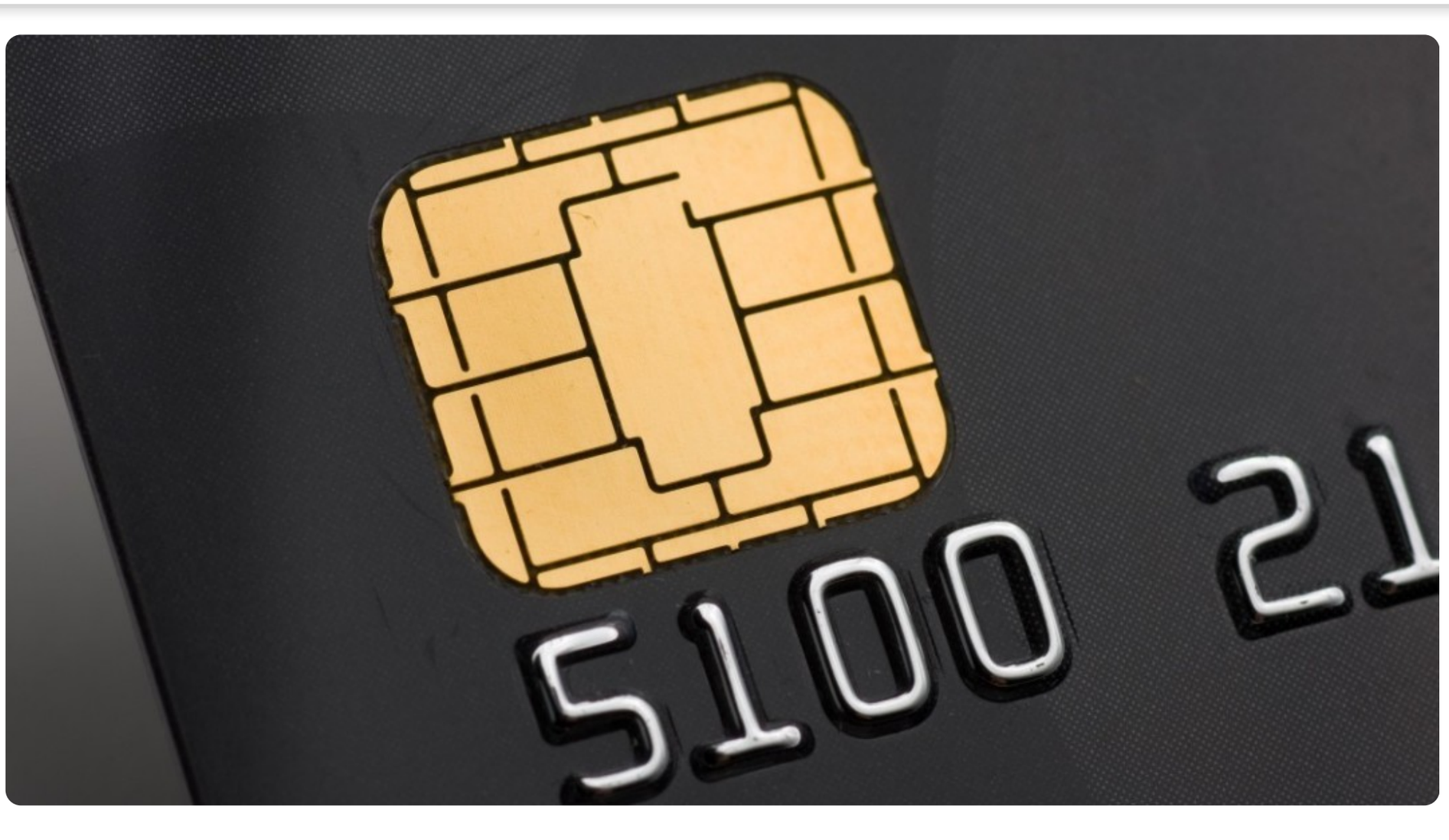


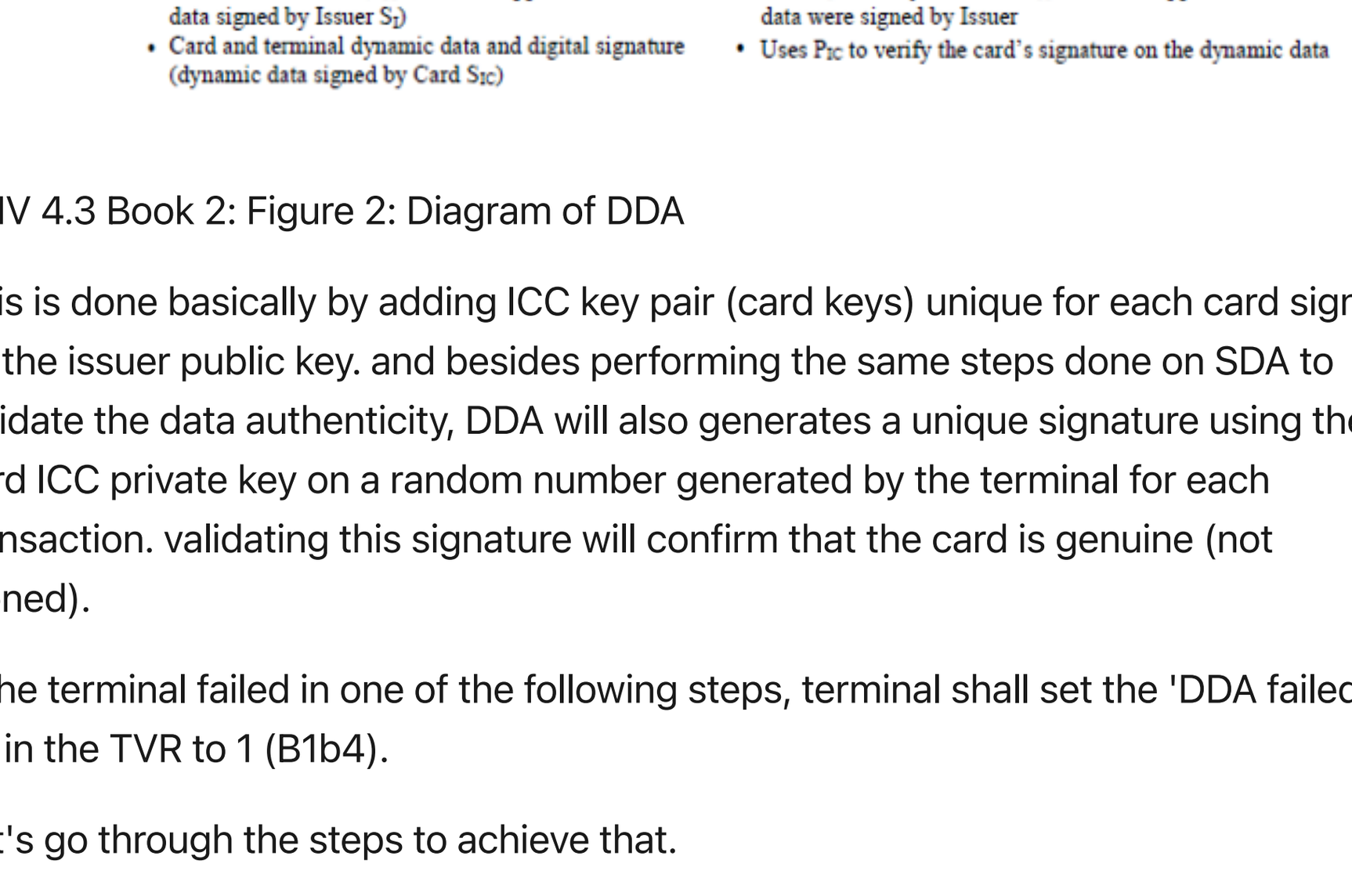
EMV Application Specification :: Offline Data Authentication (ODA) - part II

Ahmed Hemdan Farghaly
Senior Manager | Fintech | EFTPOS | mPOS | ISO8583 | EMV | NFC | QR | PCI-DSS | PA-DSS
+ Folgen
Veröffentlicht: 11. März 2022

Dynamic Data Authentication (DDA)

Please refer to the previous article about Static Data Authentication (SDA) here

DDA is a stronger authentication method where it is not only used to confirm the legitimacy of critical ICC-resident static data (same as SDA), it also validates the card itself and protect against card cloning.



EMV 4.3 Book 2: Figure 2: Diagram of DDA

This is done basically by adding ICC key pair (card keys) unique for each card signed by the issuer public key. and besides performing the same steps done on SDA to validate the data authenticity, DDA will also generates a unique signature using the card ICC private key on a random number generated by the terminal for each transaction. validating this signature will confirm that the card is genuine (not cloned).

If the terminal failed in one of the following steps, terminal shall set the 'DDA failed' bit in the TVR to 1 (B1b4).

Let's go through the steps to achieve that.

Here are the steps to perform a proper DDA. In our example, we'll use RID=A000000003 and Index=95 for VISA test card.

The card::

- Issuer generates key pair (issuer public and private key).
- The network [i.e. VISA] signs this issuer public key and generates a certificate (Issuer PK Certificate).
- Store the Issuer PK Certificate on the card [tag 90].
- Store the public key index for the key pair that was used by the network to sign the issuer public key. (i.e. PKI 95 for visa test card) [tag 8F].
- Issuer generates a key pair for the ICC itself called [ICC public & private key].
- Issuer creates another certificate containing ICC public key and a signature on important card data [signed static app data] which is signed by Issuer private key [tag 9F46].

