Innovative Secure Element Grids For Internet Of Secure Elements Servers

Pascal Urien
Telecom Paris
19 Place Marguerite Perey 91120 Palaiseau, France
Pascal.Urien@Telecom-Paris.fr

***Abstract*—This paper presents an innovative open physical and logical design for secure element grids. There is need for on-line services providing secure storage and tamper resistant computing resources. Secure Elements, typically under smartcard form factors, are widely used in bank cards or SIM modules The Internet Of Secure Elements (IOSE) project attempts to deploy on-line secure element grids, providing remote application uploads and TLS user interfaces. We introduce secure element processors (SEP) that perform bridge between ISO7816 and I²C protocol, and which support non blocking operations. We present a protocol enabling efficient support of secure element grid in multi tasks environment. Finally we present some experimental results.**

***Keywords*—** Secure Element, IOSE, Security, TLS

I. INTRODUCTION

There is a need for on-line services offering secure storage and tamper resistant computing resources. Secure storage means that cloud providers are not able to read or modify hosted data. Tamper resistant computing means that keys associated to cryptographic algorithms such as signature for blockchain transaction, can't be recovered at run time. In classical technologies, these operations are performed by *Hardware Secure Modules* (HSM), a piece of hardware whose security requirements are defined by the FIPS 140-2 standard ("*Security Requirements For Cryptographic Modules*"). Secure Elements, typically with smartcard [5] form factor, are widely used in bank cards or SIM modules. The security level of these chips ranges from EAL5+ to EAL6+ according to common criteria standards (the maximum level being EAL7+). Furthermore, many of them embed a JAVA virtual machine (JVM), and are able to execute applications written in javacard [4], a subset of the java language. The *Internet Of Secure Elements* project (IOSE [1][11]) attempts to define servers that manage grids of secure elements. In this paper we present physical and logical design of secure elements grids, working with a Raspberry Pi through I²C bus [9], as illustrated by figure 1.

This paper is organized according to the following outline. Section 2 introduces *Internet Of Secure Elements* (IOSE) server based on secure element grids. Section 3 presents *Secure Element Processors* (SEP) used as basic bricks for IOSE servers. Section 4 describes I²C protocol dedicated to SE grids. Section 5 presents some experimental results. Finally section 6 concludes this paper.

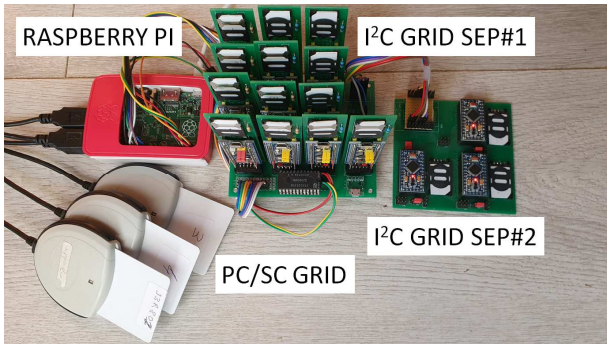

Fig. 1. Internet Of Secure Element (IOSE) server based on secure element grids, with raspberry pi host.

II. INTERNET OF SECURE ELEMENTS SERVER - IOSE

IOSE server [1] manages two planes (see figure 2), administration plane (over RACS protocol) used to download applications, and service plane (working over TLS-PSK), which provides tamper resistant computing resources. An open implementation [11] is available for Win32, Linux, and Raspberry Pi environments.

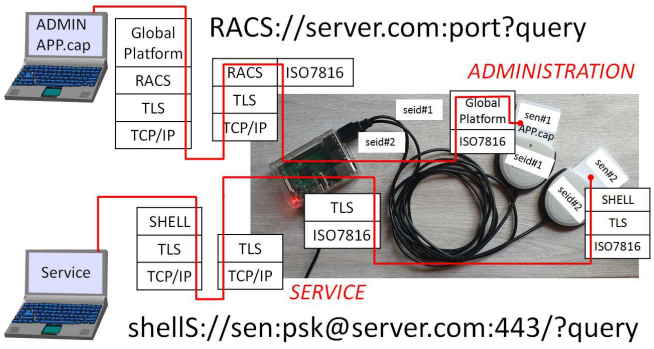

Fig. 2. IOSE server combines a RACS server and a TLS front server.

