Sistema di protezione anticopia software

Manuale dello sviluppatore

per Microsoft Windows XP a 10 (tutte 32/64-Bit), CE, Linux e Mac OS X

Sistema di protezione anticopia software

Manuale dello sviluppatore

per Microsoft Windows XP a 10 (tutte 32/64-Bit) CE, Linux e Mac OS X

Stato: Febbraio 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/it e-mail: info@sg-intec.de

SG-Lock è soggetto della direttiva Ue per gli apparecchi elettrici. Le corrispondenti tasse di smaltimento sono state pagate. WEEE-ID: DE 39431724

Indice

1 Introduzione

SG-Lock è un innovativo e flessibile sistema di protezione anticopia e crittografico basato su hardware che può trovare impiego per tutti i sistemi operativi Microsoft Windows a 32/64 bit.

Le proprietà eccezionali sono:

- Ogni SG-Lock possiede un numero di serie individuale.
- Fino a 1024 Byte di memoria interna dello SG-Lock utilizzabile liberamente.
- Codificazione a 128 bit con fino a 16 chiavi selezionabili liberamente.
- Fino a 64 cellule numeriche liberamente programmabili per il semplice rilevamento di eventi numerabili.
- Gli SG-Lock USB possono essere installati sul sistema di destinazione senza nessun'installazione di un driver e senza diritti d'amministratore (Windows XP a 10).

Particolari caratteristiche di sicurezza:

- L'intera memoria interna del modulo è trasparente per l'utente e codificata con una chiave a 128 bit individuale per ogni modulo e anche ulteriormente firmata. Attacchi all'hardware come pure delle manipolazioni di singoli valori dati oppure lo scambio dell'unità di memoria vengono individuati e respinti dal processore del modulo.
- Meccanismo di autenticazione semplice ed efficace fra applicazione protetta e API SG-Lock. L'API SG-Lock non è come spesso implementato subito utilizzabile da ogni applicazione in piena funzionalità. In linea di

massima ogni applicazione deve autenticarsi verso l'API SG-Lock per ottenere l'accesso ai moduli SG-Lock. Questo impedisce gli attacchi di programmi non autorizzati tramite l'interfaccia API ai moduli di protezione anticopia SGLock. In più l'applicazione protetta ha la possibilità di verificare l'API SGLock da sé e di individuare una libreria falsificata. L'intero meccanismo di autenticazione viene effettuato tramite il richiamo di una singola funzione con un singolo parametro.

- L'API SG-Lock lavora con un engine crittografico TEA (Tiny Encryption Algorithm) all'interno del modulo come pure dell'applicazione. Questo algoritmo crittografico simmetrico (le chiavi per la codificazione e la decifrazione sono identiche) e generalmente riconosciuto come sicuro costituisce la base per l'implementazione di molteplici strategie per la protezione di dati e di codici e anche per l'autenticazione.

2 Installazione e programmi ausiliari

L'installazione dello SG-Lock è resa in modo semplice e trasparente per facilitare l'integrazione nell'installazione dell'applicazione protetta.

2.1 Windows XP a 10

Per l'installazione di moduli USB SG-Lock vanno eseguiti 2 passi.

1. Copiate la libreria SG-Lock SGLW32.DLL nella directory di installazione dell'applicazione protetta oppure nella directory di sistema (p.es. C:\WINDOWS\SYSTEM32 con Windows XP a 10 con l'ulteriore tenete presente i diritti di scrittura!).

2. Inserite lo SG-Lock USB nell'interfaccia USB. Con Windows XP a 10 il riconoscimento hardware viene eseguito e indicato automaticamente – con questo l'installazione è conclusa.

2.2 Linux

Le istruzioni d'insediamento per Linux possono esser trovate sul CD-ROM di SG-Lock.

2.3 Mac OS X

Le istruzioni d'insediamento per MAC OS X possono esser trovate sul CD-ROM di SG-Lock.

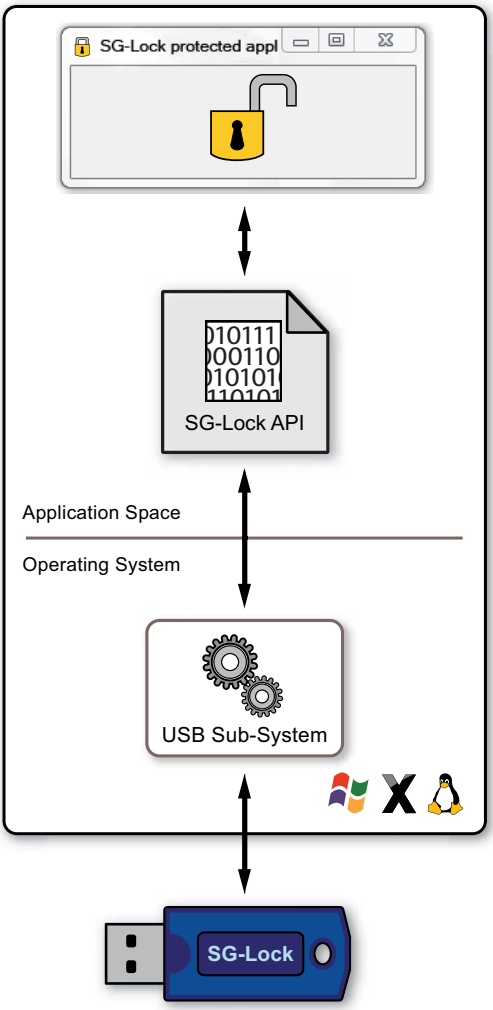


Figura 2.1: L'API SG-Lock (il file SGLW32.DLL) mette a disposizione il collegamento fra l'applicazione protetta e l'hardware SG-Lock.

2.4 Windows CE 4, 5 e 6

1. Copiate il file SGLWCE.DLL nella vostra directory di applicazione o di sistema (p.es. \Windows). Questo può avvenire tramite uno Script durante l'avvio del sistema.

2. Copiate il file SGLUSB.DLL in una directory esistente durante l'avvio del sistema (p.es. \STORAGE).

3. Adattate i due key col nome "'DLL"' nello script di registrazione SGLUSB.REG secondo il percorso di SGLUSB.DLL (se p.es. il vostro SGLUSB.DLL si trovasse in \STORAGE, i valori dei due key devono essere STORAGE\SGLUSB.DLL). Con questo non utilizzate nessun backslash all'inizio. Lasciate invariati i key PREFIX.

