// Read date to run of SG-Lock with product ABC
RC = SglReadData (PROD_ABC_ID,
                 RUN_DATE_ADR,
                 RUN_DATE_CNT,
                 RunDate);
if ( RC != SGL_SUCCESS ) {
    // no SG-Lock found!!
    printf ("SglReadData :_Error!_(code:_%d)\n", ReturnCode);
}

// read date from system , compare with RunDate
// and decide what to do
```

6.4.2 Delphi

```
interface
uses
{$INCLUDE 'SGLW32IF.PAS'}
implementation
{$INCLUDE 'SGLW32IP.PAS'}

procedure TForm1.Button1Click (Sender: TObject);
const PROD_ABC_ID = 1;
        RUN_DATE_ADR= 10; { address where date is stored
                           in SG-Lock }
        RUN_DATE_CNT= 3; { date stored as year/month/day
                           (3 DWords) }
var RC      : LongWord;
        RunDate: Array [0..2] of LongWord; { date storage for
                                                compare }

    { Read date to run of SG-Lock with product ABC }
```

```

RC:= SglReadData ( PROD_ABC_ID,
                  RUN_DATE_ADR,
                  RUN_DATE_CNT,
                  Addr(RunDate) );
if ( RC <> SGL_SUCCESS ) then
begin
  { no SG-Lock found!! }
  Memol.Text:= 'SglReadData:␣Error!␣' +
               char($0D) + char($0A);
end;

  {read date from system, compare with RunDate
  and decide what to do}

end;

```

6.4.3 Visual Basic

```

' The file SGLW32.BAS has to be included in the project
' to ensure that all SG-Lock functions and constants
' are declared !!!

' In the case a SG-Lock user protects more than 1
' application/product, he should give each of it a
' unique product ID. Then its very easy to distinguish
' the SG-Locks for each product.
Public Const PROD_ABC_ID As Long = 1
' adresse where date is stored in SG-Lock
Public Const RUN_DATE_ADR As Long = 10
' date stored as year/month/day (3 DWords)
Public Const RUN_DATE_CNT As Long = 3

Private Sub ButtonSearchSGLock_Click ()

  Dim Rc As Long           ' ReturnCode
  Dim RunDate (0 to 2) As Long ' date storage for compare

  ' Read date to run of SG-Lock with product ABC
  Rc = SglReadData ( PROD_ABC_ID, RUN_DATE_ADR,
                   RUN_DATE_CNT, RunDate () )

  Select Case Rc
  Case SGL_SUCCESS
    Text1.Caption = RunDate(0)&"/"&RunDate(1)&"/"&RunDate(2)
  Case SGL_DGL_NOT_FOUND
    Text1.Caption = "SG-Lock␣not␣found␣!"
  Exit Sub

```

```
Case Else
```

```
    Text1.Caption = "Error_" & Rc & "_occured!"
```

```
Exit Sub
```

```
End Select
```

```
'read date from system, compare with RunDate
```

```
'and decide what to do
```

```
End Sub
```

6.5 Fonction SglWriteData

6.5.1 C/C++

```
#include "SGLW32.h"
#define PROD_ABC_ID 1
// adresse where date is stored in SG-Lock:
#define RUN_DATE_ADR 10
// date stored as year/month/day (3 DWords):
#define RUN_DATE_CNT 3
unsigned int RC;
unsigned int RunDate [3]; // date storage
RunDate [0] = 2005; // new run date
RunDate [1] = 12;
RunDate [2] = 24;

// Write new date to run to SG-Lock with product ABC
RC = SglWriteData (PROD_ABC_ID,
                  RUN_DATE_ADR,
                  RUN_DATE_CNT,
                  RunDate );
if ( RC != SGL_SUCCESS ) {
    // no SG-Lock found!!
    printf ("SglWriteData: Error! (code: %d)\n", RC);
}

// new date successfully written , lets do the next thing ...
```

6.5.2 Delphi

```
interface
uses
{$INCLUDE 'SGLW32IF.PAS'}
implementation
{$INCLUDE 'SGLW32IP.PAS'}

procedure TForm1.Button1Click (Sender: TObject);
const PROD_ABC_ID = 1;
      RUN_DATE_ADR= 10; { adresse where date is stored
                        in SG-Lock }
      RUN_DATE_CNT= 3; { date stored as year/month/day
                        (3 DWords) }
var RC      : LongWord;
      RunDate: Array [0..2] of LongWord; { date storage }

      RunDate [0]:= 2005; // new run date
```

```

RunDate [1]:= 12;
RunDate [2]:= 24;

{ Write new date to run to SG-Lock with product ABC }
RC:= SglWriteData( PROD_ABC_ID, RUN_DATE_ADR,
                  RUN_DATE_CNT, Addr(RunDate) );
if ( RC <> SGL_SUCCESS ) then
begin
  { no SG-Lock found!! }
  Mem1.Text:= 'SglWriteData: Error!' +
              char($0D) + char($0A);
end;

{ new date successfully written, lets do the next thing... }

end;

```

6.5.3 Visual Basic

```

' The file SGLW32.BAS has to be included in the project
' to ensure that all SG-Lock functions and constants
' are declared !!!

