INFOY115
Secure software development and web security

Research Report

Be Prepared: The EMV Preplay Attack

Mike Bond | *University of Cambridge*
Marios O. Choudary | *University Politehnica Bucharest*
Steven J. Murdoch | *University College London*
Sergei Skorobogatov and Ross Anderson | *University of Cambridge*

Submitted to
Prof. R. Absil

Group members:

Simon-Olivier Morneau simonolivier.morneau@ulb.be
Maxime Fawe maxime.fawe@ulb.be

Fall 2021
Faculty of Computer Science
HE2B - ESI / Université Libre de Bruxelles

Introduction

EMV (named after Europay, Mastercard, and Visa, the founding members) is a set of interoperability guidelines based on the ISO/IEC 7816 (contact chip-based payment cards) and ISO/IEC 14443 (contactless payments) international standards.

Since the adoption of the standard by terminal and card manufacturers, EMV credit, debit, and prepaid cards have become ubiquitous in Europe, Canada, Latin America, and Africa. They also represent the majority of transactions in Southeast Asia and the United States.

The EMV consortium has been joined by a large number of partner payment networks around the world to encourage the adoption of common standards and compatible hardware, including Bancontact in Belgium and Carte Bancaire in France.

Considering the massive number of cards in circulation (9.89B cards in 2020) and the high amounts of money changing hands, it is not surprising that the standard has become an attractive target for criminals. According to the December 2020 Nilson report, the costs due to card fraud worldwide were estimated at 28.65 billion dollars in 2019 (of which more than 25 billion were for EMV cards). [1]

In this report, we will discuss the findings presented in the paper regarding a potential security vulnerability in the EMV transaction handling procedure, and a way to implement a preplay attack against chip-and-pin card terminals and ATMs.

EMV Transaction Processing

When accepting a payment at a point-of-sale (POS) or ATM location, the terminal proceeds in three steps, as shown in the following communications diagram:

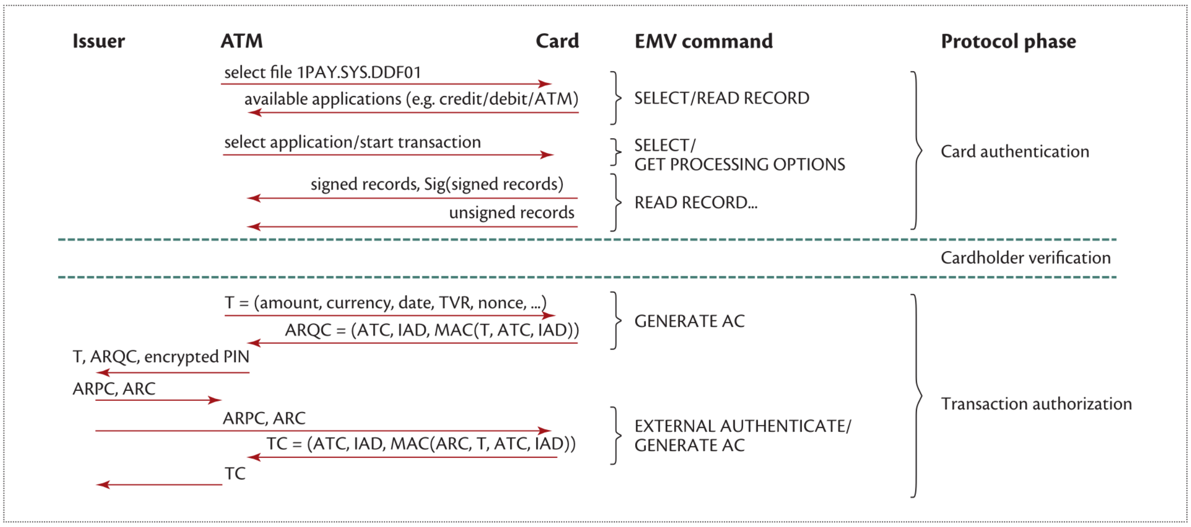

Issuer
ATM
Card
EMV command
Protocol phase
select file 1PAY.SYS.DDF01
available applications (e.g. credit/debit/ATM)
SELECT/READ RECORD
select application/start transaction
SELECT/
GET PROCESSING OPTIONS
signed records, Sig(signed records)
unsigned records
READ RECORD...
Card authentication
Cardholder verification
T = (amount, currency, date, TVR, nonce, ...)
ARQC = (ATC, IAD, MAC(T, ATC, IAD))
GENERATE AC
T, ARQC, encrypted PIN
ARPC, ARC
ARPC, ARC
TC = (ATC, IAD, MAC(ARC, T, ATC, IAD))
EXTERNAL AUTHENTICATE/
GENERATE AC
TC
Transaction authorization

Figure 1: Outline of an EMV transaction at an ATM. Although the messages between the card and ATM have been verified, messages between the issuer and ATM might vary depending on card scheme rules.