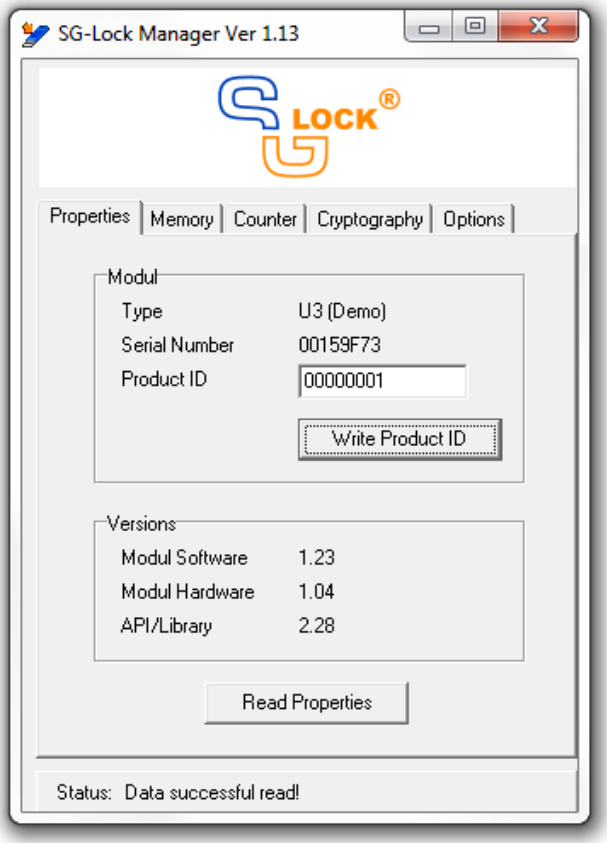
4. Eseguite l'adattato script di registrazione e memorizzate il registry (p.es. con lo AP CONFIG MANAGER sulla cartella APSystem, pulsante STORAGE/REGISTRY/SAVE), in modo che le chiavi rimangano invariate per i seguenti avvi di sistema.

2.5 Disinstallazione per tutti i sistemi

La disinstallazione avviene tramite la semplice cancellazione dei dati indicati e – se essa è stata eseguita prima per l'installazione – la cancellazione delle entrate indicate nei suddetti script di registrazione.

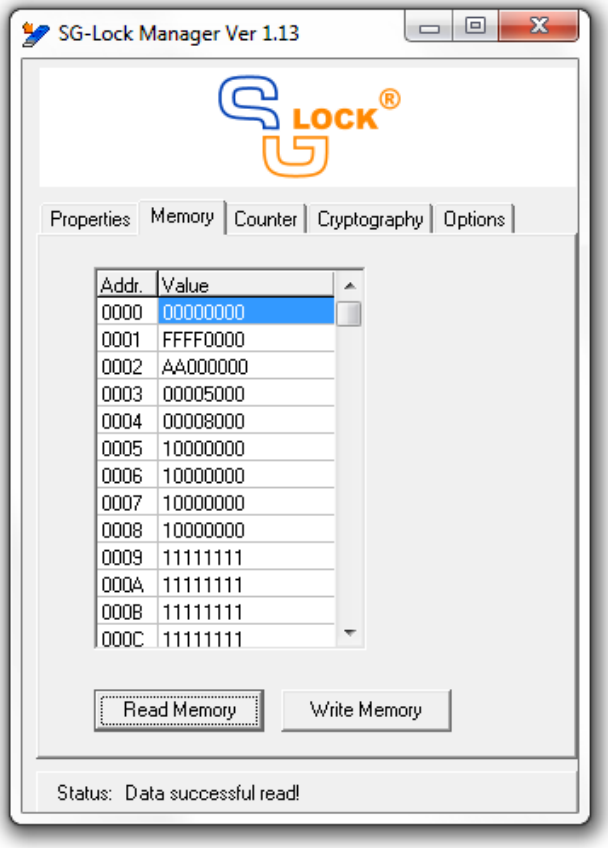
2.6 Lavorare allo SG-Lock con il manager SG-Lock

Il manager SG-Lock (SGLM) è un programma ausiliare fornito ulteriormente sul CD-Rom di SG-Lock per l'elaborazione e la prova di tutti i moduli SG-Lock. Avviate il SGLM eseguendo il file *SglMgr.Exe* dalla directory Test. Sulla cartella *Options* può essere adattata la lingua tramite il campo di selezione *Select Language*. In più su questa cartella può essere adattata la rappresentazione input/output di numeri come numeri decimali o esadecimali. Essa va rispettata anche per l'**input** di numeri!



Tutte le funzioni offerte dallo SGLM fanno parte dell'API SG-Lock e possono essere utilizzate anche da ogni applicazione protetta. La cartella *Properties* offre l'indicazione di informazioni importanti come tipo, numero di serie, ProductID e numeri di versione del collegato SG-Lock tramite il pulsante Read *Properties*.

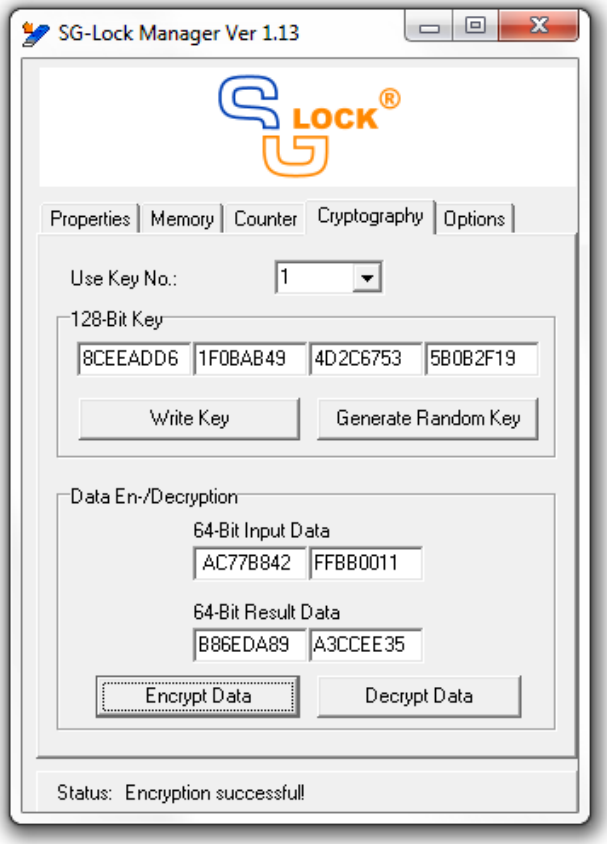
Tramite il pulsante *Write Product ID* il ProductID può essere cambiato a valori da 0 e 65535 (dec.). La funzione del ProductID è descritta in modo più preciso nel paragrafo 3.3.



La cartella *Memory* permette sia la lettura che la scrittura della memoria interna del modulo (se esiste). Per cambiare una o più posizioni di memoria i valori

desiderati vengono iscritti nella tabella e scritti nella memoria interna del modulo tramite il pulsante *Write Memory*.

