ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/262271594

Cloning credit cards: a combined pre-play and downgrade attack on EMV contactless

Conference Paper · August 2013

CITATIONS
35

READS
11,042

2 authors:

Michael Roland
Johannes Kepler University of Linz
74 PUBLICATIONS **608** CITATIONS
SEE PROFILE

Josef Langer
University of Applied Sciences Upper Austria
68 PUBLICATIONS **876** CITATIONS
SEE PROFILE

All content following this page was uploaded by Michael Roland on 07 October 2014.

The user has requested enhancement of the downloaded file.

Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless

Michael Roland, Josef Langer
NFC Research Lab Hagenberg
University of Applied Sciences Upper Austria
{*michael.roland, josef.langer*}*@fh-hagenberg.at*

Abstract

Recent roll-outs of contactless payment infrastructures – particularly in Austria and Germany – have raised concerns about the security of contactless payment cards and Near Field Communication (NFC). There are well-known attack scenarios like relay attacks and skimming of credit card numbers. However, banks and credit card schemes often mitigate these attacks. They explain that attacks are impractical (e.g. in a relay attack an attacker needs to have RF access to a victim's card while performing a payment transaction) or even impossible (e.g. skimmed data does not contain the dynamic authorization codes that are normally required to perform a payment transaction). This paper introduces an attack scenario on EMV contactless payment cards that permits an attacker to create functional clones of a card that contain the necessary credit card data as well as pre-played authorization codes. The card clones can then be used to perform a limited number of EMV Mag-Stripe transactions at any EMV contactless payment terminal.

1 Introduction

Recent announcements of roll-outs of contactless credit, debit and pre-paid card infrastructures boost the fear among customers about security issues in these contactless payment systems. With contactless payment cards the traditional contact-based smartcard interface is complemented with or replaced by an antenna. The most prominent global contactless payment card standard is the *EMV Contactless Specifications for Payment Systems* [9], which has been adopted by all major credit card brands. This standard is based on the ISO/IEC 14443 standard for proximity integrated circuit cards. The contactless interface between a smartcard terminal and a payment card uses inductive coupling at an operating frequency of 13.56 MHz. The communication technology is compatible to Near Field Communication (NFC) – a technology that is available in many new mobile phones.

Contactless communication has several benefits. For instance, transactions become more convenient because cards no longer need to be taken out of a user's wallet and inserted into a point-of-sale (POS) terminal. Also the mechanical wear down of both the cards and the terminals is significantly reduced. Above that, the roll-out of contactless credit card terminals for mobile payment use-cases seems to start the global adoption of EMV standards and to finally phase-out magnetic stripe technology [26].

Besides these advantages, this contactless technology comes with several security concerns. In particular eavesdropping, skimming and relay attacks are considered to be potential problems:

- *Eavesdropping* refers to a scenario where an attacker picks up the RF signals transmitted between a terminal and a card from a distant location.
- *Skimming* refers to a scenario where an attacker captures credit card data and later uses this information in fraudulent payment transactions.
- *Relay attack* refers to a scenario where an attacker forwards the communication between a dummy credit card ("proxy") that is used to perform a payment transaction at a credit card terminal and a reader device ("mole") that accesses the real credit card.

The current trend to include NFC technology into mobile phones significantly simplifies skimming and relay attacks. NFC-enabled mobile phones can be used to access and read data from contactless credit cards as well as to emulate credit cards in a relay

attack or based on data previously skimmed from a card.

In this paper, we present a new attack scenario based on skimming that can be used to create card clones that successfully perform the EMV Mag-Stripe protocol for contactless payment cards defined in the EMV Contactless Kernel 2 specification [8]. Valid dynamic card verification codes (CVC) which are necessary to authorize these payments are obtained from an original card with a pre-play approach. Further, we observed a second weakness with credit cards from various issuers which allows to downgrade a full EMV credit card to perform a contactless EMV Mag-Stripe transaction.

2 Related Work

Haselsteiner and Beitfuß [16] describe eavesdropping as an important issue of wireless communication technologies. They suggest that, while normal communication distances for ISO/IEC 14443 and NFC are at most 10 centimeters, eavesdropping is possible even if there is a distance of several meters between the attacker and the attacked devices.

While eavesdropping extracts information from legitimate communication between a credit card and a payment terminal, skimming uses any information that could potentially be used to perform a fraudulent payment transaction. This information could be obtained through directly reading data from a card, through eavesdropping or even through social engineering. Credit card data may range from cardholder names, credit card numbers and card verification codes to digital data extracted from real credit cards. Sufficient information for skimming could be obtained by means as simple as photocopying the plastic card or by harvesting in call centers [18]. However, already articles from the early 1990s [4,6,7] explain how to decode the magnetic stripe of a credit card and how to encode this information onto a blank card in order to create a functional card clone. Today, these credit card clones are often created by harvesting magnetic stripe data as well as PIN codes at ATMs [14]. With contactless payment cards, skimming may be possible even without being in physical possession of a card. For instance, Paget [22] describes how to extract static data from chip-based credit cards to later encode this information onto magnetic stripe cards. Even though that information lacked card verification codes the card clones were accepted by certain merchants.