Cardholder authentication

In this phase, the terminal sends a request to the integrated chip of the card for the stored record corresponding to the EMV standard. In the case of regular chip-and-pin payments, this file is called 1PAY.SYS.DDF01, and 2PAY.SYS.DDF01 for contactless payments.

The card returns a list of supported EMV applications, which correspond to a specific payment processing network (Visa, Mastercard, American Express, Bancontact, etc). A single card often contains applications for multiple card schemes, which hold different priorities depending on the situation. For example, one card network can be used domestically, and a different one internationally, which is the case of most debit cards in Belgium, with Bancontact preferred at local stores vs. Visa or Maestro abroad.

The terminal selects the highest priority application that it supports, and sends a GET PROCESSING OPTIONS command to the card to retrieve the transaction options set by the issuing bank, as well as the card information. This includes the preferred language of the customer, which cardholder verification method to use, and the card number and expiry date.

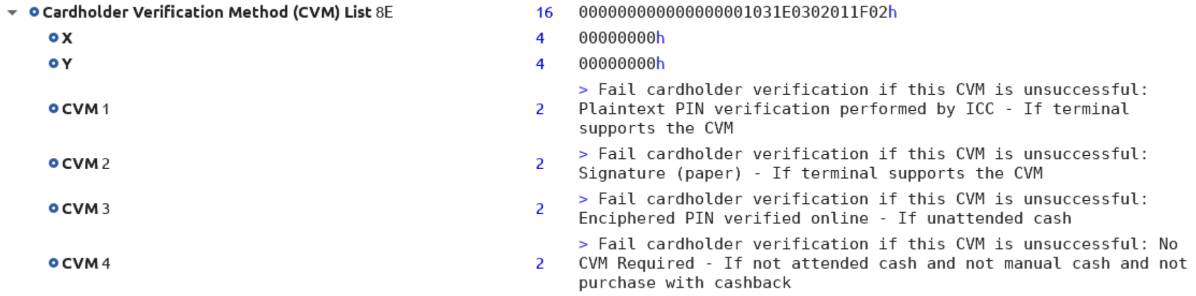
Finally, the digital signature that is joined to the records is checked to detect data corruption or tampering.

Cardholder verification

In the second phase, the terminal parses the list of CVMs (Cardholder Verification Method) returned by the card. This information allows the terminal to know which method to use to verify the identity of the person trying to use the card. Typically, this includes offline PIN verification using the PIN stored in memory on the card, online PIN verification using the issuing bank's servers, and a signature fallback option if the PIN pad is inoperable for example.

Additional conditions can be imposed by the bank to influence the allowed CVMs. For instance, many banks require the use of a PIN to make a cash withdrawal, but may allow the use of a signature or skip verification entirely for purchases under a certain threshold amount.

Once a verification method is selected, the terminal performs the required steps to confirm the user's identity, and stores the result of the verification for use in the last step.



```
▾ Cardholder Verification Method (CVM) List 8E   16  000000000000000001031E0302011F02h
    X                                              4  00000000h
    Y                                              4  00000000h
                                                      > Fail cardholder verification if this CVM is unsuccessful:
    CVM 1                                          2  Plaintext PIN verification performed by ICC - If terminal
                                                      supports the CVM
                                                      > Fail cardholder verification if this CVM is unsuccessful:
    CVM 2                                          2  Signature (paper) - If terminal supports the CVM
                                                      > Fail cardholder verification if this CVM is unsuccessful:
    CVM 3                                          2  Enciphered PIN verified online - If unattended cash
                                                      > Fail cardholder verification if this CVM is unsuccessful: No
    CVM 4                                          2  CVM Required - If not attended cash and not manual cash and not
                                                      purchase with cashback
```

Figure 2: Example of CVM data stored on a Belgian debit card, accessed and decoded using the Cardpeek software. This issuer requires offline PIN verification or a signature for most transactions, with no verification being allowed for non-cash transactions and online PIN verification being necessary for cash withdrawals at an ATM.