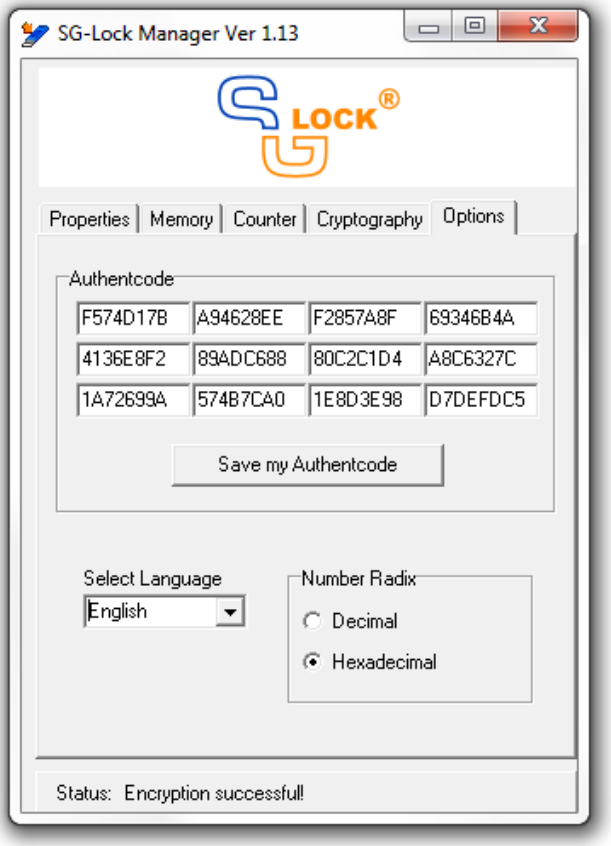
La cartella *Counter* offre le stesse possibilità per le posizioni di memoria del contatore. Esse non sono inserite nella memoria normale ma usano una memoria aggiuntiva che esclude interferenze fra memoria dati e memoria contatore.



La cartella *Cryptography* offre la possibilità di eseguire le funzioni crittografiche di SG-Lock. SG-Lock usa una crittografia a blocchi a 64 bit interna al modulo e simmetrica (vale a dire che le chiavi per la codificazione e la decifrazione sono identiche). La lunghezza della chiave è di 128 bit. L'algoritmo

crittografico è TEA. I tipi SG-Lock delle serie 3 e 4 hanno più memorie chiave. Sulle prime a *Use Key No.* si sceglie la chiave che va cambiata o usata per la codificazione.

Tramite il pulsante *Generate Random Key* è generata una chiave casuale a 128 bit. Essa può essere lavorata ulteriormente tramite la cartella "'appunti"' p.es. per fini di documentazione (è raccomandabile perché per motivi di sicurezza le chiavi possono essere **solo** scritte e **non** lette). Essa poi può essere scritta nella memoria chiave interna del modulo tramite *Write Key*.



Per criptare o decifrare dei dati di test vanno inseriti due valori 32-bit nei due

campi *Input Data*. Cliccando *Encrypt Data* o *Decrypt Data* essi vengono indicati nei campi *Output Data* con i dati criptati o decifrati all'interno del modulo con la chiave preselezionata.

Sulla cartella *Options* possono essere stabilite la lingua e la rappresentazione di numeri desiderate. Nel caso di rappresentazione di numeri esadecimali tutti i numeri a eccezione dei numeri versione sono indicati rispettivamente sulla cartella *Properties*. Poi anche l'input avviene nella versione esadecimale senza simboli speciali all'inizio o alla fine.

Attenzione: L'input di un codice autenticazione (AC) sarebbe necessario nel caso che dovessero essere impiegati dei moduli SG-Lock non-demo (retail). Senza l'input di un AC vengono individuati solo i moduli demo. Ogni produttore software che impiega SG-Lock ottiene con la prima fornitura una sola volta un AC assegnato individualmente che gli assicura l'accesso esclusivo ai suoi moduli SGLock. Tutti i moduli forniti in seguito sono inizializzati con lo stesso AC. Per ottenere con lo SGLM l'accesso anche ai moduli retail va inserito e memorizzato una sola volta l'AC.

Attenzione: L'AC viene comunicato nella versione esadecimale – la rappresentazione di numeri va eventualmente adattata per l'input dell'AC.

3 Proteggere il software con SG-Lock

3.1 Note generali

Il funzionamento di SG-Lock da protezione anticopia si basa su un collegamento realizzato tramite il richiamo delle funzioni determinate, fra il software facilmente copiabile e allora da proteggere e l'hardware SG-Lock in pratica non copiabile. Le funzioni usate per questo sono le funzioni dell'API SG-Lock (Application Programming Interface). Sono contenute nel software ulteriormente fornito con l'hardware SG-Lock – nella libreria di software SGLW32.DLL.

L'API SG-Lock mette a disposizione diversi tipi di funzioni di cui vengono impiegate differenti a secondo il tipo della protezione desiderata. Per un'efficace protezione anticopia non è necessario usare tutte quelle funzioni.

3.2 Strategie di protezione

La variante più comune per proteggere software dall'utilizzo non conforme al contratto è la semplice protezione anticopia che dovrebbe evitare che il software sia in corso su più computer che secondo l'accordo contrattuale. In questo caso per il software è al primo piano il controllo se è installato uno SG-Lock sul computer. Altre strategie di protezione dovrebbero rendere possibile che un software permetta solo un numero limitato di avvi. In questo caso tranne la presenza dello SG-Lock va inoltre controllato un contatore o meglio una variabile contatore, per impedire l'ulteriore effettuazione del software nel caso del superamento di un numero determinato.

Una simile limitazione del corso può essere la possibilità dell'effettuazione di un programma solo fino ad una data stabilita. In questo caso nello SG-Lock va depositata la data corrispondente nella memoria dati e va controllata durante ogni corso. Eventualmente l'utente deve pagare per l'ulteriore uso, così che viene memorizzata una nuova data, ed il corso ricomincia daccapo (pay per use).

Un'altra variante è la possibilità di lasciar correre il programma illimitatamente, ma comprendere nel conto l'uso di singole funzioni e farne a distanze regolari i conti a pagamento.

3.3 Il ProductID SG-Lock – per cosa mi serve?

Spesso un produttore offre diversi pacchetti software e/o delle versioni ampliate di essi. Se più di quelle applicazioni diverse impiegassero una protezione anticopia di un offerente, ne deriverebbe dell'onere amministrativo di programmazione, perché nel caso di un'interrogazione per prima cosa va controllato se il relativo modulo di protezione anticopia fa parte del pacchetto software in corso ossia se sostiene la versione ampliata.

