WCPP V0.1 PROTOCOL DEFINITION

[TOP SECRET]

Acronyms

ACRONYM	DEFINITION
AD	Authentication Data
RPI	Raspberry Pi
WCPP	WhyCard Proprietary Protocol
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
MB	Message Begin [1]
ME	Message End [1]
CF	Chunk Flag [1]
SR	Short Record [1]
IL	ID Length [1]
TNF	Type Name Format [1]
TLV	Type/Length/Value [1]
WCPPVB	WCPP Version Block
RP	Received Packet
IMEI	International Mobile Equipment Identity
MAC	Medium Access Control Address
RFU	Reserved for Future Use
DA	Does not apply
UTC	Coordinated Universal Time
I^2C	Inter-Integrated Circuit
DB	Database

WCPP Layer Definitions

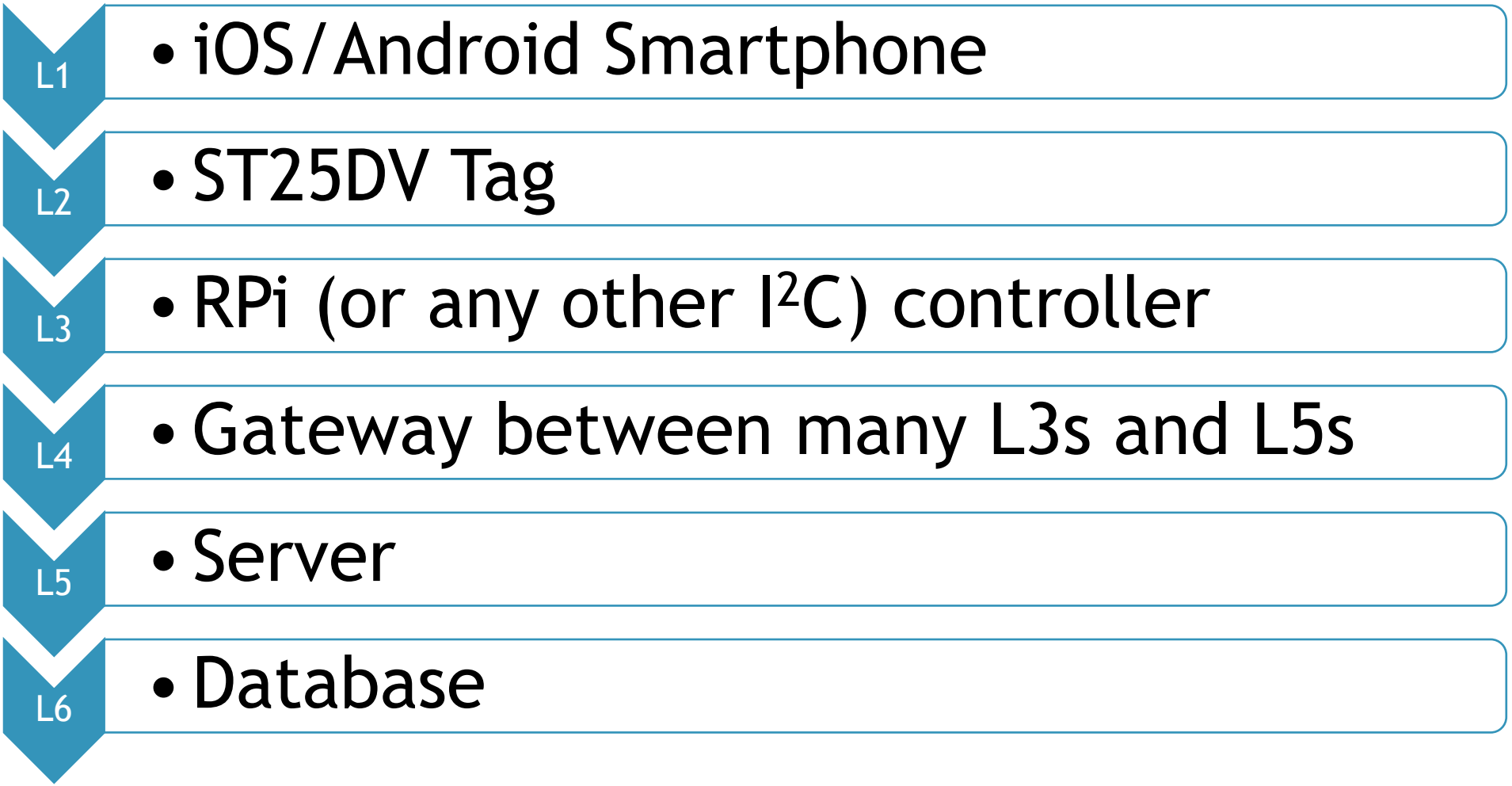

Figure 1

WCPP

WCPP (WhyCard Proprietary Protocol) is our protocol to communicate between Layers in one direction (from L1 to L5). First two bytes of our protocol are always two bytes defining version – major and minor.