Transaction authorization

In this step, the card, the ATM or terminal, and the issuing bank exchange cryptographically signed messages about the transaction.

First, the terminal sends identifying information to the card about the transaction, including the amount authorized, currency, date, results of cardholder verification from the previous step, and a nonce (called the UN, Unpredictable Number in EMV jargon). This nonce is used as a protection measure against replay attacks since it helps make every transaction unique.

The card chip can then compute a MAC (Message Authentication Code) containing the transaction data as well as the current value of the ATC (Application Transaction Counter) which is a value that gets incremented at every processed transaction. This counter helps prevent transaction online replay attacks by allowing the issuing bank to check if the transaction ID has already been used, and decline any possible duplicate transaction. In the EMV standard, this MAC is called the ARQC (**A**uthorization **Req**uest **C**ryptogram).

The ATM or terminal then sends the ARQC to the card issuer along with the transaction data and the encrypted PIN entered by the user. The issuer verifies that the ATC has not been reused, that the cryptographic signature is valid, that the entered PIN is correct, and that the account holder has enough money in their balance. If the transaction passes these checks, the issuing bank sends a reply with a second MAC (referred to as the ARPC, or **A**uthorization **R**es**p**onse **C**ryptogram) and a status code for the transaction, the ARC (**A**uthorization **R**esponse **C**ode).

The terminal can then forward the ARPC to the card for cryptographic verification (this helps protect against a man-in-the-middle attack impersonating the issuer).

Finally, the card generates a digitally signed TC (**T**ransaction **C**ertificate) that will eventually be sent to the issuer to complete the transaction settlement.

Multiple flaws in EMV protocol

The researchers refuted a formal analysis published in 2011 regarding the security of EMV by pointing out two mistakes.

First, the authors of the analysis assumed that the UN (Unpredictable Number) was a fresh nonce while EMV didn't require this in the first instance.

Second, they modeled the issuer of the card and the terminal as a single entity which is not only inaccurate but also rules out the possibility of an insecure channel between the issuer and the terminal.

These two errors prevented the analysis from finding this flaw: the issuing bank relies on a random number generated by the terminal used for the transaction. It means that if the ATM has been infected with malware to sabotage the choice of UN (which, according to the article, has already occurred in Eastern Europe and in the US), the security of the transaction would be compromised.

However, even if the terminal is trustworthy, it does not mean that the exchange is secure: with a weak RNG algorithm, the UN can be predictable and a hacker could mount an attack indistinguishable from card cloning. The fact is that the initial EMV standard had few (if any) requirements for the randomness of the nonce. It simply required that four consecutive transactions performed by the same terminal must have unique UNs. Thus, an ATM manufacturer could pass the test with a simple counter.

Finally, a second protocol flaw was identified by the researchers: the terminal ID is not required in authenticated messages (which typically means an attacker could record some transaction data on one terminal and reuse it afterwards on another one).

Empirical data on UNs

Based on previous work proving that a preplay attack was possible against EMV if the RNG was manipulated and their own analysis of the EMV protocol, the researchers needed to assess the quality of RNG algorithms implemented in ATMs and POS terminals with empirical data.

With this aim in mind, they performed more than 1.000 transactions on 22 different ATMs and 5 POS terminals using modified debit cards. They added a microcontroller and memory to these in order to record UNs produced by ATMs and POS terminals during balance inquiries. They used this type of transaction to minimize the number of withdrawals and avoid triggering fraud monitoring systems.