Questo onere amministrativo con le possibili intersezioni e fonti di errori può essere risolto con SG-Lock semplicemente tramite l'assegnazione di un ProductID. L'API SG-Lock decide sulla scorta del ProductID se l'appartenente modulo di protezione anticopia è infilato o no.

Esempio: la ditta X ha in programma tre pacchetti software A, B e C. Ai prodotti A, B e C vengono assegnati rispettivamente i ProductID 1, 2 e 3. Dall'utente viene avviato il software B, e sono inseriti nel computer tre moduli SG-Lock per tutti i tre pacchetti.

Senza la possibilità dell'uso di un ProductID allora nel caso del richiamo di una funzione API ognuno dei tre moduli dovrebbe consecutivamente essere chiamato e controllato se si trattasse del modulo di protezione anticopia appartenente al software, solo dopo può essere letto p.es. il numero di serie.

Con l'uso del ProductID dell'API SG-Lock la procedura di interrogazione è sensibilmente semplificata e meno pregna di errori. Durante l'interrogazione del modulo di protezione anticopia del software B l'API SG-Lock attende l'appartenente ProductID come parametro e ottiene adeguatamente l'assegnato valore 2. Risultato: il modulo dell'applicazione B viene trovato, e gli altri due sono tolti dall'API SG-Lock. L'applicazione B lavora virtualmente su un computer in cui è sempre inserito al massimo 1 modulo di protezione anticopia.

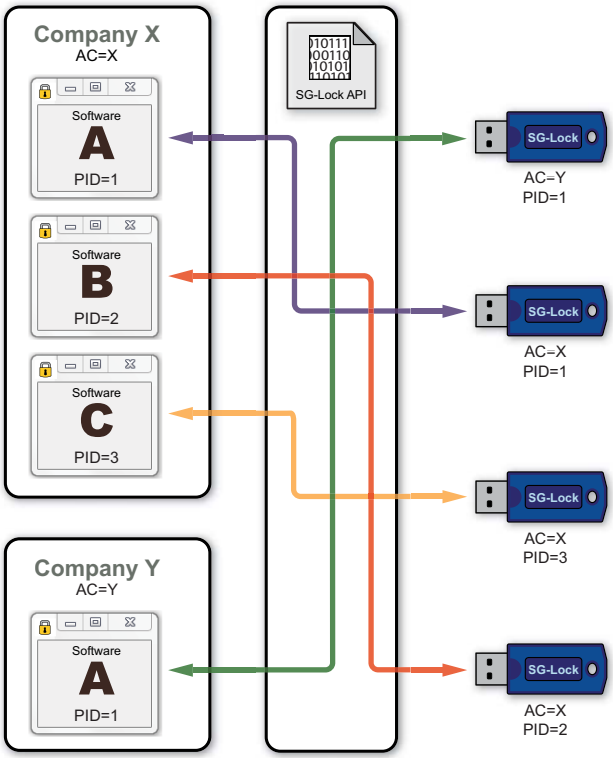


Figura 3.1: Il ProductID SG-Lock rende possibile la semplice distinzione di prodotti diversi di un produttore. Il codice autenticazione divide tassativamente i produttori fra loro.

3.4 Codificazione e autenticazione challenge/response

La codificazione TEA (Tiny Encryption Algorithm) usata da SG-Lock è adatta bene per la codificazione di dati importanti – ma non solo per questo. È possibile migliorare ancora ogni – anche semplice – protezione di software con SG-Lock impiegando l'integrata codificazione TEA.

Il concetto di sicurezza che ci sta dietro è un'autenticazione del modulo di protezione anticopia per mezzo della codificazione corretta di un numero casuale e di una chiave conosciuta ad entrambe le parti ma oltre a ciò segreta.

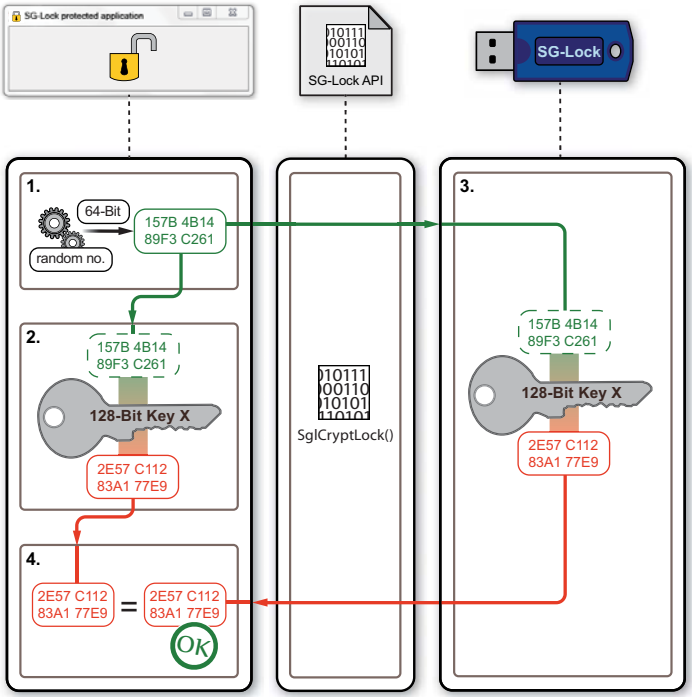


Figura 3.2: L'autenticazione challenge/response SG-Lock mette a disposizione un collegamento sicuro fra l'applicazione protetta (file EXE) attraverso tutto il sistema operativo ed i bus USB fino al modulo SG-Lock stesso.

Questo sistema nella codificazione è noto come autenticazione challenge/response (sfida/risposta) e diffuso molto. La chiave a 128 bit ci serve da password che deve essere nota allo SG-Lock autenticante ossia deve esserci nella memoria

chiave. Durante il processo di autenticazione però la password stessa non viene passata perché così sarebbe intercettabile nel canale di trasmissione e allora non più segreta, ma viene controllata solo la conoscenza della password ossia l'esistenza della chiave corretta.

La procedura è la seguente (esempio di programmazione: vedere paragrafo 6.6):

1. Generate (p.es. con il manager SG-Lock) una chiave a 128 bit casuale, programmatela nel modulo SG-Lock (funzione API *SglWriteKey*) e dichiarate in più essa come costante nel vostro programma da proteggere. Con questo la chiave è nota ad entrambe le parti (programma protetto e modulo SG-Lock). Attenzione: questo passo salta per i moduli di serie 2 perché le chiavi programmate dal produttore di questa serie non possono essere sovrascritte. Utilizzate la chiave del produttore, ve la viene comunicata separatamente con la prima fornitura. I moduli demo hanno chiavi proprie che sono indicate nel capitolo 5.

