Technical Reference

VMPC Encryption Module

By ITnnovative

Using STM32 technology

Contents

Introduction

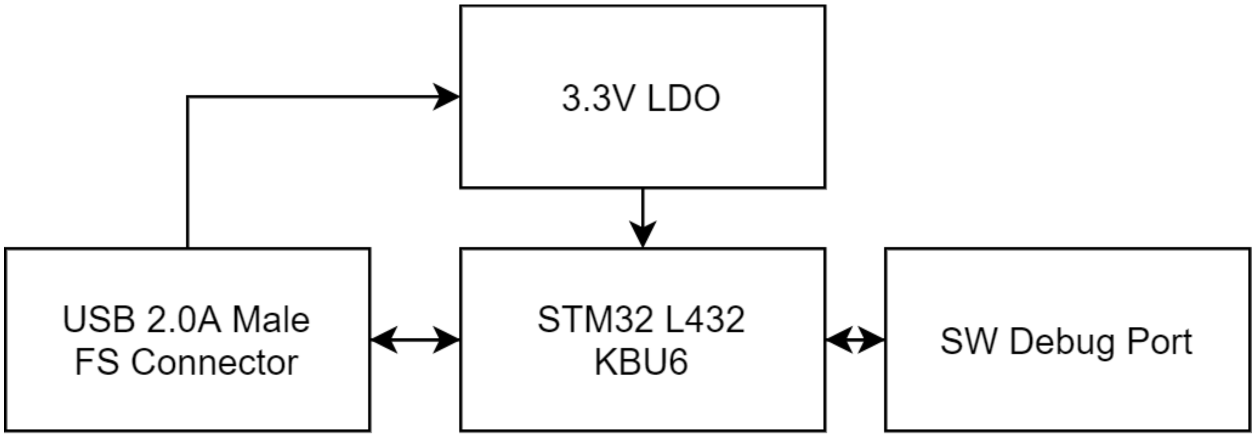
VMPC Encryption Module is a hardware implementation of VMPC cipher using STM32 microcontroller as processor. It's a co-processor for microcontroller units, single board computers or processors allowing to quickly serialize and deserialize stream of data.

It's based on VMPC cipher which is almost impossible to break and uses 512-bit passphrase and 512-bit initialization vector. Thanks to this time required to break algorithm is comparable to RSA-32768, when the small chip is able to serialize the data in real-time.

Max supported mode for this device is 2048-bit passphrase and 2048-bit IV, but IV size change requires firmware modifications.

Idea Schematic

Idea Schematic describes USB-based Encryption Module Peripheral for PC. It can work with theoretical speed of 650kB/s with 80MHz MCU, however communication interface takes a bit of its computation power and it lands around 200kB/s for USB.



Device is powered via USB and consumes less than 10mA during operation. This means that it can be powered from USB connector without any problem (it has current capability of 200mA).

Sometimes it has added SPI FLASH for on-device storage for ex. passwords (EMO-HPM).

Power

Device in SPI/UART revision is powered using 3.3V and GND pins. In peripheral revision it's powered from USB port using 3.3V regulators.

The hardware

The heart of the board is STM32-series microcontroller (usually STM32 F series).

Voltage regulator is mandatory to provide constant power to the device.

The device using STM32L432KBU6 processor running at max frequency reaches around 200kBps chunk serialization speed. This is due to USB limitations, using SPI it could be increased.

Casing

Casing is made out of PLA or ABS on 3D rep-rap printer eg. Prusa i3 MK3S. Beware that it should be printed using 0.1/0.15mm layer thickness (unless you love seeing ugly marks on sides of your casing, then you can go whatever you want).

STL files for casing are stored in same location as this document and are named as follow: **usb_case<part_num>_rev<rev_num>.stl.** Sample casing printed with 0.15mm layer height, 0.4mm nozzle at 90mm/s looks like this.

The commands

Device is communicating over SPI/UART or USB CDC, so here's table of specific commands.

Command	**Argument0**	**Argument1**	**Meaning**
0x00	None	None	Dummy command – do nothing (await next command)
0x04	Passphrase length (0-255)	Passphrase consisting of Argument0 bytes.	Sets passphrase in device.
0x05	0x0 – dummy byte	None	Initializes VMPC Cipher for new stream.
0x06 (very slow)	None	None	All further bytes are bytes that should be serialized (stream mode)

0x07	Data length (0-8192) MSB first, two bytes	None	Used for chunk serialization. Max chunk size is 8192 bytes.
0x08 – 0x4F			RESERVED
0x50	0x0 – dummy byte	None	Dump current serialization data.
0x51	Serialization data, described in separate section. (258 bytes)	None	Loads serialization data into module.
0x55-0xEF	None	None	FOR FURTHER USAGE
0xF0	0x0 – dummy byte	None	Get Supported Features (32B of data)
0xF1	<index>	0x0 – dummy byte	Get config value at index
0xF2	<index>	<New value>, see table in Config Management section	Set config value at index
0xF3 – 0xFF	None	None	FOR FURTHER USAGE

Chunk-mode encryption

While the device receives a command for chunk serialization it awaits chunk length. Max chunk length is 8192 bytes. **Default Chunk** is 64 bytes.