' In the case a SG-Lock user protects more than 1
' application/product, he should give each of it a
' unique product ID. Then its very easy to distinguish
' the SG-Locks for each product.
Public Const PROD_ABC_ID As Long = 1
' adresse where date is stored in SG-Lock
Public Const RUN_DATE_ADR As Long = 10
' date stored as year/month/day (3 DWords)
Public Const RUN_DATE_CNT As Long = 3

Private Sub ButtonSearchSGLock_Click ()

  Dim Rc As Long           ' ReturnCode
  Dim RunDate (0 to 2) As Long ' date storage

  RunDate (0) = 2005      ' new run date
  RunDate (1) = 12
  RunDate (2) = 24

  ' Write new date to run to SG-Lock with product ABC
  Rc = SglWriteData( PROD_ABC_ID, RUN_DATE_ADR,
                    RUN_DATE_CNT, RunDate () )

  Select Case Rc

```

```
Case SGL_SUCCESS
Text1.Caption = RunDate(0)&"/"&RunDate(1)&"/"&RunDate(2)
Case SGL_DGL_NOT_FOUND
Text1.Caption = "SG-Lock not found!"
Exit Sub
Case Else
Text1.Caption = "Error" & Rc & " occurred!"
Exit Sub
End Select

' new date successfully written , lets do the next thing ...

End Sub
```

6.6 Authentification stimulation-réponse d'un SG-Lock

6.6.1 C/C++

```

#include <time.h>
#include <stdlib.h>
#include "SGLW32.h"
#define PROD_ABC_ID          1
#define TEA_KEY_NUM         1
#define CRYPT_MODE_ENCRYPT   0

unsigned int RC;
unsigned long int RandomNumber[2]; // test random number
unsigned long int RanSglResult[2]; // encryption result
                                     // of SG-Lock
unsigned long int RanAppResult[2]; // encryption result
                                     // of application

// 1. step: generate a 128-bit key
// (NOT for U2/L2 - fixed key!)
unsigned long int TEA_Key[4]={ 0x238A3F10,
                               0x61EAB67A,
                               0x092E1CD2,
                               0x832FAEC3 };

// ATTENTION ! ATTENTION ! ATTENTION ! ATTENTION !
// Do this only once when initialising the key prior to
// delivery of the dongle and NOT in the protected application!
// Writing the key into the SG-Lock modul
RC = SglWriteKey( PROD_ABC_ID, TEA_KEY_NUM, TEA_Key );
if( RC != SGL_SUCCESS ) {
    printf("SglWriteKey :_Error!_(code:_%d)\n", ReturnCode);
}
// ATTENTION END

// 2. step: generate two 32-bit (= one 64-bit) random numbers
srand( clock() ); // force every time different
                 // start of sequence
RandomNumber[0] = rand() << 16 | rand();
RandomNumber[1] = rand() << 16 | rand();

// 3. step: encrypt the random number in the SG-Lock modul
RanSglResult[0]= RandomNumber[0];
RanSglResult[1]= RandomNumber[1];

```

```
RC = SglCryptLock( PROD_ABC_ID,
                  TEA_KEY_NUM,    // number of key
                  CRYPT_MODE_ENCRYPT, // encrypt
                  1,              // block count
                  RanSglResult );

// 4. step: encrypt the random number in the protected
// application
SglTeaEncipher( RandomNumber, RanAppResult, TEA_key );

// 5. Step: compare both results
if( ( RanSglResult[0] != RanAppResult[0] ) ||
    ( RanSglResult[1] != RanAppResult[1] ) ) {

    // authentication failed !!
    printf( "SG-Lock_Modul_authentication :_Error!\n" );

}

// authentication successful ...
```

Le CD-Rom SG-Lock contient d'autres exemples de programmation ainsi que les fichiers Include nécessaires au procédé.

7 Données techniques

7.1 SG-Lock U2/U3/U4

Connexion	USB
type de mémoire	mémoire vive (RAM) non volatile
mémoire	U2 : pas de mémoire U3 : 256 octets U4 : 1024 octets
Compteur 32 bits	U2 : pas de compteur U3 : 16 U4 : 64
Clé 128 bits	U2 : 1 (fixe) U3 : 2 (librement programmable) U4 : 16 (librement programmable)
Algorithme	TEA
cycles de lecture	illimités
cycles d'écriture	> 1.000.000
stockage de données	chiffré à 128 bits
Conservation de données	> 20 ans
Consommation	< 50 mA
Température de fonction	0 to 70°C
Température d'emmagasinage	-30 to 70°C
dimensions	47 × 16 × 8 mm (L×B×H)
poids	5 g

Notes