2. Generate un numero casuale a 64 bit (in pratica due numeri casuali a 32 bit) nel vostro programma da proteggere, con una funzione da generatore di numeri casuali del vostro linguaggio di programmazione. Ci usate – se è possibile – un'altra funzione che garantisce che dopo l'avvio del programma venga cominciato sempre con altri numeri (appunto "'seed"'), perché in caso contrario si presenteranno sempre le stesse sequenze di numeri casuali, cosa diminuisce la protezione.

3. Adesso criptate il numero casuale a 64 bit con la funzione API SG-Lock *SglCryptLock* e memorizzate il risultato. Questa codificazione viene eseguita **nel modulo di protezione anticopia SG-Lock.**

4. Poi criptate lo stesso numero casuale a 64 bit (non il risultato precedente) con la funzione *SglTeaEncipher* contenuta nel file include SglW32, altrettanto per mezzo della chiave a 128 bit generata in precedenza. Memorizzate anche questo risultato per poter metterlo a confronto nel prossimo passo. Questa codificazione si eseguirà **nell'applicazione protetta**.

5. Adesso confrontate se il risultato crittografico del modulo corrisponde al risultato (corretto) della codificazione dell'applicazione. L'autenticità è valida solo nel caso di due risultati identici perché tutte le due codificazioni evidentemente sono state eseguite con l'applicazione della corretta chiave

a 128 bit la quale al di fuori dell'applicazione sussiste solo nel richiesto modulo SG-Lock.

Il confronto dei due risultati crittografici può essere distribuito su più parti del programma confrontando il risultato complessivo oppure solo parti singoli di esso (p.es. la prima sequenza a 16 bit) su un punto del programma e controllando più tardi (ancora una volta) il risultato complessivo ossia le parti rimanenti. Il togliere del confronto e con questo la protezione anticopia nel codice macchina dell'applicazione protetta così viene reso più difficile. Alternativamente si può criptare anche più blocchi a 64 bit in un unico passo e confrontarli in più passi.

3.5 Incorporamento nei diversi linguaggi di programmazione

Le funzioni dell'API SG-Lock possono essere richiamate direttamente nel programma protetto. Per questo i file include sul CD-Rom di SG-Lock ulteriormente fornito vanno incorporati nel testo di programma a secondo il linguaggio di programmazione. C#, Visual Basic, Delphi e alcuni altri contengono già l'indicazione per il linker che le funzioni SG-Lock vanno trovate nella libreria esterna (a volte chiamata anche "'third party DLL"') SGLW32.Dll. Se sussistesse C/C++ nel caso del solito legame statico, andrebbe ancora comunicato al linker per mezzo di una cosiddetta import libreria che le funzioni dell'API SG-Lock vanno trovate in una libreria esterna, cioè quella SGLW32.Dll. Questo avviene includendo nel progetto anche l'import libreria SGLW32.Lib nella lista delle librerie da tenere presenti. Se questo non avvenisse, il linker interromperebbe il processo di linking con la segnalazione di errore di non aver trovato le usate funzioni SG-Lock. Sfortunatamente il formato di file di import librerie è diverso da compilatore a compilatore. Le import librerie dei compilatori più correnti si trovano sul CD-Rom di SG-Lock. Per il caso che per un compilatore non ci fosse reperibile nessuna libreria adatta di regola si potrebbe generarla da sé perché è allegato a ogni compilatore un relativo tool. Per questo consultate la documentazione del compilatore (appunto possibile: "'generazione di una libreria"'). Alternativamente a ciò la libreria SG-Lock può essere incorporata anche in modo dinamico, vale a dire al tempo di transito con le funzioni Win32 *LoadLibrary()* und *GetProcAddress()*. Informazioni più precise trovate nella documentazione SDK di Microsoft Windows.

4 L'API SG-Lock

4.1 Prospetto delle funzioni

Le funzioni dell'API SG-Lock possono essere divise nei quattro gruppi seguenti: funzioni di base, funzioni ampliate e crittografiche come pure le funzioni amministrative. Le funzioni di base che sono fondamentali in merito al loro modo di lavorare, come p.es. l'interrogazione se uno SG-Lock è veramente collegato a un PC, sono impiegate praticamente per ogni tipo di protezione di software. Le funzioni ampliate con possibilità particolari mettono a disposizione delle funzionalità apposite per usi specifici, come p.es. memorie e contatori di accessi con cui p.es. possono essere memorizzati string oppure può essere contato e limitato il numero dei richiami di un programma. Le funzioni crittografiche rendono possibile la codificazione e la firma di qualsiasi tipo di dati come testi, immagini, e-mail, film ecc. Funzioni di questo gruppo sono incluse nel codice di programmazione solo a secondo della strategia di protezione e commercializzazione.

La serie delle funzioni amministrative è prevista prima di tutto come supplemento per la preparazione dei moduli SG-Lock alla consegna con il software protetto, e di regola non è inclusa nel codice di programmazione dell'applicazione protetta. Con questo si può piuttosto realizzare velocemente semplici programmi di inizializzazione individualmente per tale fine.

Nome della funzione	Descrizione
Funzioni di base	
SglAuthent	Autenticazione della libreria SG-Lock.
SglSearchLock	Cerca un'unità SG-Lock.
SglReadSerialNumber	Legge il numero di serie di un'unità SG-Lock.
Funzioni ampliate	
SglReadData	Legge dati dalla memoria di un'unità SG-Lock.
SglWriteData	Scrive dati nella memoria di un'unità SG-Lock.
SglReadCounter	Legge un valore contatore da un'unità SG-Lock.
SglWriteCounter	Scrive un valore contatore in un'unità SG-Lock.
Funzioni crittografiche	
SglCryptLock	Cripta e decifra uno o più blocchi di dati con un'unità SG-Lock per mezzo di una chiave a 128 bit interna all'unità.
SglSignDataApp	Firma dati col PC.
SglSignDataLock	Firma dati con un'unità SG-Lock.
SglSignDataComb	Firma dati in combinazione con PC e unità SG-Lock.
Funzioni amministrative	
SglReadProductId	Legge il ProductID da un'unità SG-Lock.
SglWriteProductId	SScrive un ProductID in un'unità SG-Lock.
SglWriteKey	Scrive una chiave a 128 bit in un'unità SG-Lock.
SglReadConfig	Legge dati di configurazione dall'ambiente SG-Lock oppure da un'unità SG-Lock (p.es. il tipo di un'unità SG-Lock).

4.2 Funzioni di base

4.2.1 Funzione: SglAuthent

Descrizione

Autenticazione della libreria SG-Lock verso l'applicazione protetta e viceversa.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

AuthentCode	Serie a 48 byte che viene assegnata individualmente da SG-Lock ad ogni utente.

