GNT USB Token

Multifactor authentication

E-mail encryption and digital signature, on-the-fly disk encryption

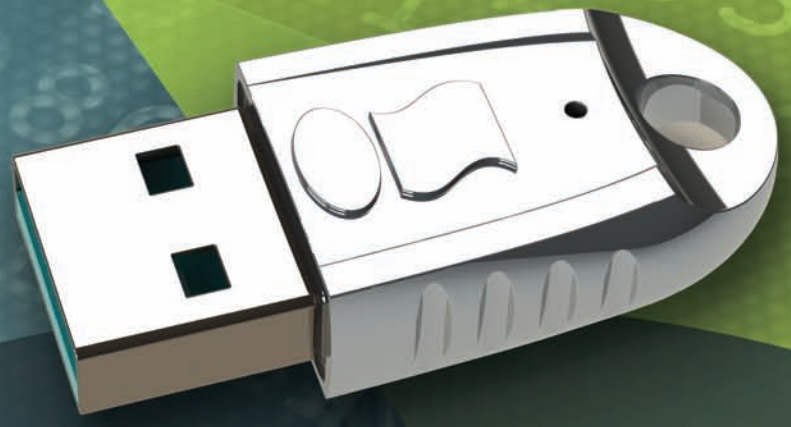
Software and hardware licensing

Processing of classified information

Digital rights management (DRM)

Compatible to PGP, Thunderbird, Firefox, VeraCrypt, OpenVPN

e-Government



Unique serial number

Failed Login Counter

RSA key pair generation up to 4096 bit

128 bit AES, 64 bit DES, 192 bit Triple DES, SHA-1, SHA-256

Random number generator (FIPS 140-2)

State-of-the-art countermeasures against reverse engineering

34,5 KB of secure user memory

Minimum 10 years data retention and 500 000 write/erase cycles

Compatible with PKCS #11 standard

Rugged water proof metal case, LED diode indicating activity

Common Criteria EAL 4+ ready

amset

Tel.: +421-2-44460444 | amset@amset.sk | www.amset.sk

SOFTIDEA

GNT USB Token

datasheet

November 2018

Contents

List of Tables

1 Features

- **Unique serial number**
- **LED diode indicating activity**
- **Wrong Login Counter**
- **Rugged water resistant metal case**
- **2048 bit RSA, key pair generation, encryption, decryption, sign and signature verification, symmetric key wrap and unwrap**
- **128 bit AES in CBC, ECB, CTR, OFB modes**
- **64 bit DES in CBC, ECB, CTR, OFB modes**
- **192 bit Triple DES in CBC, ECB, CTR, OFB modes**
- **SHA-1 and SHA-256**
- **Random number generator (FIPS 140-2)**
- **State-of-the-art countermeasures against reverse engineering**
- **34,5 KB of secured user memory**
- **10 years minimum data retention**
- **500 000 minimum write/erase cycles at 25°C**
- **SDK including**
  - **low level API in C,**
  - **PKCS #11 provider,**
  - **bindings for .NET, C#, Python, Pascal,**
  - **sample code**
- **Compatibility with third party applications including PGP, Thunderbird, Firefox, TrueCrypt and other PKCS #11 enabled applications**
- **Genuine solutions for multifactor workstation logon and email security**
- **CC EAL 4+ ready**

2 Overview

GNT USB Token ("Token") is a PKI hardware security module with *USB 2.0 "Full Speed"* interface dedicated to data encipherment, digital signature and data integrity check. Encipherment prevents unauthorized access to the digital content, digital signature allows for unambiguous identification and authentication of parties and data integrity check allows to detect additional changes in the digital content.

At the core of the Token there is the integrated circuit specially designed to reach extremely high data security level. There are several security features and countermeasures implemented at the physical chip architecture level and along the software stack that ensures that the state-of-the-art data security is achieved.

Token provides mechanisms for asymmetric and symmetric cipher, hash calculation and generation of random number sequences. These operations are executed within the protected firmware environment inside the Token. Access to the protected Token resources and operations is secured using the 128 bit password. To mitigate potential risk of brute force attack to the Token password there is the Wrong Login Counter feature that destroys all sensitive data on the Token if attacked.