978-1-6654-6975-3/22/$31.00 ©2022 IEEE

A. *Remote APDU Call Secure - RACS*

The open *Remote APDU Call Secure* (RACS) protocol [6] was designed to secure SE servers with TLS. Client and server are mutually authenticated by X509 certificates. SE identifier within server is named *Secure Element IDentifier* (SEID); it maybe deduced from a slot number or from a smartcard reader serial number. The security policy is based on the client certificate Common Name (CN), which is associated to a set of SEIDs. RACS manages the competition for SE use. A TCP session is identified by an integer (*Session IDentifier*, SID), a SE can be used (i.e. powered up) only by one SID. Thanks to *Global Platform* (GP) protocols, it is possible to download applications in secure elements, and thereafter to remotely interact with them through ISO7816 commands.

B. *TLS Pre Shared Key - TLS-PSK*

The main idea of IOSE [1][2] is to deploy TLS-PSK (Pre-Shared-Key) servers in secure elements; an embedded shell processes messages received over TLS session. To reach this goal IOSE server runs a TLS daemon (TLS front server), which routes packets to/from TLS backend servers (see figure 2) hosted within secure elements. In a way similar to internet servers, backend servers are identified by their *Server Name* (that we call *Secure Element Server Name*, SEN). The SEN attribute is inserted in the *Answer To Reset* (ATR) ISO7816 message, returned upon physical reset. According to TLS1.3, TLS-PSK works with a pre-shared secret (256 bits) known by secure element and its owner.

III. SECURE ELEMENT PROCESSOR (SEP)

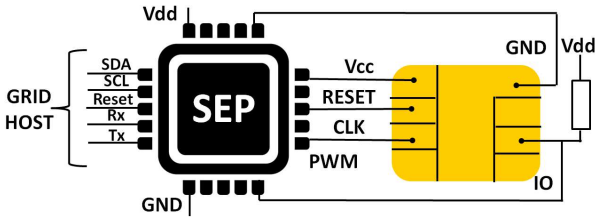

Fig. 3. Secure Element Ptocessor (SEP) realizes a bridge between ISO7816 and I²C or UART protocols

Many computing platforms such as Windows, Linux, or Raspberry Pi, support PC/SC (short for "*Personal Computer/Smart Card*") frameworks and their associated smartcard readers. Although it is an efficient way to develop and test IOSE software (as illustrated by figure 2), this kind of platform is not scalable in terms of costs (due to smartcard reader price) and form factor (USB controllers are limited to 127 devices, and discovery process or power supply may fail when too much devices are plugged). Our approach is to use a micro-controller (SEP, see figure 3) for every secure element. SEP realizes ISO7816 [3] communication, and provides serial or bus interface (such as I²C [9]) with the host, to which is connected the SE grid. Secure element operating systems are non-preemptive [8], i.e. ISO7816 requests are blocking, and no communication is possible before response. So the SEP provides facilities (UART, I²C bus…) needed by multi tasks processing environment.

A. *ISO7816-3 communication protocols*

Secure elements exchange small messages (named *Application Protocol Data Unit* - APDU) whose size is less than 260 bytes. A request comprises a five bytes header (CLA INS P1 P2 P3) and optional data (up to 255 bytes). A response comprises optional data (up to 256 bytes) and two status bytes (SW1, SW2).

APDUs are transported by TPDUs (*Transport Protocol Data Unit*). Two main serial transmission protocols are available: T=0 i.e. byte stream, and T=1 that works with frames including a three bytes header (NAD, PCB, LEN) and one or two bytes trailer (checksum or CRC). Contrary to traditional UART, there is a single IO pin (see figure 3).