Valori restituiti

SGL_SUCCESS	Autenticazione con successo.
SGL_AUTHENTICATION_FAILED	Autenticazione fallita.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Per prima cosa va richiamata con successo questa funzione dell'API SG-Lock perché tutte le altre funzioni dell'API SG-Lock non vengono abilitate prima. Nel caso di un legame dinamico questo va fatto per ogni processo di legame (richiamo di LoadLibrary). I kit DEMO hanno un proprio codice di autenticazione che è inserito negli esempi di programmazione.
Attenzione: un'interrogazione se un modulo di protezione anticopia SG-Lock è inserito non avviene tramite questa funzione!

4.2.2 Funzione: SglSearchLock

Descrizione

Cerca uno SG-Lock.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.

Valori restituiti

SGL_SUCCESS	SG-Lock trovato.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Funzione di standard per controllare se è inserito un modulo SG-Lock.

4.2.3 Funzione: SglReadSerialNumber

Descrizione

Legge il numero di serie individuale per ogni SG-Lock.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock.
SerialNumber	Puntatore sulla variabile che viene ridata tramite il numero di serie dello SG-Lock.

Valori restituiti

SGL_SUCCESS	Numero di serie SG-Lock letto con successo.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Ogni modulo SG-Lock ha, indipendentemente dalla variante dell'interfaccia USB o LPT, un individuale numero di serie (non vale per i moduli DEMO). Questo individuale numero di serie può servire non solo per l'inequivocabile identificazione di un modulo SG-Lock ma anche da "'Master Value"' per derivare in modo protetto degli individuali numeri, chiavi e codici di ogni tipo tramite delle adatte funzioni di conversione.

4.3 Funzioni di memoria

4.3.1 Funzione: SglReadData

Descrizione

Legge valori dati a 32 bit dalla memoria SG-Lock.

Modelli

U2: - U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.
Address	Indirizzo di partenza del blocco di valori dati da 0 a 63 – SG-Lock U3 da 0 a 255 – SG-Lock U4
Count	Numero dei valori dati.
Data	Puntatore sul campo di dati nel quale vanno copiati i valori dati (lo sviluppatore deve assicurare la dimensione sufficiente del campo di dati).

Valori restituiti

SGL_SUCCESS	Valori dati letti con successo dallo SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

La memoria dati interna al modulo può essere utilizzata per la memorizzazione sicura di qualsiasi tipo di dati come codici, numeri, chiavi, password, licenze ecc.

4.3.2 Funzione: SglWriteData

Descrizione

Scrive valori dati a 32 bit nella memoria SG-Lock.

Modelli

U2: - U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.
Address	Indirizzo di partenza del blocco di valori dati da 0 a 63 – SG-Lock U3 da 0 a 255 – SG-Lock U4
Count	Numero dei valori dati.
Data	Puntatore sul campo di dati dal quale vanno rilevati i valori dati (lo sviluppatore deve assicurare la dimensione sufficiente del campo di dati).

Valori restituiti

SGL_SUCCESS	Valori dati scritti con successo nella memoria SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

vedere *SglReadData*.

4.3.3 Funzione: SglReadCounter

Descrizione

Legge un valore contatore a 32 bit dalla memoria SG-Lock.

Modelli

U2: - U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.
CntNum	Numero del contatore da 0 a 15 – SG-Lock U3 da 0 a 63 – SG-Lock U4
Data	Puntatore sulla variabile che deve assumere il valore contatore.

Valori restituiti

SGL_SUCCESS	Valore contatore letto con successo dallo SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Counter sono campi di dati a 32 bit nella memoria SG-Lock che possono essere usati non solo come contatori ma anche per tutte le altre applicazioni che permettono delle variabili di lettura/scrittura. Così la memoria del modulo può essere ampliata ulteriormente.

4.3.4 Funzione: SglWriteCounter

Descrizione

Scrive un valore contatore a 32 bit nella memoria SG-Lock.

Modelli

U2: - U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.
CntNum	Numero del contatore da 0 a 63 – SG-Lock U3 da 0 a 255 – SG-Lock U4
Data	Valore contatore da scrivere.

Valori restituiti

SGL_SUCCESS	Valore contatore scritto con successo nello SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

vedere *SglReadCounter*.

4.4 Funzioni crittografiche e di firma

4.4.1 Funzione: SglCryptLock

Descrizione

Cripta o decifra uno o più blocchi di dati a 64 bit con chiavi a 128 bit. L'algoritmo è TEA.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock richiesto.
KeyNum	Numero della chiave da da 0 a 1 – SG-Lock U3 da 0 a 15 – SG-Lock U4.
CryptMode	Modalità di lavoro 0 – criptare 1 – decifrare.
BlockCnt	Numero dei blocchi di dati a 64 bit da trattare.
Data	Puntatore sul campo di dati nel quale si trovano i blocchi di dati da trattare (lo sviluppatore deve assicurare la dimensione sufficiente circa il parametro BlockCnt).

Valori restituiti

SGL_SUCCESS	Codificazione eseguita con successo.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

La funzione sovrascrive i valori di partenza trasferiti tramite il parametro *Data*. Se essi dovessero essere lavorati ulteriormente in altro modo, dovrebbero essere salvati prima del richiamo della funzione.

4.4.2 Funzione: SglSignDataApp

Descrizione

Firma o controlla la firma di un campo di dati. Si svolge completamente e senza l'intervento di uno SG-Lock nell'applicazione protetta. La firma ha una lunghezza di 64 bit.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

```
ULONG SglSignDataApp(
        ULONG *AppSignKey,
        ULONG Mode,
        ULONG DataLen,
        ULONG *Data,
        ULONG *Signature );
```

Parametro

AppSignKey	Chiave di applicazione a 128 bit da impiegare con la generazione o il controllo. Puntatore sul campo di dati di quattro valori a 32 bit che formano la chiave a 128 bit.
Mode	Modalità di lavoro 0 – generare la firma 1 – controllare la firma.
DataLen	Numero dei valori a 32 bit del campo di dati.
Data	Puntatore sul campo di dati (Array) con dati da firmare.
Signature	Puntatore sul campo di dati nel quale o la firma generata viene ridata o la firma viene trasmessa per una verifica. Puntatore su un campo di dati con due valori a 32 bit (lo sviluppatore deve assicurare la dimensione ufficiente circa il parametro DataLen).

Valori restituiti

SGL_SUCCESS	Firma generata con successo, firma valida.
SGL_SIGNATURE_INVALID	Firma non valida.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Vantaggio: siccome la funzione si svolge nell'ambito dell'applicazione, è possibile firmare e controllare velocemente grandi quantità di dati.
Svantaggio: la chiave è contenuta nell'applicazione, ossia eventualmente potrebbe essere trovata nel codice macchina.