To suit the highest security demands the system should be configured in the following manner:

- asymmetric key pair is generated inside the Token,
- asymmetric private key never leaves the Token (it is impossible to extract asymmetric private key off the Token),
- the symmetric secret keys are exported off and imported into the Token in the wrapped form only (enciphered with asymmetric cipher),
- Wrong Login Counter feature is enabled and set to the suitable value.

Token has compact construction designed for everyday carry, is waterproof and durable due to the metal case. On the customer request it is possible to laser engrave any text or graphics (e. g. user name or serial number) onto the metal case of the Token.

In the case of acute threat the Token can be relatively easy physically disposed what makes cryptography keys unreachable and any sensitive data encrypted for these keys forever lost.

3 Security

- Access to data stored in the secured user memory is protected with 128-bit password.
- All cryptographic operations are performed directly in the Token hardware.
- Token can be configured so that the private part of the RSA key pair never leaves the Token.
- Security system can be designed so that the symmetric encryption key leaves Token only in wrapped form.
- There is the Wrong Login Counter feature. After the continuous sequence of *N* incorrect login trials (regardless of password type) the Token automatically destroys all user data and Token returns to the production state. The value *N* of the wrong login counter is defined by administrator during the Token initialization. Setting *N=0* disables Wrong Login Counter

feature.

- The contents of Token memory are encrypted.
- Security policies for usage of RSA keys are enforced at the hardware level. RSA keys are stored in the dedicated memory slots. Each slot has assigned set of non mutable security attributes. Hardware allows or deny the particular operation based on the security attributes.
- Countermeasures against Simple Power Analysis (SPA) and Differential Power Analysis (DPA) are present. Such attacks are based on power consumption measurement during Token usage.
- Countermeasures against voltage, frequency and temperature manipulation based attacks are present. If adversary manipulates supply voltage, frequency of oscillator or Token temperature out of the defined range the Token immediately resets.
- Countermeasures against attempts to influence data using the optical signal are present. Example of such attack is an attempt to erase some layout area using UV radiation.
- Random number generator has analog noise entropy source.
- Countermeasures against Timing Attack are present. Login takes always the same time regardless of the count of correct or incorrect characters in the password.
- Specific layout structures are implemented on-chip dedicated to prevent interception of data flows. The design does not contain visible bus structures. Data and address buses are concealed under the active metal layers.

4 Operations

Operation core of the Token is designed to be very versatile to suit the majority of commonly used data security protocols based on asymmetric and symmetric cryptography and their combinations. The following operations and their combinations are supported:

- Random number sequence generation
- Generation, export, import, deletion of RSA key pair
- Read and write access to sensitive data in the secure non volatile storage
- Hash calculation
- Digital signature generation
- Digital signature verification
- Data encryption and decryption using RSA, AES, TDES, DES
- Simultaneous symmetric data encryption and digital signature generation
- Simultaneous symmetric data decryption and digital signature verification
- Simultaneous import of wrapped symmetric key its decryption and usage for data decryption
- Simultaneous data encryption using symmetric key and export of wrapped symmetric key.

Generation of RSA key pair takes typically couple of seconds however from statistical nature of this operation it implies that it can take even longer. During this time the LED diode stays off.

5 Life cycle

In Figure 1 there is the state diagram of the Token life cycle. Description of states is in Table 1. Transitions between states are explained in the following text.

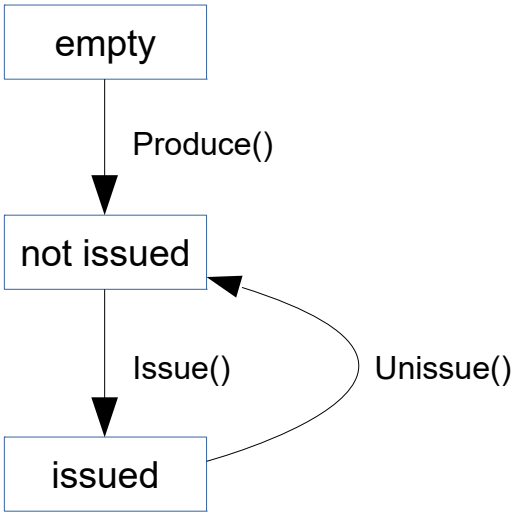


Figure 1: State diagram of the Token life cycle

The *Produce()* transition is performed by the *manufacturer* after authenticated using *production password (PPW)*. During this transition the Token gets assigned the unique s*erial number* and the *administrator password (APW)* is set to its default value.