By default, according to ISO7816, the clock frequency F_{CK} (applied on the CLK pin, see figure 3) is divided by D/F factor with D=1 and F=372. The SEP provides the SE clock thanks to PWM (*Pulse-width Modulation*) timer facilities. For example a 4MHz clock provides a bit time (named ETU by ISO7816 standard) of 93µs. For T=0, a character comprises at least 12 bits: a start bit, a byte (8 bits), a parity bit, a notification bit of parity error, and a stop bit. For T=1, a character comprises at least 11 bits: a start bit, a byte (8 bits), a parity bit and a stop bit.

The SEP implements software ISO7816 UART that writes or reads characters. This library [7] is written in C++ and is compatible with Arduino IDE. It works with AVR ATmega328 (Arduino Mini Pro 8MHz), and STM32F103C8T6 (alias blue pill, 72MHz). In our experiments we notice a limit of about 372x4 processor cycles for T=x protocols processing. As an illustration SEP with 8MHz clock handles ETU of 186µs ($372/F_{CK}$, with F_{CK} = 2MHz), and SEP with 72MHz clock handles ETU of 31µs ($372/2/F_{CK}$, with F_{CK} = 6MHz).

B. *ISO7816 power-up*

According to ISO7816 reset is kept low, VCC is set high, and clock is started; when reset goes high, the SE returns a set of bytes called *Answer To Reset* (ATR). This message includes information about available transmission protocols and throughput (for example. maximum F and D parameters), and contains a programmable set of bytes (named *Historical Bytes*) used in IOSE context to read the SEN value (i.e. the secure element name (SEN) needed by TLS embedded server).

The parameters F and D can be modified according to a procedure named *Protocol Type Selection* (PTS). A proposal of new protocol parameters is sent to SE, which is echoed in case of success. The PTS request comprises the following elements: first byte is FF; second byte MSB nibble number of

bytes (1), LSB nibble protocol (T=0); third byte (TA) MSB nibble encoded F value (9), LSB nibble encoded D value (5), last byte checksum (i.e. exor of previous bytes).

C. Examples of SEP boards

The first SEP board (SEP#1, see figure 4) is built over STM32F103C8T6 microcontroller, with 72 MHz clock, including 64KB FLASH and 20KB RAM. The *Secure Element* frequency is 6 MHz, with F=372 and D=2; so the ETU is 31µs. The maximum SE throughput for T=0 protocol is about 2,7 KB/s. UART and I²C interfaces are available through a 10 pins connector.

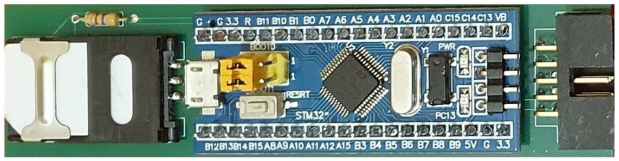

Fig. 4. SEP board (SEP#1) built over a STM32 micro-controller, provinding I²C and UART interfaces

The second SEP board (SEP#2) is built over ATmega328 microcontroller, with 8MHz clock, including 32KB FLASH and 2KB SRAM. The secure element frequency is 2 MHz, with F=372 and D=1; so the ETU is 186µs. The maximum SE throughput for T=0 protocol is about 0,45 KB/s. Figure 5 shows a grid of these SEP boards connected a raspberry pi thanks to I²C bus.

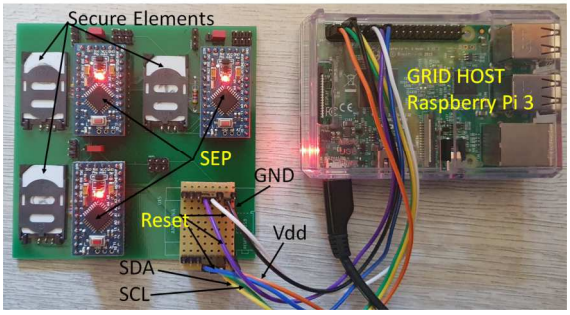

Fig. 5. An IOSE server based on Raspberry Pi and ATmega328 micro-controler (Arduino Mini Pro 8 MHz)