Another scenario, the relay attack, was initially described by Conway [3] as the "*Grandmaster Chess Attack*" and by Desmedt et al. [5] as "*mafia fraud*". This attack simply extends the communication distance between a genuine credit card terminal and a genuine credit card. Thus, a team of two attackers can forward the communication of a credit card terminal (operated by attacker A) to a victim's credit card (operated by attacker B). Hancke [15] found that the relay attack is particularly useful in combination with contactless smartcards: In that case attacker B does not need to be in physical possession of the relayed credit card but, instead, only needs to place the mole in close proximity to the card-under-attack.

The current trend to include NFC technology into mobile phones significantly simplifies skimming and relay attacks. Francis et al. [10, 11] propose the use of NFC-enabled mobile phones as platforms for attacks against ISO/IEC 14443 based smartcard systems. NFC-enabled mobile phones can be used in reader/writer mode to access contactless credit cards to extract data for skimming or to relay communication to a proxy in a relay attack (cf. [1, 12, 13]). In card emulation mode, an NFC-enabled mobile phone could be used as a card-clone in a skimming attack or as the proxy in a relay attack (cf. [10, 13]).

Nevertheless, particularly skimming is hindered in modern chip-based credit cards by the use of "strong cryptography". While potentially sensitive information (e.g. the credit card number, the expiry date, and – with older cards – also the cardholder's name [17]) can be skimmed from contactless credit cards, this information is usually considered insufficient to conduct a fraudulent payment transaction. Though, the static information that is freely readable from the chip would be enough to pay at some online merchants (e.g. Amazon), most merchants would require the card verification code that is written on the back of the card. However, this code is not available on the chip. Instead, the chip authorizes transactions based on a secret key that is securely stored inside the smartcard chip and that cannot be read through smartcard commands.

Despite their use of secure smartcard technology and state-of-the-art cryptography, even chip-based payment cards have known weaknesses. For instance, there is a well-known issue with the offline PIN verification protocol of EMV's *Chip & PIN* discovered by Murdoch et al. [21]. This weakness allows to completely bypass PIN verification on certain cards. A new attack described by Bond et al. [2] reveals that many EMV terminal implementations trade security for simplicity: Supposedly unpredictable (random) numbers generated by these implementations for use in cryptographic protocols become predictable. As a consequence, the "strong"

cryptographic protocols are severely weakened. Due to the weakened cryptographic protocol, an attacker could calculate a series of transaction authorizations with a real credit card in advance. Later, these pre-calculated authorizations could be used on a card clone to perform actual payment transactions (or to withdraw cash at an ATM).

Our attack scenario uses an approach similar to that by Bond et al. [2]. It also aims at abusing weakened cryptographic protocols to perform a pre-play attack. While their approach targets specific terminal implementations that have predictable random number generators, our approach targets general limitations of the EMV contactless protocol in Mag-Stripe mode. Compared to the attack by Bond et al. [2], our attack does not require the authorized amount to be known during the pre-calculation. Moreover, our attack does not need any specific knowledge about the implementation and the initialization state of the random number generator of the terminal that is later fed with the pre-played data. However, our attack is limited by the maximum amount that can be authorized with a PIN-less contactless transaction.

3 EMV Contactless Kernel 2

The *EMV Contactless Specifications for Payment Systems* [9] come in four different flavors: Kernel 1, 2, 3, and 4. They are named "kernel" specifications as they primarily target the terminal software implementation for interacting with compliant payment cards. Each kernel specification covers the payment systems of specific credit card brands.

Our discoveries focus on the *Kernel 2 Specification* [8], though we have not looked for similarities in other kernel specifications. According to the specification document, Kernel 2 covers the protocols required to interact with payment cards supporting the MasterCard PayPass brand or any other payment card that explicitly requests usage of Kernel 2 [8].

The EMV protocol of Kernel 2 supports two different operating modes: emulation of the magnetic stripe system over contactless transactions (Mag-Stripe mode) and the full EMV protocol (EMV mode).