The *Issue()* transition is performed by the Token *administrator* after authenticated using *APW*. This transition represents Token initialization during which all configurable features including count and attributes of RSA slots, count, length and attributes of MEM areas, default *user passwords (PIN1, PIN2)* and threshold value of *Wrong Login Counter* feature are configured.

The *Unissue()* transition is performed by the Token *administrator* after authenticated using *APW* or is triggered by *Wrong Login Counter* feature after its value exceeds the configured threshold. This transition can also be performed by *manufacturer* after authenticated using *PPW*. During this transition the *APW* is reset to its default value.

Caution: During the Unissue() transition all user data and configuration are irreversibly erased.

Table 1. List of Token states

State	Description
empty	Token has been just manufactured. The user memory is empty and

State	Description
	serial number is not defined. Only allowed operation is *Produce()* transition.
not issued	Default value for *APW* is set, serial number is set. The user memory is empty and Token allows only the following operations: • *administrator* login (using *APW*) • change of *APW* • *Issue()* transition. Any other operations are forbidden.
issued	*User* makes use of Token. Token is fully initialized in accordance with configuration defined by the *administrator* during *Issue()* transition. Token architecture is given by this configuration and cannot be changed. All features are operational.

6 Architecture

Architecture of the Token is defined by the configuration written into the Token during its initialization by *administrator*. *User* cannot change Token architecture. The only way how to change architecture of initialized Token is through *Unissue()-Issue()* transition sequence. Such sequence however destroys all user data stored in the Token.

6.1 Memory

Internal Token memory has length of 36 KB, from which minor part is used for system data and *RSA slot 0* which is always present. The rest of 35328 B is designated as "u*ser memory*" dedicated for user data. Logical structure of *user memory* is disclosed in Table 2:

Table 2. Structure of user memory

User memory						
35328 B						
RSA slots				MEM areas		
RSA slot 1	RSA slot 2	RSA slot 3	...	MEM1	MEM2	MEM3

User memory is configurable during Token initialization (transition *Issue()*) during which *administrator* defines the following parameters:

- number of user defined *RSA slots* in the range 0-32. Each *RSA slot* takes 544 B of *user memory*
- attributes of each user defined *RSA slot* (see 6.2)
- number of *MEM areas* in the range 0-3
- access rights for each *MEM area* (see 6.3)
- length of each *MEM area*, sum of their lengths is limited by available free *user memory*

6.2 RSA slots

Each RSA key pair is stored in the *user memory* in the dedicated *RSA slot*. *RSA slots* store both private and public key and hold attributes of the stored RSA key pair. Count of *RSA slots* is configurable, *administrator* can configure up to 32 user defined *RSA slots* during Token initialization. In addition to user defined *RSA slots* there is the *RSA slot 0* which is always present and its attributes are fixed (not configurable). RSA key pair can be either generated in or imported into the *RSA slot*. Logical structure of *RSA slots* is disclosed in Table 3. Meaning of each of the *RSA slot* attributes and its value for the fixed *RSA slot 0* is disclosed in the Table 4.