IV. I²C Protcol for Secure Element Grids

The Inter-Integrated Circuit (I²C) protocol is a bus interface designed by Philips Semiconductor in 1982. It works with two lines pulled high by resistors (Rp). Serial Clock (SCL) carries the clock signal. Serial Data (SDA) transfers data. The master generates the clock (whose common values range between 100 KHz to 400 KHz). Slaves are identified by addresses, whose size is typically seven bits. Both master and slaves can pull SDA and SCL lines to ground. When SCL is high SDA must remain stable (i.e. bits are sampled at SCL rising time). Two exceptions, handled by master, define begin (START) and end (STOP) transmission events; START: SCL high and SDA falling edge; STOP SCL high and SDA rising edge. I²C frames also include R/W bit (SDA low for read) in order to tag read or write operations, and acknowledgement bit (ACK) either positive (SDA low) or negative (SDA high).

A. I²C Write Operation

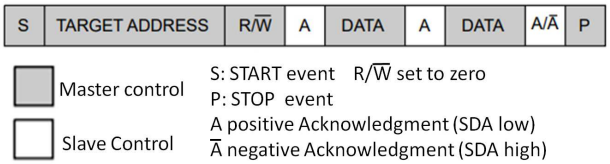

Fig. 6. Write operation in I²C bus [10]

WRITE operation (see figure 6) begins with START event, follows by the slave address (7 bits) and R/W bit set to zero. The slave delivers a positive ACK. Thereafter master transmits bytes which are acknowledged by slave. The transmission ends with a slave negative acknowledgment, follows by a STOP event.

B. I²C Read Operation

READ operation (see figure 7) begins with START event, follows by the slave address (7 bits) and R/W bit set to one. The slave delivers a positive ACK. Thereafter slave transmits bytes, which are acknowledged by master. The transmission ends with a slave negative acknowledgment follows by a STOP event.

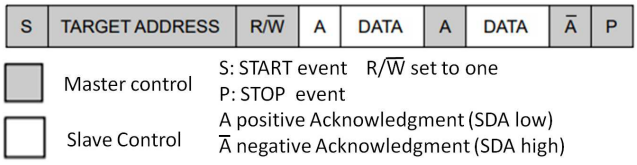

Fig. 7. Read operation in I²C bus [10]

V. Designing I²C protocol for Secure Elements Grid

Fig. 8. A secure element grid, with 4x4 SEP slots. The estimated bus capacitance is 250 pF (450ns rise time for 1,8 kΩ I²C Rp resistor)

Our goal is to design a server able to manage N_{SE} simultaneous accesses to N_{SE} secure elements. We estimate the needed throughput of about 0,5KB/s per SE, what leads to $N_{SE}/2$ KB/s. The grid illustrated by figure 8 supports a 200 KHz clock (about 20 KB/s)

A. *Message over I²C bus*

A message over I²C bus is encoded according to the following format:

NAD PCB LEN PAYLOAD CRC, in which

- NAD: 5A for writing, and A5 or 00 (no data) for reading

- PCB values are detailed by figure 9 and figure 10

- LEN is the length of the SEP payload

- PAYLOAD is the information sent or received to/from SEP

- CRC is a sixteen bits CRC (as defined by ISO/lEC 3309)

CF (LEN=0)	C0	C3 (LEN=0)	C1	83 (LEN=0)
Power-Up Secure Element Return historical bytes	Echo LEN bytes	Power-Down Secure Element	Send ISO7816 request LEN bytes	Error CRC (master side) Resend message

Fig. 9. Encoding of PCB byte for wrtting operations

CF	C0	C3 (LEN=0)	C1	82 (LEN=0)	81 (LEN=0)
Power Up LEN historical bytes	Echo LEN bytes	SE Power Down Success	ISO7816 response LEN bytes	Error CRC (SEP side)	SEP Error

Fig. 10. Encoding of PCB byte for reading operations

Although messages end by CRC, CRC errors should never happen. It is rather a metric for hardware quality. Nevertheless a single retransmission is tried when such event is detected.