According to MasterCard's requirement specification for PayPass M/Chip [20], a PayPass card using the MasterCard brand must always support contactless Mag-Stripe mode transactions and may optionally support EMV mode transactions. Similarly, that specification requires that MasterCard PayPass terminals must always support contactless Mag-Stripe mode transactions and may optionally support EMV mode transactions. Moreover, MasterCard's rules [19] suggest that within the Single European Payment Area (SEPA), cards and terminals issued in 2011 and later must support both, PayPass EMV mode and PayPass Mag-Stripe mode. PayPass cards using the Maestro brand, however, must never support contactless Mag-Stripe [20].

In EMV mode, the static data contained in the card is signed by the card issuer. Thus, the payment terminal can verify that the card data is authentic. In addition the card signs the payment transaction using a secret key that is only known to the card and that can usually not be extracted from the card. This can be used to verify that the card itself is authentic. As a consequence, a payment terminal could even verify and store transactions authenticated by a card offline for later processing.

Compared to processing a classic magnetic stripe transaction, the authorization of EMV mode transactions requires additional interfaces between the payment terminals, the acquiring bank and the card issuer. In order to also use the existing magnetic stripe infrastructure without significant modifications, Kernel 2 supports Mag-Stripe mode.

In Mag-Stripe mode, the card stores information comparable to the data on a magnetic stripe. Instead of a static authentication code encoded into the Mag-Stripe data (or printed on the back of the card), the card generates dynamic authentication codes to authorize payments. The authentication code (dynamic *card verification code*, CVC3) authenticates only the card and not the contents of a payment transaction. The codes are calculated from secret key material that is only known to the card and its issuer. Besides the secret key, the dynamic CVC3 is derived from a transaction counter (ATC) that is incremented by the card for each generated code and an unpredictable number (UN) that is provided by the POS terminal. The transaction counter hinders re-use of previously used authentication codes (*re-play*). The unpredictable number hinders pre-generation of authentication codes on a real card for later use in fraudulent transactions (*pre-play*).

A typical Mag-Stripe mode contactless credit card transaction consists of the following smartcard command sequence (a detailed trace can be found in Appendix A):

1. The POS selects (SELECT command) the Proximity Payment System Environment (PPSE) and the card responds with a list of supported EMV payment applications.

2. The POS selects (SELECT command) the credit/debit card application and the card responds with application details.

3. The POS requests the credit card application's processing options (GET_PROCESSING_OPTIONS command). The credit card applet responds with the application interchange profile and one or more application file locators. The application interchange profile indicates if the card supports EMV mode in addition to Mag-Stripe mode, what types of data authentication the card supports and if cardholder verification is supported. The application file locators point to files that contain static credit card data (e.g. the Mag-Stripe data which is typically located in record 1 of an elementary file with the short file ID 1).

4. The POS reads (READ_RECORDS command) the Mag-Stripe data from record 1 of the data file with the short file ID 1. The credit card applet responds with the Mag-Stripe version, track 1 and track 2 data. This data also contains information on how to embed the dynamic CVC3, the ATC and the UN into the track 1 and track 2 discretionary data.

5. The POS instructs the card to compute the cryptographic checksum (COMPUTE_CRYPTOGRAPHIC_CHECKSUM command) for a given unpredictable number (UN). The credit card applet responds with the application transaction counter (ATC) and with the dynamically generated CVC3 for track 1 and track 2.

Most of the data exchanged in a Mag-Stripe transaction is static for all transactions (e.g. the Mag-Stripe data). COMPUTE_CRYPTOGRAPHIC_CHECKSUM is the only APDU command-response pair that contains dynamically generated data that differs for each transaction: the unpredictable number (UN, 4 bytes) generated by the POS, the transaction counter (ATC, 2 bytes) and the dynamic CVC3s (2 bytes for each track) generated by the card. Each COMPUTE_CRYPTOGRAPHIC_CHECKSUM command that is sent to the card must be preceded by a fresh GET_PROCESSING_OPTIONS command. Thus, the minimum sequence for generating a dynamic CVC3 is

1. SELECT the credit/debit card application,

2. GET_PROCESSING_OPTIONS, and

3. COMPUTE_CRYPTOGRAPHIC_CHECKSUM.

4 Attack Surface

During our evaluation of EMV contactless credit/debit cards, we identified several weaknesses in Kernel 2's Mag-Stripe protocol and in current authorization systems. An attacker could use these weaknesses to create functional card clones from existing contactless payment cards.

In the first attack scenario, a pre-play attack, the attacker pre-calculates a number of dynamic card verification codes (CVC3) for the Mag-Stripe protocol from a genuine contactless Mag-Stripe card. These pre-played codes are then stored on the functional card clone and used when an authorization code is requested by the POS.

In the second attack scenario, a downgrade attack, the pre-play attack is extended to modern credit/debit cards that support both, EMV mode and Mag-Stripe mode by selectively hiding the EMV mode capabilities from the POS terminal.