4.4.3 Funzione: SglSignDataLock

Descrizione

Firma o controlla la firma di un campo di dati. Tutte le due procedure vengono eseguite con lo SG-Lock collegato. La firma ha una lunghezza di 64 bit.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

```
ULONG SglSignDataLock(
        ULONG ProductId,
        ULONG LockSignKeyNum,
        ULONG Mode,
        ULONG DataLen,
        ULONG *Data,
        ULONG *Signature );
```

Parametro

ProductId	Indica il ProductID dello SG-Lock da impiegare.
LockSignKeyNum	Il numero della chiave a 128 bit nello SG-Lock da impiegare per la generazione o il controllo.
Mode	Modalità di lavoro 0 – generare la firma 1 – controllare la firma.
DataLen	Numero dei valori a 32 bit del campo di dati.
Data	Puntatore sul campo di dati (Array) con dati da firmare.
Signature	Puntatore sul campo di dati nel quale o la firma generata viene ridata o la firma viene trasmessa per una verifica. Puntatore su un campo di dati con due valori a 32 bit (lo sviluppatore deve assicurare la dimensione sufficiente circa il parametro DataLen).

Valori restituiti

SGL_SUCCESS	Firma generata con successo, firma valida.
SGL_SIGNATURE_INVALID	Firma non valida.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Vantaggio: siccome la funzione si svolge nell'ambito dello SG-Lock, la chiave per la firma nello SG-Lock è protetta.
Svantaggio: per effetto della limitata velocità di elaborazione dello SG-Lock possono essere firmate solo piccole quantità di dati.

4.4.4 Funzione: SglSignDataComb

Descrizione

Firma o controlla la firma di un campo di dati. La lavorazione ci viene eseguita sia dall'applicazione (CPU del computer) che dallo SG-Lock collegato. Questa funzione abbina i vantaggi delle due funzioni precedenti – **Condizione importante:** le due chiavi (dell'applicazione e quella interna al modulo SG-Lock) devono essere diverse fra loro! La firma ha una lunghezza di 64 bit.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock da impiegare.
AppSignKey	Chiave di applicazione a 128 bit da impiegare con la generazione o il controllo. Puntatore sul campo di dati di quattro valori a 32 bit che formano la chiave a 128 bit.
LockSignKeyNum	Il numero della chiave a 128 bit nello SG-Lock da impiegare per la generazione o il controllo.
Mode	Modalità di lavoro 0 – generare la firma 1 – controllare la firma.
LockSignInterval	Indica come la velocità di elaborazione per la firma viene divisa fra l'applicazione e lo SG-Lock. Il valore determina come potenza di 2 quale blocco di dati viene lavorato dallo SG-Lock. Esempio: valore=8, 2 alla 8=256, ossia SG-Lock lavora i dati per ogni 256esimo valore dati, tutti gli altri vengono lavorati dall'applicazione (CPU del computer).
DataLen	Numero dei valori a 32 bit del campo di dati.
Data	Puntatore sul campo di dati (Array) con dati da firmare.
Signature	Puntatore sul campo di dati nel quale o la firma generata viene ridata o la firma viene trasmessa per una verifica. Puntatore su un campo di dati con due valori a 32 bit (lo sviluppatore deve assicurare la dimensione sufficiente circa il parametro DataLen).

Valori restituiti

SGL_SUCCESS	Firma generata con successo, firma valida.
SGL_SIGNATURE_INVALID	Firma non valida.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

La funzione combina i vantaggi delle due funzioni descritte precedentemente: siccome per la generazione della firma ci vogliono assolutamente sia la chiave dell'applicazione protetta che quella dello SG-Lock impiegato, l'alta velocità di elaborazione dell'applicazione (CPU del computer) viene abbinata con l'alta sicurezza dello SG-Lock. Lo SG-Lock lavora sempre il primo blocco di dati, e per effetto della concatenazione dei valori dati utilizzata durante la generazione della firma tutti i successivi ne sono dipendenti.
Importante: entrambe le impiegate chiavi a 128 bit (quella dell'applicazione e quella interna al modulo SG-Lock) devono distinguersi l'una dall'altra, e possibilmente non dovrebbero essere simili fra loro per poter sfruttare completamente i vantaggi della funzione.

4.5 Funzioni amministrative

4.5.1 Funzione: SglReadProductId

Descrizione

Legge il ProductID a 16 bit dallo SG-Lock.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Puntatore sulla variabile a 32 bit che deve assumere il ProductID.

Valori restituiti

SGL_SUCCESS	ProductId letto con successo dallo SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Il ProductID serve per la semplificazione dell'amministrazione di più applicazioni protette ossia di versioni ampliate di un'applicazione. Attenzione: Possono essere utilizzate solo i 16 bit inferiori ossia i ProductID dal 0 a 65535. Una descrizione più dettagliata in merito all'impiego del ProductID trovate nel paragrafo 3.3.

4.5.2 Funzione: SglWriteProductId

Descrizione

Scrive un nuovo ProductID a 16 bit nello SG-Lock.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

OldProductId	Indica il ProductID valido finora dello SG-Lock.
NewProductId	Indica il nuovo ProductID dello SG-Lock.

Valori restituiti

SGL_SUCCESS	ProductId scritto con successo nello SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

vedere *SglReadProductId*.

4.5.3 Funzione: SglWriteKey

Descrizione

Scrive una chiave a 128 bit nella memoria chiave dello SG-Lock.

Modelli

U2: - U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock.
KeyNum	Numero della chiave da scrivere da 0 a 1 – SG-Lock U3 da 0 a 15 – SG-Lock U4
Key	Chiave a 128 bit da scrivere. Puntatore sul campo di dati di quattro valori a 32 bit che formano la chiave a 128 bit.

Valori restituiti

SGL_SUCCESS	ProductId scritto con successo nello SG-Lock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

4.5.4 Funzione: SglReadConfig

Descrizione

Preleva configurazione e informazioni tramite SG-Lock.

Modelli

U2: ✓ U3: ✓ U4: ✓

Dichiarazione

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

Parametro

ProductId	Indica il ProductID dello SG-Lock.
Category	Tipo dell'informazione richiesta 0: informazione sul modulo SG-Lock.
Data	Puntatore sul campo di dati di 8 numeri interi con una lunghezza di 32 bit. I significati dei singoli valori sono: Indice 0: tipo Indice 1: interfaccia Indice 2: versione software Indice 3: versione hardware Indice 4: numero di serie Indice 5: grandezza di memoria in DWords Indice 6: numero di contatori Indice 7: numero di chiavi a 128 bit.