Table 3. Structure of RSA slots

RSA slots				
RSA slot 0	RSA slot 1	RSA slot 2	RSA slot 3	...
fixed attributes	configurable attributes	configurable attributes	configurable attributes	configurable attributes

Table 4. Attributes of RSA slots

Attribute	Description	*RSA slot 0*
exportable	If set the complete RSA key pair including private key can be exported out of the Token.	not set
importable	If set the RSA key pair can be imported into the RSA slot.	set
generatable	If set the RSA key pair can be generated in the RSA slot.	set
exportable during generation only	If set the complete RSA key pair including private key can be exported out of the Token but only during its generation. This is useful if creation of backup of the unexportable key is desired.	not set
deletable	If set the RSA key pair can be generated from the RSA slot.	set
PIN1 accessible	If set the RSA key pair stored within the slot can be used by *PIN1* authenticated *user*.	set
PIN2 accessible	If set the RSA key pair stored within the slot can be used by *PIN2* authenticated *user*.	set
encryption allowed	If set the RSA key pair stored within the slot can be used for data encryption and decryption.	set
signature allowed	If set the RSA key pair stored within the slot can be used for data signing and signature verification.	set
wrapping allowed	If set the RSA key pair stored within the slot can be used for wrapping and unwrapping of the symmetric key.	set

6.3 MEM areas

MEM areas are byte addressable memory regions for user data storage. Each *MEM area* has defined access rights. *Administrator* can define up to 3 *MEM areas* during Token initialization. Specification of access rights to the *MEM areas* consists of:

- *user* authentication level required for write and delete access
- *user* authentication level required for read access

Note: To delete data from the MEM area the write access right is required.

It is not possible to specifically grant any access rights for *administrator* (authenticated by *APW)*. A*dministrator* defines access rights during the Token initialization. The desired configuration of *MEM areas* access rights can be described by filling the Table 5.

Table 5. Representation of MEM area access rights

Access right				
MEM area	**-**	**free access**	**PIN1 required**	**PIN2 required**
MEM1	**read**			
	write			
MEM2	**read**			
	write			
MEM3	**read**			
	write			

7 Access

7.1 Access levels

Token recognizes five authentication levels:

- free access (without authentication),
- *user* authenticated by password *PIN1,*
- *user* authenticated by password *PIN2,*
- *administrator* authenticated by password *APW,*
- *manufacturer* authenticated by password *PPW.*

Access rights to the Token resources for different authentication levels are defines in the following text. (Rights related to life cycle transitions are defined in chapter 5).

Without authentication it is possible to query *serial number*, publicly available Token configuration data and to perform some cryptography operations (generate random numbers and calculate hash of the input data).

The only rights of the *manufacturer* are to set the *serial number* and default *APW* during *Produce()* transition.

Administrator can only configure Token architecture during *Issue()* transition. *Administrator* can not do cryptography with RSA keys nor with symmetric algorithms.

User has granted right to do any operations with the Token resources after authenticated for the required level (*PIN1* and/or *PIN2*).

If it is vital for the designed application that *administrator* uses cryptography, it is possible to identify *administrator* with *PIN2* authenticated *user*. Similar concept was used in SIPKCS provider [1].

7.2 Passwords

User can be authenticated to the Token using two passwords *PIN1* a *PIN2*. This can be used for vertical division of access to cryptographic keys. For example one can define *PIN1* accessible RSA key for data encryption and decryption and other *PIN2* accessible RSA key for digital signature.

For password change the following rules are in force:

- *APW* can be changed after authentication to the current *APW* level.
- *PIN2* can be changed after authentication to the current *PIN2* level.
- *PIN1* can be changed after authentication to the current *PIN1* level or after authentication to the current *PIN2*.

This allows to use *PIN2* as an unblocking password for *PIN1*. The complete list of options to change passwords is in the table Table 6.

Table 6. Authentication required for password change

Authentication required for password change	
Password to be changed	**Required authentication**
PIN1	current PIN1
	PIN2
	APW [1]
PIN2	current PIN2
	APW [1]
APW	current APW
	PPW [2]

[1)] only during *issue()* transition, see chapter 5.

[2)] only during *produce()* transition, see chapter 5.

8 Interfaces

GNT USB Token is delivered with SIPKCS provider [1] which implements the PKCS #11 standard developed by RSA Security Inc [2] as a main API. SIPKCS allows for easy plug in of Token into the existing PKCS #11 enabled applications. Compatibility of SIPKCS with the widely used PKCS #11 enabled applications is subject of continuous maintenance in the manufacturer labs. For .NET developers we provide .NET PKCS #11 adapter (written in C#).