After sending length of the chunk to the device, you provide the chunk itself containing of specified bytes. Device returns serialized data as soon as it becomes serialized.

Sequence:

0x7 – enter chunk mode

[0x0000 – 0xFFFF] – chunk size (16-bit number)

[Bytes] – chunk bytes

Setting passphrase

To set a passphrase you need to send specific sequence:

0x4 – Set passphrase

[0x01 – 0xFF] – passphrase length

[Passphrase Bytes] – passphrase data – bytes of password in amount specified above

Rebuilding cipher

Cipher data needs to be serialized in stream way. So, every time you start serializing new data you need to reset the cipher to default state to make it able to deserialize it.

To do it so you need to send specific commands:

0x5 – initialize cipher command

0x0 – dummy byte

Entering stream encryption

You can enter chunk-mode encryption or stream encryption. The second one is recommended. To do so, you need to send **0x6** as command. The device will serialize all further bytes into encrypted data and return it to the receiving channel (either host or pass forward). All bytes will be considered as data, so no commands can be sent to the device.

Beware: this method is slow (it needs to send USB packet after every single byte). It is recommended to change default value for stream encryption to 64B to prevent overhead. It needs to be done inside device firmware and inside API. Warning: you need to send data in 64B and fill it to n*64 to get all data from the device.

Dumping and loading serialization data.

Serialization data can be dumped to support multi-user functionality (different users have different passwords). To do so, you need to send dump command. Device will respond with serialization data that consists of exactly 258 bytes.

First 256 bytes are cipher data and two next bytes are jump bytes, however this is specific for VMPC algorithm. You can also store all 258 bytes together.

To load the data, you need to provide load command and send bytes in same order to the device you've received them. This way you can save and load encryption session data on separate devices. It may be useful if you have multiple simultaneous users on your device and every user has own password.

Dump data:

0x50 - dump data command

0x0 – dummy byte

Response: 258 bytes of stream data **(order matters!)**

The returned data is:

A) P[] - permutation table of VMPC algorithm

B) s – s-value of VMCP algorithm

C) n – n-counter of VMPC algorithm

Load data:

0x51 – load data command

[DATA] – 258 bytes of dumped data **(order matters!)**

Sample encryption procedure

0x4 0x4 ‘t’ ‘e’ ‘s’ ‘t’ – set passphrase of 4 bytes “test”

0x5 0x0 – initialize cipher

0x7 0x0 0x4 – serialize sequence of 4 bytes

‘t’ ‘e’ ‘s’ ‘t’ - our data sequence

It will drop something like:

0xE4 0xCF 0x40 0x65 onto receive buffer. Those are 4 serialized bytes. Deserialization is same, excluding that instead of putting “test” in data sequence, you put serialized data. **Beware: bytes marked red will differ basing on IV of VMPC algorithm.**

For decryption it goes like: **0x4 0x4 ‘t’ ‘e’ ‘s’ ‘t’ 0x7 0x4 0x4 0xE4 0xCF 0x40 0x65.**

Listing device capabilities

Device capabilities can be listed by **0xF0 0x0** command. It returns 32B of data where n-th bit (LSB first, 0-255) describes if the command is supported on the device or not.

Config management

To change config variable use **0xF2 <index> <value>**. To get config variable use **0xF1 <index> 0x0**. Available config variables are listed below:

Index	Description
0x0	STREAM_CHUNK_SIZE – manage stream chunk size to define how large data needs to be processed before sending stream result. **Possible values: (1-8192), two bytes, MSB first**
0x1	ENABLE_FAST_USB – enables fast USB mode. Beware that in this mode the device responds with USB_READY (USB_IDLE) state after it receives data (not after it processes data which is default). This increases device speed, but may cause significant errors in encryption thus it’s not recommended. **Possible values: (0-1), one byte.** **Only with USB DRIVER 1.0, after upgrading USB Driver this feature has been replaced with data caching.**

Changelog

Version	Date	Changes	Affects Hardware?
1.0	03.11.2020	Initial Version	YES
1.1	05.11.2020	Increased device chunk serialization length to 2^32-1. Increased password length to max 255 bytes. Documentation update for new features.	YES
1.2	5.11.2020	Added serialization dump/load commands. Updated docs.	YES
1.3	9.11.2020	Added Idea Chart Added information about FLASH load/storage commands Added information about Random Password Generation Command	YES
1.4	28.11.2020	Improved Docs Added information about Flash Limitations	NO
1.5	3.12.2020	Improved docs, redacting, removed obsolete data	YES
1.5	7.01.2021	Removed FLASH from device, updated docs	NO
1.6	14.01.2021	Added chapter about casing. Improved docs.	NO
1.7	26.01.2021	Some additions for commands	YES
1.8	27.01.2021	Added 0xF1/0xF2 commands	YES
1.9	30.01.2021	Added new USB Driver data	YES

Firmware license