Interaction Rules

- If major version of RP is different from Layer's current – the RP is rejected due to incompatibility.
- If minor version of RP is lower from Layer's current – the RP is processed with warning.
- If minor version of RP is higher from Layer's current – the RP is rejected due to incompatibility.
- If the versions of RP are equal to Layer's current – the RP is processed normally.

Software Rules

- The software handling RP's should be able to process all known minor versions before it's current.
- The software handling RP's should notify the owner in case it found a newer major version, as staying on deprecated major version could cause security issues.
- Minor versions are expected only to introduce newer features, but not restricted to.

Visual Description

<table>
<tr><th colspan="3">WCPP</th></tr>
<tr><th colspan="2">WCPPVB</th><th rowspan="2">Payload</th></tr>
<tr><th>Major version</th><th>Minor version</th></tr>
<tr><td>1 byte</td><td>1 byte</td><td>Unrestricted</td></tr>
</table>

Figure 2

WCPP V0.1

WCPP V0.1 is our proprietary protocol to communicate in way L1 => L5.

Restrictions

- All WCPP V0.1 message should have no more than 65534 bytes of data.
- As a result Payload lengths should be no more than 65451 bytes of data.

Visual Description

<table>
<tr><th colspan="8">WCPP V0.1</th></tr>
<tr><th colspan="2">WCPPVB</th><th colspan="2">HEADER</th><th>DATA</th><th colspan="3">SIGNATURE</th></tr>
<tr><th colspan="2">Version block</th><th>Request block</th><th>Checksum block</th><th>Payload block</th><th colspan="3">ID Block</th></tr>
<tr><th>Major version</th><th>Minor version</th><th>Request Type</th><th>SHA-256 checksum</th><th>Payload</th><th>L1 ID</th><th>L3 ID</th><th>L4 ID</th></tr>
<tr><td>1 byte</td><td>1 byte</td><td>1 byte</td><td>32 bytes</td><td>Unrestricted</td><td>16 bytes</td><td>16 bytes or
0 bytes (a)</td><td>16 bytes or
0 bytes (b)</td></tr>
<tr><td>= 0</td><td>= 1</td><td>Request Type</td><td>Payload checksum</td><td>Payload, according to Request Type</td><td>Unique ID of L1 dev.</td><td>Unique, in terms of L4, ID of L3 dev.</td><td>Unique ID of L4 dev.</td></tr>
<tr><td colspan="8">(a) applies on transfers L3 => L5
(b) applies on transfers L4 => L5</td></tr>
</table>

Figure 3

NFC Encapsulation

NFC uses NDEF TVL [2] and NDEF format with TNF(5) = Unknown Type Record [1] in order to store data as an amount of bytes. In NDEF we combine everything in only one NDEF Message.

Restrictions

- WCPP V0.1 should be no more than 65534 bytes.

Visual Description

<table>
<tr><th colspan="11">NFC Encapsulation</th></tr>
<tr><th colspan="10">TLV Message 1</th><th>TLV Message 2</th></tr>
<tr><th rowspan="2">T</th><th colspan="2">L</th><th colspan="7">V</th><th>Terminator</th></tr>
<tr><th rowspan="2">.1</th><th rowspan="2">.2</th><th colspan="6">NDEF HEADER</th><th rowspan="2">Type Length</th><th rowspan="2">Payload length</th><th rowspan="2">Payload</th><th rowspan="2"></th></tr>
<tr><th>MB</th><th>ME</th><th>CF</th><th>SR</th><th>IL</th><th>TNF</th></tr>
<tr><td>1 byte</td><td>1 byte</td><td>0 bytes (b)
Or
2 bytes</td><td>1 bit</td><td>1 bit</td><td>1 bit</td><td>1 bit</td><td>1 bit</td><td>3 bits</td><td>1 byte</td><td>4 bytes</td><td>(a) bytes</td><td>1 byte</td></tr>
<tr><td>= 0x03</td><td>0x00..0xFE (b)
Or
0xFF</td><td>Absent (b)
Or
0x00FF..0xFFFE</td><td>= 1</td><td>= 1</td><td>= 0</td><td>= 0</td><td>= 0</td><td>= 5</td><td>= 0</td><td>= (a)</td><td>WCPP V0.1</td><td>= 0xFE</td></tr>
<tr><td colspan="11">(a) bytes = number of bytes used in order to encapsulate WCPP V0.1 Payload
(b) applies when WCPP V0.1 length is less then 255 bytes</td></tr>
</table>