Sample of collected data from ATMs

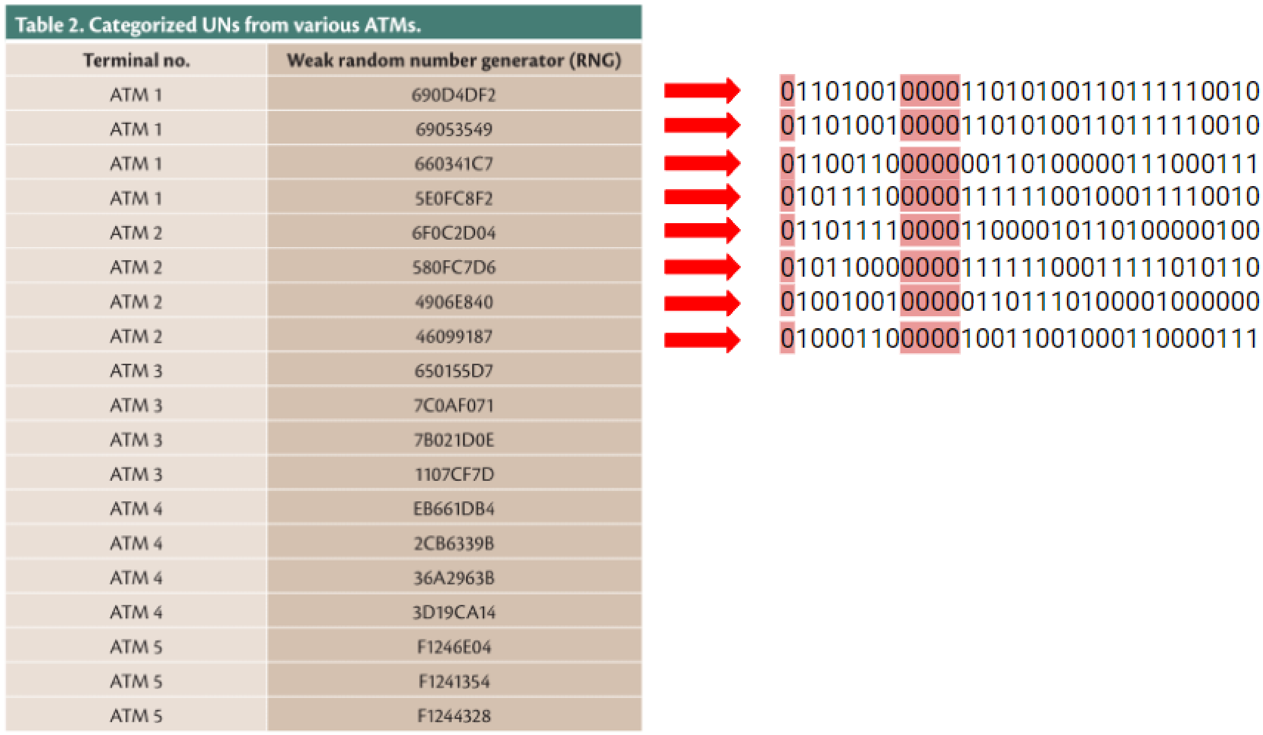


Table 2. Categorized UNs from various ATMs.

Terminal no.	Weak random number generator (RNG)
ATM 1	690D4DF2
ATM 1	69053549
ATM 1	660341C7
ATM 1	5E0FC8F2
ATM 2	6F0C2D04
ATM 2	580FC7D6
ATM 2	4906E840
ATM 2	46099187
ATM 3	650155D7
ATM 3	7C0AF071
ATM 3	7B021D0E
ATM 3	1107CF7D
ATM 4	EB661DB4
ATM 4	2CB6339B
ATM 4	36A2963B
ATM 4	3D19CA14
ATM 5	F1246E04
ATM 5	F1241354
ATM 5	F1244328

```
01101001000011010100110111110010
01101001000011010100110111110010
01100110000000110100000111000111
01011110000011111100100011110010
01101111000011000010110100000100
01011000000011111100011111010110
01001001000001101110100001000000
01000110000010011001000110000111
```

Figure 3: Table showing UNs generated by several ATMs with evidence of pattern in the random number generator.

As highlighted in the table above, the researchers found a pattern in the numbers generated by the ATMs: the first bit and the third nibble are set to zero for each UN. This reduces the entropy of the nonce from 32 to 27 bits. Out of all tested ATMs, half of them shared this issue. As the sample contained ATMs of different generations and running on different operating systems, the researchers think this is a problem coming from EMV kernel post processing rather than from the RNG.