4.1 Pre-play Attack

EMV systems are known for weaknesses related to the unpredictable numbers that are used to prevent pre-play in transactions. For instance, Bond et al. [2] discovered that the random number generation is weak in several ATM and POS terminal implementations. An attacker could abuse this to predict the random number sequence of a terminal or to even trigger the terminal to use a specific random number. However, their attack requires detailed knowledge of the random number generator implementation of the target terminal and may also require thorough timing of the attack.

With the Mag-Stripe protocol, the attacker does not need to rely on issues in terminal implementations. The unpredictable numbers used in the COMPUTE_CRYPTOGRAPHIC_CHECKSUM command are systematically weakened by the protocol design. As a result of this design flaw, the possible range of unpredictable numbers is greatly reduced.

The "*Unpredictable Number (Numeric)*" field used in COMPUTE_CRYPTOGRAPHIC_CHECKSUM is a 4-byte value. Consequently, in theory, the number could range from 0 to 4,294,967,295 ($2^{32} - 1$). However, the EMV Kernel 2 specification [8] limits the contents of this field to a BCD-encoded numeric value. BCD (binary coded decimal) is an encoding where the digits of a decimal number are used as digits in a hexadecimal number. For instance 1500_d becomes $\texttt{00001500}_h$. Thus, each nibble of the 4-byte value can hold one decimal digit. As a result, the unpredictable number can range from 0 to 99,999,999.

While this is about 43 times less than the full range, it would still not be feasible to pre-compute authorization codes for each possible unpredictable number in that range.

However, the Mag-Stripe protocol further reduces the size of the unpredictable number to n_{UN} digits:

$$\begin{aligned} n_{\text{UN}} &= k_{\text{TRACK1}} - t_{\text{TRACK1}} \\ &= k_{\text{TRACK2}} - t_{\text{TRACK2}}, \end{aligned}$$

where $k_{\text{TRACK}x}$ is the number of bits set in the "Track x bit map for UN and ATC" ($\text{BMAP}_{\text{ATC, UN, TRACK}x}$) and $t_{\text{TRACK}x}$ is the "Track x number of ATC digits". Both values are stored in the card's Mag-Stripe data file.

$\text{BMAP}_{\text{ATC, UN, TRACK}x}$ is the bit mask that defines the positions within the discretionary data of track x where the unpredictable number and the application transaction counter will be embedded. Typical values that we encountered during our tests were $\texttt{00000000FE00}_h$ for track 1 and $\texttt{1FC0}_h$ for track 2.

$t_{\text{TRACK}x}$ is the size (in digits) of the application transaction counter. A typical value that we encountered during our tests is $\texttt{04}_h$.

Based on these values we can calculate n_{UN}:

$$\begin{aligned} \text{BMAP}_{\text{ATC, UN, TRACK1}} &= \texttt{00000000FE00}_h \\ k_{\text{TRACK1}} &= 7 \\ t_{\text{TRACK1}} &= \texttt{04}_h = 4 \\ \text{BMAP}_{\text{ATC, UN, TRACK2}} &= \texttt{1FC0}_h \\ k_{\text{TRACK2}} &= 7 \\ t_{\text{TRACK2}} &= \texttt{04}_h = 4 \\ n_{\text{UN}} &= k_{\text{TRACK1}} - t_{\text{TRACK1}} \\ &= k_{\text{TRACK2}} - t_{\text{TRACK2}} \\ &= 3 \end{aligned}$$

As a consequence, the unpredictable number may have at most 3 digits. Therefore, it is in the range from 0 to 999 (more than 4,000,000 times less than the maximum possible range). Tests with various cards (Table 1) revealed that the maximum size of the unpredictable number ranges between 1 and 3 in existing products (see Table 2).

In order to generate dynamic CVC3s, the credit/debit card application must be selected (SELECT) and then a sequence of GET_PROCESSING_OPTIONS followed by COMPUTE_CRYPTOGRAPHIC_CHECKSUM has to be repeated for every CVC3 (cf. section 3). Using this minimum command sequence, we achieved an average computation speed of 1,000 CVC3s per minute with an Android app running on a Google Galaxy Nexus[1] for cards 1, 2, and 3. Thus, a full set of authorization codes that covers the whole range (0 to 999) of the "Unpredictable Number (Numeric)" field can be pre-calculated in only about a minute.

[1]We achieved similar results for a Google Nexus S.