B. *Writing operations*

For writing operations the master performs bursts, whose maximum size is dependant of the slave receiving buffer. The default value is 32 bytes in Arduino IDE, but it may be increased. In Linux *System Management Bus* (SMBus) based on I²C protocol, reads and writes 32 bytes blocks. So we choose to support 32 bytes bursts. What means that writing (and reading) procedures perform segmentation in 32 bytes blocks, each of them being preceded by the slave address (according to section IV-A)

C. *Reading operations*

A writing procedure sends a request to SEP, which is afterwards processed by secure element (i.e. I²C PAYLOAD transports APDU request). Depending on the command, the computing time could take a few hundred milliseconds. We need to define a non blocking procedure for reading operation, because software manages the competition for I²C bus access (by semaphore techniques). The principle is that SEP will always respond as fast as possible to I²C reading request. A dedicated SEP interruption returns a null byte when no response is available from secure element. After a writing operation, master periodically (about every 10ms) polls a slave SEP, which returns either NAD=0 (no data) or NAD=5A (data ready). Upon success (data ready) it performs two single readings in order to collect PCB and LEN values. Payload (if any) and CRC are read (according to VI B) thanks to bursts whose maximum size is 32 bytes.

VI. Testing Performances

A performance test establishes non-stop TLS-PSK sessions, one per SEP. For each session an ECDSA signature is computed by secure element; this is an emulation of signature service for blockchain transactions. The results are illustrated by figure 11; PC/SC readers and SEPs are equipped with identical javacards. We are currently working on performance evaluation/modelization for I²C grid equipped with SEP#1 device.

number	PC/SC grid	I²C grid SEP#1	I²C grid SEP#2
1	2038 ms	2390 ms	4580 ms
2	2044 ms		
3	2036 ms	2460 ms	4598 ms
6		2516 ms	

Fig. 11. Average of TLS session duration for 100 occurences, with one, two, three or six secure elements. Time is expressed in milliseconds

VII. Conclusion

In this paper we presented on open design for secure elements grids dedicated to IOSE server using I²C bus. We are looking for grids supporting 32 secure elements. Given the cost of raspberry pi, we expect a low price per on-line secure element.

Reference

[1] IETF Draft "Internet of Secure Elements", draft-urien-coinrg-iose-04.txt,IETF Draft, October 2021

[2] Urien, P., Demonstrating Internet Of Secure Elements Server, IEEE CCNC2022, 8-11 January 2002

[3] ISO7816, "Cards Identification - Integrated Circuit Cards with Contacts", The International Organization for Standardization (ISO).

[4] Chen, Z., "Java Card™ Technology for Smart Cards, Architecture and Programmer's Guide", ADDISON-WESLEY, 2000

[5] Jurgensen T.M., Guthery, S.B., "Smart Cards: The Developer's Toolkit", O'Reilly, 200

[6] P. Urien, "RACS: Remote APDU call secure creating trust for the internet," 2015 International Conference on Collaboration Technologies and Systems (CTS), 2015, pp. 351-357, doi: 10.1109/CTS.2015.7210448.

[7] P. Urien, "An Innovative Four-Quarter IoT Secure Architecture Based on Secure Element," 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018, pp. 1074-1080, doi: 10.1109/IWCMC.2018.8450435.

[8] Bouzefrane, S. Samia, Paradinas, P. "Les Cartes à Puce,", Paris : Hermès Science : Lavoisier, 2013, ISBN: 9782746239135 2746239132.

[9] NXP Semiconductors, I²C-bus specification and user manual, UM10204, Rev. 7.0 October 2021

[10] Texas Instrument, "Understanding the I²C Bus", Application Report SLVA704, June 2015

[11] Urien, P., "The IOSE project", https://github.com/purien/IoSE