Valori restituiti

SGL_SUCCESS	ProductId letto con successo dallo SGLock.
SGL_DGL_NOT_FOUND	SG-Lock non trovato.

L'elenco completo dei valori restituiti si trova nel paragrafo 4.6.

Commento

Ulteriori informazioni circa singoli valori sono elencate nei file include e header dll'API SG-Lock.

4.6 Restituzioni di errori

Ogni funzione dell'API SG-Lock rida un valore restituito in base al quale può essere controllata l'esecuzione priva di errori della funzione. Se comparisse un errore durante l'esecuzione, sarebbe ridato un valore diverso da 0.
Il valore restituito descrive l'errore in modo più preciso.

Restituzione	Valore	Descrizione
SGL_SUCCESS	0	Esecuzione priva di errori.
SGL_DGL_NOT_FOUND	1	SG-Lock non trovato.
SGL_AUTHENTICATION-_REQUIRED	5	Autenticazione con *SglAuthent* in precedenza non è stata eseguita oppure non è stata eseguita in modo privo di errori..
SGL_AUTHENTICATION-_FAILED	6	Autenticazione con *SglAuthent* fallita.
SGL_FUNCTION_NOT-_SUPPORTED	7	La funzione richiamata non è supportata dal modulo SG-Lock trovato (p.es. *SglReadData* nel caso di modulo U2).
SGL_PARAMETERS-_INVALID	8	Un parametro della funzione richiamata non si trova nel consentito campo di valori (p.es. indirizzo 5000 per la funzione *SglReadData*).
SGL_SIGNATURE_INVALID	9	Firma non valida.
SGL_USB_BUSY	10	Il SG-Lock è occupato.

Tabella 4.1: Valori restituiti dell'API SG-Lock e descrizioni appartenenti

5 Codificazione, firma e amministrazione delle chiavi

SG-Lock dispone di una simmetrica (ossia le chiavi per la codificazione e la decifrazione sono identiche) funzione crittografica a blocchi. La grandezza del blocco di dati è di 64 bit ossia 2 parole doppie. La lunghezza della chiave è di 128 bit ossia 4 parole doppie.

La funzione crittografica può essere usata per la codificazione, la decifrazione e la firma di qualsiasi tipo di dati, come p.es. per informazioni di configurazione, dati riservati ecc. La codificazione ci può essere eseguita direttamente nell'hardware SG-Lock, cosa costituisce una sicurezza molto alta perché la chiave non comparisce al di fuori dello SG-Lock e allora non può essere spiata.

La sicurezza alta raggiunta tramite questo funzionamento si accompagna con la limitazione di una velocità di codificazione più bassa di circa 100 blocchi al secondo, ossia circa 0.8 kb/s che deriva dalla limitata velocità di elaborazione dell'hardware SG-Lock. In pratica questa funzione rimane limitata a piccole quantità di dati per quanto viene eseguita una codificazione unicamente interna allo SG-Lock.

Una codificazione con il CPU del PC va considerata più insicura perché la chiave è memorizzata da qualche parte nel sistema e allora sarebbe rintracciabile con relativi tool. Il vantaggio di questa variante è una velocità di codificazione molto alta con più di 10 Mb/s.

Ma è possibile anche eseguire una combinazione della codificazione interna allo SG-Lock e quella interna al PC unendo entrambi i vantaggi – alta velocità di codificazione del CPU del PC e sicurezza alta della codificazione interna allo SG-Lock – in un unico processo. La velocità di codificazione in sostanza è altrettanto alta come nel caso della codificazione unicamente interna al PC. Con questa variante di codificazione tutti i blocchi di dati a 64 bit vengono codificati a uno a uno e collegati fra loro. Il primo blocco e periodicamente pochi altri vengono codificati dall'hardware SG-Lock, tutti gli altri dal CPU del PC. Particolarmente importante per la sicurezza di questa procedura è che le due chiavi a 128 bit

utilizzate (la interna allo SG-Lock e quella interna al PC) sono **diverse** fra loro. Così la chiave interna al PC in linea di principio è ancora rintracciabile, ma per la codificazione interna allo SG-Lock e il collegamento dei blocchi di dati fra loro i dati sono ormai "'pre-criptati"' con un'**altra** (e allora sconosciuta) chiave a 128 bit.

Prima della fornitura tutte le memorie chiave (n. 1, 2 o 16, a secondo il tipo di modulo) vengono inizializzate con chiavi dal produttore. Ogni utente SG-Lock ci ottiene proprie chiavi segrete che all'occorrenza può sovrascrivere con delle chiavi generate da sé stesso (le chiavi dei tipi di modulo U2/L2 non sono sovrascrivibili). Tutti i moduli SG-Lock di un utente ci contengono una identica serie di chiavi. Questa serie di chiavi viene comunicata all'utente con il primo ordine.

Tutti i moduli DEMO (USB e LPT) hanno una individuale serie di chiavi che è rappresentata in seguito.

No	Tipo	Chiave a 128 bit (esadecimale)			
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

Tabella 5.1: chiavi a 128 bit di moduli DEMO programmate dal produttore

6 Esempi di programmazione

6.1 Funzione 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 Funzione 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!! }
```

```
    Memo1.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 Funzione 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!! }
    Memo1.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 Funzione 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!! }
    Memo1.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
' addresse 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 Funzione SglWriteData

6.5.1 C/C++

```
#include "SGLW32.h"
#define PROD_ABC_ID 1
// addresse 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; { addresse 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!! }
    Memo1.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
' addresse 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 & " occured !"
         Exit Sub
      End Select

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

End Sub
```

6.6 Autenticazione challenge/response di un modulo 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 appliction

// 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 ...
```

Altri esempi di programmazione e i necessari dati include si trovano sul CD-Rom di SG-Lock.

7 Dati tecnici

7.1 SG-Lock U2/U3/U4

Porta	USB
Tipo di memoria	RAM non volatile
Memoria	U2: nessuna memoria U3: 256 byte U4: 1024 byte
Contatori a 32 bit	U2: nessun contadore U3: 16 U4: 64
Chiavi a 128 bit	U2: 1 (fissa) U3: 2 (liberamente programmabili) U4: 16 (liberamente programmabili)
Algoritmo	TEA
Cicli di lettura	illimitati
Cicli di scrittura	$>$ 1.000.000
Memorizzazione dati	Codificata a 128 bit
Conservazione dei dati	$>$ 20 anni
Consumo di corrente	$<$ 50 mA
Temperatura di lavoro	da 0 a 70°C
Temperatura di magazzino	da -30 a 70°C
Dimensioni	$47 \times 16 \times 8$ mm (L$\times$P$\times$A)
poids	5 g

Appunti