Table 1: Tested payment cards

Card	Issuer	Product[a]	Protocol
1	LBBW[b]	MobileTag, debit, reloadable	Mag-Stripe only
2	LBBW[b]	debit, reloadable	M/Chip
3	RBI AG[c]	PRELOAD, debit, non-reloadable	M/Chip
4[f]	RBI AG[c]	GOLD, credit	M/Chip
5[f]	Bancorp[d]	Google Wallet Prepaid, debit, reloadable	Mag-Stripe only
6[f]	Vincento[e]	Kalixa watch2pay, debit, reloadable	M/Chip

[a] All cards were MasterCard PayPass cards compatible to EMV Contactless Kernel 2.
[b] LBBW: Landesbank Baden-Württemberg
[c] RBI AG: Raiffeisen Bank International AG
[d] Bancorp: The Bancorp Bank
[e] Vincento: Vincento Payment Solutions Limited
[f] We did not use cards 4, 5, and 6 for any payment transactions or any tests that could change the internal state of the cards because the cardholders did not want their cards to be used for any activities that might get their cards blocked for abuse.

Table 2: n_{UN}, k and t for various cards

Card	n_{UN}	$k_{\text{TR.1}}$	$t_{\text{TR.1}}$	$k_{\text{TR.2}}$	$t_{\text{TR.2}}$
1	3	7	4	7	4
2	3	7	4	7	4
3	3	8	5	8	5
4	3	8	5	8	5
5	2	6	4	6	4
6	1	6	5	4	3

As a consequence, an attacker needs about a minute of communication with an EMV Mag-Stripe card to pre-generate sufficient information for performing a successful payment transaction. Opportunities to conduct such an attack could be a crowded bus or even while customers hand over their card to POS staff operating a credit card terminal.

However, the usability of this attack is limited by the application transaction counter (ATC). The CVC3 is calculated from the unpredictable number, the card data, the card's secret key and the ATC:

$$\text{CVC3} = f(\text{UN, Card data, Secret card key, ATC}).$$

The ATC increases monotonically for each transac-

tion and, consequently, protects against re-play attacks. Thus, a CVC3 (and its associated ATC) that has been used in one transaction cannot be re-used in another transaction. This is true even if the unpredictable number is the same for both transactions. However, this does not affect our attack scenario as the ATC+CVC3 sets generated during the pre-play attack will not be reused by the original card for legitimate transactions anyways.

In addition to the re-play protection, the ATC may be used for a second purpose: An issuer could reject all transactions that use an ATC lower than the highest ATC received from that card in a previous transaction. Applied to our pre-play attack, this means that

1. each full set of pre-played authorization codes can only be used for one (worst-case) payment transaction and
2. the pre-played authorization codes become invalid as soon as the original card is used in a payment transaction.

Therefore, for our attack scenario we assume that issuers would block cards that use ATCs in a non-monotonic sequence. Consequently, once an attacker uses a certain ATC+CVC3 set, all ATC+CVC3 sets with a lower ATC must be discarded. For instance, if the attacker harvests CVC3 codes for all unpredictable numbers from 0 to 999 (in incremental order), and later uses these ATC+CVC3 sets in response to the unpredictable number "4", the pre-played sets for UN = 0, 1, 2, 3, and 4 cannot be used in another transaction. Therefore, in the worst-case scenario an attacker can only perform one transaction when only one ATC+CVC3 set had been collected for each possible unpredictable number. Moreover, an attacker's "window of opportunity" closes as soon as the legitimate card is used in another transaction as all previously pre-played ATC+CVC3 sets would become invalid.

4.2 Downgrade Attack

So far the pre-play attack scenario works only for Mag-Stripe mode. However, MasterCard's rules [19] state that new PayPass cards and terminals issued within the Single European Payment Area (SEPA) after January 1st, 2011 must support the full PayPass M/Chip protocol (EMV mode and Mag-Stripe mode). If both, the card and the terminal, support EMV mode, they will perform an EMV mode transaction and will not fall back to Mag-Stripe mode. Therefore, a card clone that contains a copy of all static card data and the pre-played list of UN+ATC+CVC3 sets will cause a terminal to perform an EMV mode transaction which is not supported by that simple card clone. As a consequence, the pre-play attack for such a card would only work for terminals that support only Kernel 2's Mag-Stripe mode.

Table 3: Tested POS terminals

POS	Model
1	Hypercom Artema Hybrid with ViVOpay 5000 contactless
2	VeriFone Vx680 GPRS CTLS
3	Ingenico iCT250

Nevertheless, tests with several POS terminals (see Table 3) revealed that terminals can be tricked into using the Mag-Stripe protocol even though the original card supports EMV mode. The attacker simply needs to change the *application interchange profile* contained in the response to the GET_PROCESSING_OPTIONS command: While the original application interchange profile would contain a flag that indicates support for EMV mode, the modified application interchange profile would have this flag cleared. To achieve this, the attacker could simply set the new application interchange profile to `0000`. This trick works because Mag-Stripe mode does not provide any means to authenticate the data returned in response to the GET_PROCESSING_OPTIONS command.