Sample of collected data from POS terminals

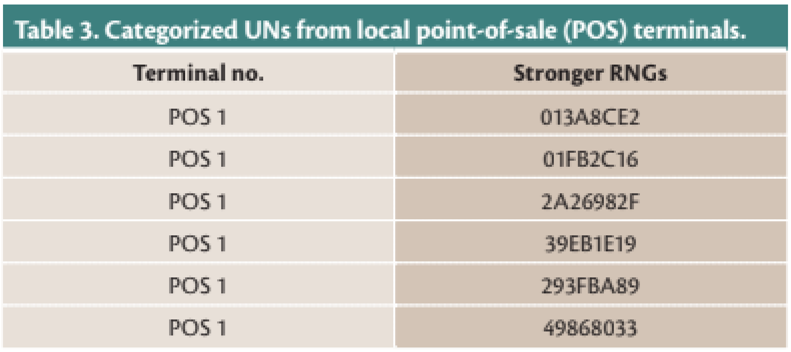


Table 3. Categorized UNs from local point-of-sale (POS) terminals.

Terminal no.	Stronger RNGs
POS 1	013A8CE2
POS 1	01FB2C16
POS 1	2A26982F
POS 1	39EB1E19
POS 1	293FBA89
POS 1	49868033

Figure 4: Table showing UNs generated by a POS terminal: even if the RNG is stronger, the first bit remains zero, which could indicate the use of a signed integer.

Classes of ineffective RNGs

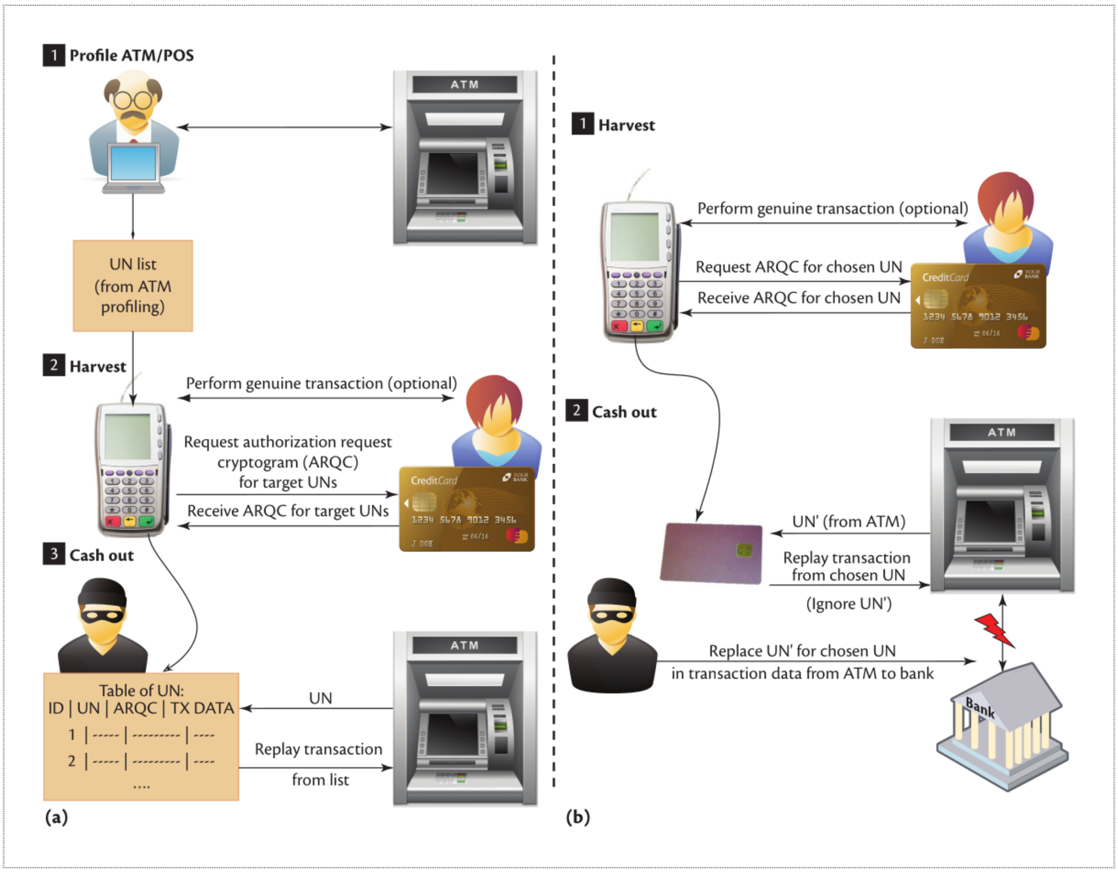
Based on their analysis of the retrieved UNs, the researchers identify three broad classes of ineffective RNGs:

- Weak RNG algorithms such as counters, timestamps,... directly used as the UN.
- RNG with little or no seeding using, for example, libc calls like time() or rand().
- RNG with a predictable state (randomness restarted on power-up or RNG that only relies on previous transactions data).