Figure 4

WCPP Type 0

Definitions and Constraints

Usage

- Uses all layers from L1 to L6 in order to process query.
- L5 is responsible for authentication

Short idea

- At initialization, L1 receives AD directly from L5, after giving credentials. AD is unique token, that could be used in order to authenticate user during some period of time. AD should be attached to device. User can have only one active AD at some point of time.
- During authentication, L1 sends to L2 AD encapsulated with WCPP Type 0.1 via NDEF. L2 transfers data to L3. L3 accepts the NDEF, unwraps it and does checksum (SHA-256) check. Adds technical fiels, converting it into WCPP Type 0.2. Then transfers to L4. L4 accepts it, adds technical fields, converting it into WCPP Type 0.3, and then transfers it to L5.
- L5 accepts the packet, unwraps the AD from package, does security checks based on all received data, possibly connects with L6 in order to authenticate the user and after that, the authorisation data is transferred from L5 to L3 and the action is done.

Security Notice

- L1 stores data in secure way, but the critical data could be stolen from L1 using Root-rights/Jailbreak.
- L2 could be mocked, and data transferring from L1 could be stolen, exposing WCPP Type 0 internals.
- WCPP Type 0 could be easily mocked having only critical data from L1.
- WCPP Type 0 could be easily exposed by decompiling L1 Applications.
- L4 -> L5 should have valid SSL certificate.
- L6 should be inaccessible outiside (L5, L6) network.

Conclusion: *Insecure*. WCPP Type 0 shouldn't be allowed on devices with Root-rights/Jailbreak. WCPP Type 0 is recommended to use only in debug-related cases.

Visual Description

Note: showing only data encapsulated in WCPP V0.1 Payload field

WCPP V0.1 Type 0 Payload				
Timestamp	AD	Additional L1 Data		
		MAC-address	IMEI (a)	RFU
8 bytes	128 bytes	6 bytes	8 bytes	2 bytes
= Current UTC Timestamp	= AD from L1 internals	= NFC MAC-address of L1	= IMEI of L1 or 0 if DA	= 0x00
(a) in case of two or more IMEI's per L1, using the first one				

Figure 5

WCPP V0.1 Type 0 Layers Cycle

Step 1

Figure 6

Step 2

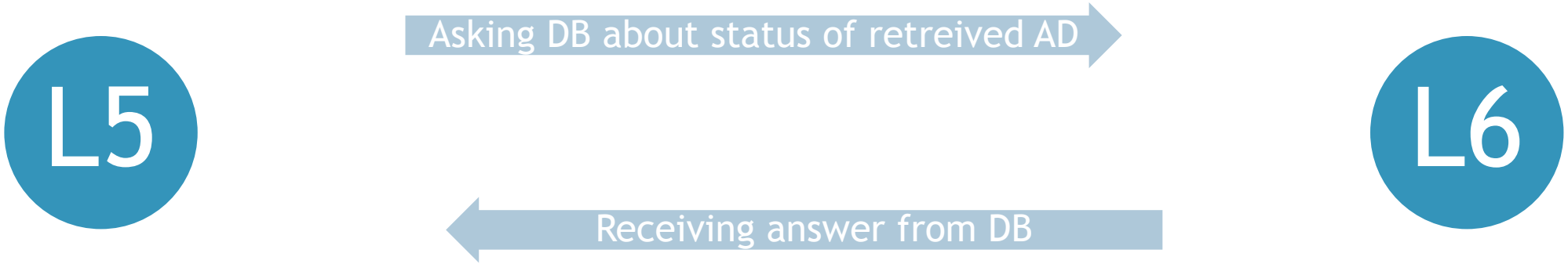

Figure 7

Step 3

Figure 8

References

1. W3C Community Group NFC, p. 4.2
2. STMicroelectronics, AN 3408, p. 2.3