As a result, an attacker could create a functional contactless Mag-Stripe card with pre-played data extracted from a genuine card that supports EMV mode. This extends the pre-play attack scenario to any Kernel-2-compatible contactless EMV card and to acceptance at any payment terminal that implements Kernel 2.

5 Mounting the Attack

We created a proof-of-concept system to verify the combined pre-play and downgrade attack scenario. The system consists of a Java Card application and an Android app. The Java Card application contains a simple Mag-Stripe credit card structure that can be filled with the contents of the Mag-Stripe data file and with a list of pre-play data. The Android app runs on a Google Galaxy Nexus (or any other NFC-enabled Android device). The app collects static data (Mag-Stripe data file) and pre-play data (dynamic CVC3 codes) from a genuine credit card and stores that information in a table. The app can later transfer the collected information onto a smartcard

that contains our Java Card application.

5.1 Collecting Pre-play Data

In order to collect the pre-play data, the Android app accesses the genuine credit card through its contactless interface. The data is retrieved with the same command sequence that is used for regular payment transactions:

1. SELECT the Proximity Payment System Environment (PPSE).
2. SELECT the first credit/debit card application listed in the PPSE.
3. GET_PROCESSING_OPTIONS and extract the location of the Mag-Stripe data.
4. READ_RECORDS for the Mag-Stripe data and store the retrieved Mag-Stripe data:
   `9F6C 02 0001`
   `9F62 06 000000000038`
   `9F63 06 00000000FE00` $\rightarrow k_{\text{TRACK1}} = 7$
   `56 34 423533...30`
   `9F64 01 04` $\rightarrow t_{\text{TRACK1}} = 4$
   `9F65 02 0038`
   `9F66 02 1FC0` $\rightarrow k_{\text{TRACK2}} = 7$
   `9F6B 13 53...0F`
   `9F67 01 04` $\rightarrow t_{\text{TRACK2}} = 4$
5. The Mag-Stripe data is used to calculate the number of digits of the unpredictable number:

$$\begin{aligned} n_{\text{UN}} &= k_{\text{TRACK1}} - t_{\text{TRACK1}} \\ &= 7 - 4 \\ &= 3. \end{aligned}$$

6. For every possible unpredictable number (in this case for every number in the range from 0 to 999), the app issues a COMPUTE_CRYPTOGRAPHIC_CHECKSUM command followed by a fresh GET_PROCESSING_OPTIONS command to prepare for the next CVC3 computation.
7. The result of COMPUTE_CRYPTOGRAPHIC_CHECKSUM is stored in a table (cf. Table 4).

5.2 Creating a Clone-Card

The card clone Java Card application runs on an NXP JCOP card and provides a rudimentary contactless EMV Mag-Stripe interface and a second interface ("CloneCard interface") to copy pre-play data onto the card. The EMV Mag-Stripe interface responds with static data structures extracted from the transaction analysis in Appendix A for the commands SELECT PPSE, SELECT credit/debit application, and GET_PROCESSING_OPTIONS.

Table 4: Storage of pre-play data sets

		CVC3	
UN	ATC	Track 1	Track 2
00000000	001B	1FE4	2AB2
00000001	001C	EF32	C91F
	...		
00000998	0402	BAFD	E01B
00000999	0403	149B	01A3

By using the application interchange profile of a Mag-Stripe card in response to the GET_PROCESSING_OPTIONS command, the card clone automatically performs the downgrade attack as it does not advertise EMV mode capabilities.

In response to the READ_RECORDS command for the Mag-Stripe data (record 1 of the elementary file with the short file ID 1), the card clone responds with a byte array that can be customized through the CloneCard interface. The CloneCard interface provides a command SET_MAGSTRIPE_DATA for this purpose.

In response to COMPUTE_CRYPTOGRAPHIC_CHECKSUM, the card clone looks up the random number received from the POS terminal in a list of available UN+ATC+CVC3 sets and returns the ATC and the CVC3 codes. If no UN+ATC+CVC3 set is available for the given unpredictable number, the card returns the error code `6F00`. The list of UN+ATC+CVC3 sets can be loaded into the card through the CloneCard interface's command SET_MAGSTRIPE_AUTH.

After collecting the pre-play data from a real credit card, the Android app expects the user to tap a second card with the CloneCard interface. The Android app first stores the collected Mag-Stripe data onto the card clone with the SET_MAGSTRIPE_DATA command. Then, the app stores all collected UN+ATC+CVC3 sets onto the card clone using the SET_MAGSTRIPE_AUTH command.