Implementing an attack

Knowing that a given implementation of the UN generator is flawed, it should be possible for an attacker to capture genuine transaction data from a card terminal or ATM, and use this recorded information to make a fraudulent transaction later.

The researchers suggest two possible plans of attack to exploit this vulnerability, as shown in the following diagram.



1 Profile ATM/POS
ATM
UN list (from ATM profiling)
2 Harvest
Perform genuine transaction (optional)
Request authorization request cryptogram (ARQC) for target UNs
Receive ARQC for target UNs
CreditCard
3 Cash out
Table of UN:
ID | UN | ARQC | TX DATA
1 | ----- | --------- | ----
2 | ----- | --------- | ----
....
UN
Replay transaction from list
(a)
1 Harvest
Perform genuine transaction (optional)
Request ARQC for chosen UN
Receive ARQC for chosen UN
2 Cash out
UN' (from ATM)
Replay transaction from chosen UN
(Ignore UN')
Replace UN' for chosen UN in transaction data from ATM to bank
Bank
(b)

Figure 5: Overview of the preplay attack: (a) using a weak random number generator and (b) tampering with the UN at an ATM or POS terminal.

In option (a), on the left, the attacker probes a vulnerable ATM for a list of UNs. The genuine card is asked to pre-generate valid ARQCs from the transaction data corresponding to the collected UNs. The fraudster can then access the ATM, and replay the ARQC that matched the UN submitted by the ATM.

In option (b), the attacker again harvests valid ARQCs from a card. This time, however, the attacker has tampered with the terminal or ATM by infecting it with a piece of malware to allow them to change the UN submitted to the issuing bank with the one from a previously generated ARQC. In addition, they use a programmable smart card to run custom software that can receive the EMV command to generate an ARQC, but replies with a previously recorded response from the legitimate card.

Final recommendations

The most important element to protect cardholders and issuers against fraud using the preplay attack is to ensure the strength of the random number generator used to provide the UN (Unpredictable Number) to the card chip in future implementations of the EMV standard.

While the EMV standard previously did not enforce a specific method of implementing UN generation, the most recent version of the implementation guidelines states that terminal hardware vendors should rely only on cryptographically secure implementations of hashing functions, incorporating as many unique transaction variables into the calculation as possible. This may include a terminal unique identifier, which would prevent attackers from capturing transaction data on one terminal and carrying out a transaction on a different machine.

In addition, implementers must be careful not to assume that there exists a secure communications channel between the card terminal and the issuing bank. In other words, the ATM or terminal could have been targeted by malware, or the RNG logic could have been sabotaged by a determined attacker with physical access to the hardware. This is what the authors suggest has been happening to an ATM network operator in Majorca, where a string of unexplained cash withdrawals that looked like legitimate transactions had been detected in the months before the paper was finalized.

In such a scenario, the terminal cannot be trusted, and the issuer should rely as much as possible on other fraud detection mechanisms to stop a replay attack in progress and prevent future financial losses. In particular, this attack would cause the ATC (Application Transaction Counter) to be incremented by a large amount in a short time, resulting in a large jump in value or transactions arriving in the wrong order from the point of view of the issuing bank. This should never happen for legitimate transactions and should result in a card being flagged as compromised immediately.

Bibliography

[1] Nilson Report, “Annual fraud report.” Dec. 2021. Accessed: Dec. 07, 2021. [Online]. Available: https://nilsonreport.com/upload/content_promo/1187_9123%5b2%5d.pdf

[2] The EMV Consortium, “Integrated Circuit Card Specifications for Payment Systems - Book 3 - Application Specification.” EMVCo, Nov. 2011. Accessed: Dec. 07, 2021. [Online]. Available: https://www.emvco.com/wp-content/uploads/2017/05/EMV_v4.3_Book_3_Application_Specification_20120607062110791.pdf

[3] The EMV Consortium, “A Guide to EMV Chip Technology.” EMVCo, Nov. 2014. Accessed: Dec. 07, 2021. [Online]. Available: https://www.emvco.com/wp-content/uploads/2017/05/A_Guide_to_EMV_Chip_Technology_v2.0_20141120122132753.pdf