The terminal::

- Terminal retrieves the PKI stored on the card [tag 8F].

8F Certification Authority Public Key Index "95"

- Terminal loads the CA public key for that index (terminal should have a way of storing and loading all supported CAPKs for all the supported RIDs).

Key Modulus: "BE9E1FA5E9A803852999C4AB432DB28600DCD9DAB76DFAAA47355A0FE37B1508AC6BF38860D3C6C2E5B12A3CAAF2A7005A7241EBAA7771112C74CF9A0634652FBCA0E5980C54A64761EA101A114E0F0B5572ADD57D010B7C9C887E104CA4EE1272DA66D997B9A90B5A6D624AB6C57E73C8F919000EB5F684898EF8C3DBEFB330C62660BED88EA78E909AFF05F6DA627B" *Exponent:* "03"

- Terminal retrieves Issuer Public Key Certificate [tag 90], Issuer Public Key Reminder if any [tag 92], and Issuer Public Key Exponent [tag 9F32]

90 IssuerPKCert "8B3901F6253048A8B2CB08974A4245D90E1F0C4A2A69BCA469615A71DB21EE7B3AA94200CFAEDCD6F0A7D9AD0BF79213B6A418D7A49D234E5C9715C9140D87940F2E04D6971F4A204C927A455D4F8FC0D6402A79A1CE05AA3A526867329853F5AC2FEB3C6F59FF6C453A7245E39D73451461725795ED73097099963B82EBF7203C1F78A529140C182DBBE6B42AE00C02"

92 Issuer Public Key Remainder "33F5E4447D4A32E5936E5A1339329BB4E8DD8BF0044CE4428E24D0866FAEFD2348809D71"

9F32 Issuer Public Key Exponent "03"

- Terminal decrypts the issuer public key certificate using CA public key.

Decrypted IssuerPKCert: "6A02476173FF12150040540101901A687AF619B88CBAD371903C89579B5890D605F905B093C1F856801AE33C12E65D02B64454D9921468283ED397835909BCBB2F659460833BAAC1C75343FF671EB93F04953C6AEF428F07EE28FC9ABFB65CF6A961B4A085AF297CD1453CF4719868883D20A8F624E45920BA3C98C5453DBF74927FD240C07C4262F736E460BB5FABC"

Refer to EMV 4.3 Book 2 Table 6 for the format of data recovered from issuer public key certificate.

- Validate the recovered data against that format to make sure it is correct.
- Extract Issuer Public Key (append Issuer Public Key Reminder if any).

From the above IssuerPKCert, we can extract the Issuer Public Key (or part of it) and append Issuer Public Key Reminder if any.

Here we have only part of it: "A687AF619B88CBAD371903C89579B5890D605F905B093C1F856801AE33C12E65D02B64454D9921468283ED397835909BCBB2F659460833BAAC1C75343FF671EB93F04953C6AEF428F07EE28FC9ABFB65CF6A961B4A085AF297CD1453CF4719868883D20A8F624E45920BA3C9"

After appending the Issuer Public Key Reminder.

Full IssuerPK: "A687AF619B88CBAD371903C89579B5890D605F905B093C1F856801AE33C12E65D02B64454D9921468283ED397835909BCBB2F659460833BAAC1C75343FF671EB93F04953C6AEF428F07EE28FC9ABFB65CF6A961B4A085AF297CD1453CF4719868883D20A8F624E45920BA3C933F5E4447D4A32E5936E5A1339329BB4E8DD8BF0044CE4428E24D0866FAEFD2348809D71"

- Terminal retrieves (ICC) Public Key Certificate [tag 9F46] and (ICC) Public Key Remainder [tag 9F48] if any.

9F46 Integrated Circuit Card (ICC) Public Key Certificate: "868A4EBE29CC8906810F90F45B7C2DCA73D8C63C8AB58E2D449F2DABF621DF21BA997C383DFCAA75E164A70F654503943B2DE5CBF090B91A0B3093036DFA74FA0C2BB968928F65ECEB01D8BF38FADC342AE994C3A5677FC5AD3A7941DC5F715922E35712E66C5810BF2F98694A70BB9A4C20CAB512CFE8D1FF8474F28863C79C19AEE14D4E104C4626B962BB07D1EE15"

9F48 Integrated Circuit Card (ICC) Public Key Remainder: "FBDADA20082FD6D0439BC9085D12F4F906AF8DA660DC8A9AA5A6B4B5922992D76506160ECB3F9B5327C5"

- Terminal decrypt ICC public key certificate using the issuer public key.