5.3 Recording a Transaction Log

For every transaction, the card clone records the used unpredictable number (and whether a UN+ATC+CVC3 set was found or an error was returned) in a list. The list can be read through the CloneCard interface's command GET_MAG-

STRIPE_HISTORY. This information is later used to analyze which unpredictable numbers were tried by the POS terminal.

5.4 Improving the Attack

During our evaluation of the attack scenario, we found two ways to improve the attack:

1. It is possible to further reduce the pool of unpredictable numbers for some cards.
2. One of the POS terminals we used for our tests permits to reject a limited number of unpredictable numbers without interrupting the payment transaction.

5.4.1 Further Reducing n_{UN}

In the Mag-Stripe protocol, the size of the unpredictable number is defined by the card with the values $k_{\text{TRACK}x}$ and $t_{\text{TRACK}x}$:

$$n_{\text{UN}} \quad = \quad k_{\text{TRACK}x} - t_{\text{TRACK}x}.$$

As a consequence, by clearing the bits reserved for the unpredictable number in the "Track x bit map for UN and ATC" ($\text{BMAP}_{\text{ATC, UN, TRACK}x}$), $k_{\text{TRACK}x}$ can be reduced to $t_{\text{TRACK}x}$. Therefore, n_{UN} becomes zero. In addition, it may be necessary to set all digits of the "Track x discretionary data" that are supposed to be filled by the unpredictable number to "0".

If, for example, the following data has been extracted from the Mag-Stripe data record:

- Track 1 bit map for UN and ATC:
  `9F63 06 00000000FE00`
- Track 1 data:
  `56 34 42...31393138 38323231 30303030`
  `30303032 32313030 30303030`
- Track 1 number of ATC digits:
  `9F64 01 04`
- Track 2 bit map for UN and ATC:
  `9F66 02 1FC0`
- Track 2 data:
  `9F6B 13 53...0000000000000F`
- Track 2 number of ATC digits:
  `9F67 01 04`

This data will be changed to:

- Track 1 bit map for UN and ATC:
  `9F63 06 00000000F000`
- Track 1 data:
  `56 34 42...31393138 38323231 30303030`
  `30303032 32313030 30303030`
- Track 1 number of ATC digits:
  `9F64 01 04`
- Track 2 bit map for UN and ATC:
  `9F66 02 1E00`
- Track 2 data:
  `9F6B 13 53...0000000000000F`
- Track 2 number of ATC digits:
  `9F67 01 04`

The POS terminal will then always use 0 as the unpredictable number. As a result, the attacker only needs to pre-play ATC+CVC3 sets for UN = 0.

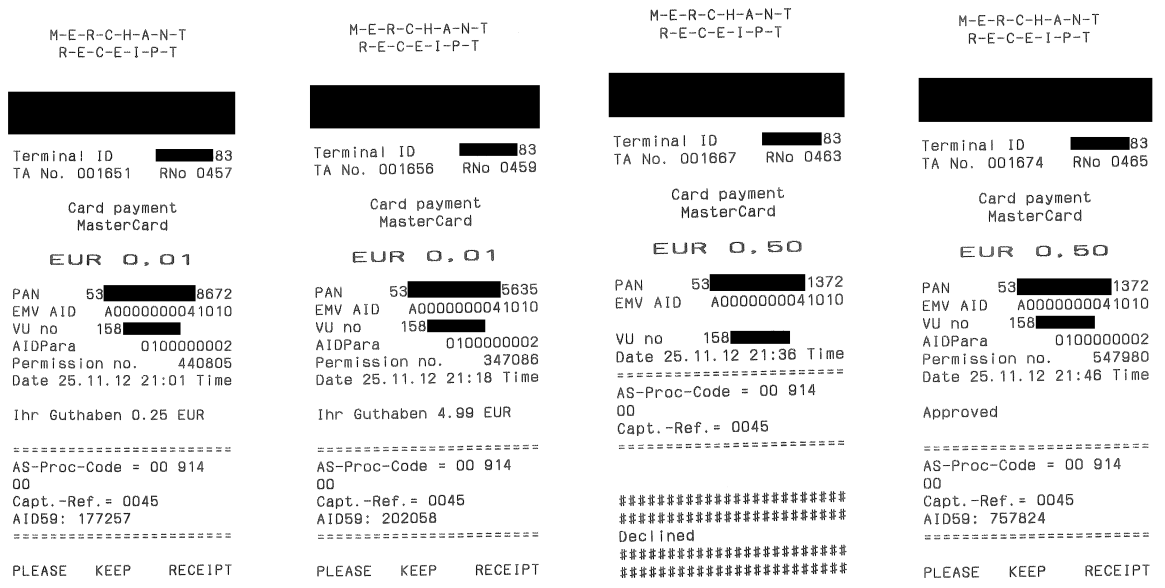
We found that this improvement is possible with some cards while it does not work with other cards. Specifically, the attack with UN limited to 0 worked with all cards issued by LBBW but did not work with cards issued by RBI AG. The reason is that (according to [8]) n_{UN} is sent to the card issuer as the least significant digit in the "Track x discretionary data". Therefore, an issuer has a means to detect and prevent this type of improved attack by checking if the expected n_{UN} has been used for a transaction. Nevertheless, some issuers obviously do not have such security checks in place.