For C/C++ developers we provide the straightforward API in ANSI C [3] that allows to exploit the complete set of the Token features. Bindings (Pascal, Python) for other programming languages are included. Sample code for all APIs with instantly compilable solutions in MSVS are part of the delivery.

The administration utility GINIT [4] that allows to perform transitions *Issue()* and *Unissue()*, to change and reset passwords, to query publicly available information and to manage configuration templates is part of the delivery.

Based on the SIPKCS we provide the complete solution for email security SMAIL [5] including support for creation and management of root certification authority and tree of user certificates.

For those dealing with classified information we provide SILOGON Credential Provider [6] that allows for multifactor logon to workstation using GNT USB Token in agreement with strong security policy settings[1].

9 Standards

Table 7 Conformance of cryptographic operations with standards

Operation	Standards
RSA key pair generation	PKCS #11 v2.20 (CKM_RSA_PKCS_KEY_PAIR_GEN)
RSA encrypt, decrypt	PKCS #1 v1.5 (RSAES-PKCS1-v1_5), X.509, ISO/IEC 9594-8 PKCS #11 v2.20 (CKM_RSA_PKCS, CKM_RSA_X_509)
RSA sign, verify	PKCS #1 v1.5 (RSASSA-PKCS1-v1_5) X.509, ISO/IEC 9594-8 PKCS #11 v2.20 (CKM_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_SHA256_RSA_PKCS, CKM_RSA_X_509)
SHA-1, SHA-256	FIPS 180-2 PKCS #11 v2.20 (CKM_SHA_1, CKM_SHA256)
RNG	FIPS 140-2
AES	FIPS 197 PKCS #11 v2.20 (CKM_AES_KEY_GEN, CKM_AES_ECB, CKM_AES_CBC_PAD) PKCS #11 v2.30 (CKM_AES_OFB, CKM_AES_CTR)
DES	FIPS 46-3 PKCS #11 v2.20 (CKM_DES_KEY_GEN, CKM_DES_ECB, CKM_DES_CBC_PAD, CKM_DES_OFB8)
TDES	FIPS 46-3 PKCS #11 v2.20 (CKM_DES3_KEY_GEN, CKM_DES3_ECB, CKM_DES3_CBC_PAD)
Padding for CBC mode	PKCS #7
ECB,CBC, OFB modes	FIPS 81
CTR mode	ATM Security Specification v1.1

1 MS CNG provider for GNT USB Token is currently under development and will be available in one of the next releases.

10 Applications

GNT USB Token is dedicated to applications with strong demands on data privacy and consistency and evidence of data originator. It is suitable for use as a personal identification and authentication facility in networks of mobile users. Example applications comprises the following:

- Multifactor authentication
- E-mail encryption and digital signature
- Software and hardware licensing
- Virtual private network
- Processing of classified information
- Intellectual property rights management

11 References

[1] SIPKCS - PKCS #11 provider for GNT USB Token, SoftIdea, s.r.o. , May 2011, http://www.softidea.sk/sipkcs_specification_en.pdf

[2] PKCS #11 v2.20: Cryptographic Token Interface Standard, RSA Laboratories, June 2004, http://www.rsasecurity.com

[3] GNTAPI - Application programming interface in ANSI C, SoftIdea, s.r.o. , January 2006, http://www.softidea.sk/gntapi_specification_en.pdf

[4] GINIT - User manual, SoftIdea, s.r.o. , May 2011, http://www.softidea.sk/ginit_manual_en.pdf

[5] SMAIL - Securing email communication using GNT USB Token, Datasheet (AN101010), SoftIdea, s.r.o. , June 2012, http://www.softidea.sk/an101010_en.pdf

[6] SILOGON - Multifactor workstation logon using GNT USB Token, Datasheet (AN232310), SoftIdea, s.r.o. , March 2012, http://www.softidea.sk/an232310_en.pdf

SoftIdea s.r.o.
Sliačska 2/D, 831 02 Bratislava
tel.: +421 2 444 60 444
fax.: +421 2 446 40 441 http://
www.softidea.sk
info@softidea.sk