Decrypted ICCPK Cert:: "6A0447617390010100036FFFF121500002201019001C3B66C72EA96D3FFC19392475093CAD9BC0E5646479020D45FBBB9DD05115B13E93FF3BEAA12DB70A1DF86DA06DF0B00F41B2D30EDB56A5D8BD12325225141E70A618E3AE8EBFD340ADD689B27E5FF1F64AD2941A631D591B703455CA0A086F2C4E742DDEDD293BE13CA8718DCF65F71891FBF25D1FE319733BCBC"

- Refer to emv book 2 table 14 for the format of data recovered from ICC public key certificate.
- Validate the recovered data against that format to make sure it is correct.
- Extract ICC public key [append ICC public key reminder if any] [issuer trust this ICC card key]

From the above ICCPK Cert we can extract the issuer public key (or part of it)(based on EMV Book 2 table 14)

Here we have only part of it: "C3B66C72EA96D3FFC19392475093CAD9BC0E5646479020D45FBBB9DD05115B13E93FF3BEAA12DB70A1DF86DA06DF0B00F41B2D30EDB56A5D8BD12325225141E70A618E3AE8EBFD340ADD689B27E5FF1F64AD2941A631D591B703455CA0A086F2C4E742DDEDD2"

For the rest, we'll read it from ICC) Public Key Remainder tag 9F48: "FBDADA20082FD6D0439BC9085D12F4F906AF8DA660DC8A9AA5A6B4B5922992D76506160ECB3F9B5327C5"

Full ICCPK (card key):: "C3B66C72EA96D3FFC19392475093CAD9BC0E5646479020D45FBBB9DD05115B13E93FF3BEAA12DB70A1DF86DA06DF0B00F41B2D30EDB56A5D8BD12325225141E70A618E3AE8EBFD340ADD689B27E5FF1F64AD2941A631D591B703455CA0A086F2C4E742DDEDD2FBDADA20082FD6D0439BC9085D12F4F906AF8DA660DC8A9AA5A6B4B5922992D76506160ECB3F9B5327C5"

- Extract the signature from ICC public key certificate.

ICCPK hash buffer1:: "93BE13CA8718DCF65F71891FBF25D1FE319733BC"

- Using the SHA-1 algorithm, terminal recreate the signature that we extracted from ICC public key certificate by retrieving static data indicated by the card to be signed [tag 9F4A].
- Build a block of data following emv book 2 table 11.
- Compute a signature for this block of data [using SHA-1].
- Compare the 2 signatures and if they are equal, that means that data is authentic [terminal trust card data].

ICCPK hash buffer2:: "93BE13CA8718DCF65F71891FBF25D1FE319733BC"

The below steps will validate that the card is genuine by challenging the card to return a signature on a terminal generated unique data (random number) [using the ICC private key] and terminal validates this signature using ICC public key.

- Terminal provides a random number to the card. "4975101F"

This is done by issuing an INTERNAL AUTHENTICATE command to the card with the random number.

CMD : "0088000004<u>4975101F</u>00"

- Card respond with a certificate containing that random number encrypted under ICC private key.

RSP : "80819059731B6A2B5067E7CEC397A5494817A197A0C22AF93EA3BDED28C14FCED209A9F07C32B1D7CF048C04AC6D87E59B03E7D931057D4E8CB8180D969E484D7337D758BD91E38E20BE0D629A9A251B1F5AE3588C9D14A8D391ECC1A669453398FCC76EF1BF59D90A4AF8C79B711244370CABBF84847FB250C135858A999B0E14354A3D889578561B6CF8C5CA47264044FA04 9000"

- Terminal decrypt the certificate using ICC public key.

"6A0501<u>0302001F</u>BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB<u>D94062502C77AF1983C8F5DCD0F4DB88CAD153B1</u>BC"

- Refer to emv book 2 table 17 for the format of data recovered from signed dynamic app data.
- Validate the recovered data against that format to make sure it is correct.
- Extract the hash signature.

Dynamic Signature hash buffer1:: "D94062502C77AF1983C8F5DCD0F4DB88CAD153B1"

- Terminal recreates the signature by providing the random number that was previously shared with the card. (4975101F).
- Build a block of data following EMV Book 2 table 15.

"0501<u>0302001F</u>BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB<u>4975101F</u>"

- Compute a signature for this block of data [using SHA-1].

Dynamic Signature hash buffer2:: "D94062502C77AF1983C8F5DCD0F4DB88CAD153B1"

- Compare the 2 signatures and if they are equal, that means the card is genuine, and DDA was successful.
- The terminal shall set the 'Offline data authentication was performed' bit in the TSI to 1 (B1b8).

Up next... Cmobined/Dynamic Data Authentication CDA.