5.4.2 Abusing Terminal-specific Weaknesses

We identified a weakness in one of the POS terminals (POS 1, see Table 3) that we used for our tests of the attack: When the card returns an error code (`6F00`) in response to the COMPUTE_CRYPTOGRAPHIC_CHECKSUM command, the terminal waits a few seconds and then retries to perform the payment transaction using a *different* unpredictable number. The terminal gives up only after approximately six unsuccessful tries. As a result, the card can choose between up to six unpredictable numbers. This means that the attacker does not need to pre-compute authentication codes for every possible unpredictable number. Consequently, this might be used to speed up the pre-play attack. However, we found that other POS terminals (POS 2 and 3) do not have this weakness.

6 Results

We tested the attack with POS 1, 2, and 3 (see Table 3) and cards 1, 2, and 3 (see Table 1). Cards 2 and 3 were downgraded from full EMV mode to Mag-Stripe mode only. We were able to successfully use the improved attack with UN forced to zero for cards 1 and 2 (both issued by LBBW) on all three terminals. We tried both, the improved attack and the regular pre-play attack for card 3 (issued by RBI



(a) successful transaction (b) successful transaction (c) failed transaction (d) successful transaction

Figure 1: Resulting merchant receipts for payment transactions using the pre-play attack. (a) is the result of card clone 1 with UN forced to 0 (ATC = `38FE`, CVC3 = `F940/4535`). (b) is the result of card clone 2 with UN forced to 0 (ATC = `0015`, CVC3 = `5F7F/1A95`). (c) is the result of card clone 3 with UN forced to 0 (ATC = `2E3B`, CVC3 = `74F8/ACA4`). (d) is the result of card clone 3 with regular pre-play (UN = `00000676`, ATC = `32DE`, CVC3 = `10EB/817C`).

AG). While the regular pre-play attack was successful with all tested terminals, the improved attack for card 3 failed at all of them. Figure 1 shows the resulting merchant receipts of POS 1. The tests with POS 2 and 3 had comparable results.

7 Possible Workarounds

Our analysis of the protocols involved in transactions with EMV Contactless Kernel 2 lead to several possible workarounds to mitigate some of the above attack scenarios.

7.1 Mag-Stripe vs. EMV Mode

MasterCard's requirement specification for PayPass M/Chip [20] defines that acquirers must include the following data into authorization request messages for PayPass (i.e. MasterCard contactless) transactions:

- A "*POS Entry Mode*" data element that indicates if the transaction has been performed using contactless EMV mode or contactless Mag-Stripe mode.
- A "*POS Terminal Device Input Capabilities*" data element that indicates if the transaction has been performed at a terminal that supports both contactless EMV mode and contactless Mag-Stripe mode or at a terminal that only supports contactless Mag-Stripe mode.

The same specification demands that the issuer host is capable of understanding this information in received authorization messages. Therefore, an issuer is capable of detecting the following transaction types:

- A contactless Mag-Stripe mode transaction has been performed at a terminal that supports only contactless Mag-Stripe mode.
- A contactless Mag-Stripe mode transaction has been performed at a terminal that supports both contactless EMV mode and contactless Mag-Stripe mode.
- A contactless EMV mode transaction has been performed at a terminal that supports both contactless EMV mode and contactless Mag-Stripe mode.

In addition to that information an issuer should know if an issued card has support for the full contactless EMV protocol (including EMV mode) or if it only supports contactless Mag-Stripe mode.

As a consequence, an issuer should be capable of detecting the fraudulent case where an EMV mode capable card was used in a Mag-Stripe mode transaction at a terminal that supports EMV mode. If the issuer rejects that case, the pre-play attack can only be used if either the card or the terminal supports only Mag-Stripe mode. A condition that, for instance, should never be the case for new cards and terminals within the Single European Payment Area (SEPA). However, our tests revealed that issuers do not currently perform such checks.

7.2 Inhibiting the Improved Attack

During our analysis we found that some cards (or more specifically one of the involved issuers) accepts transactions even if the unpredictable number is forced to zero. This would significantly improve the pre-play attack as the pre-calculation of authorization codes would be much faster. However, the EMV specification [8] implemented a counter measure: n_{UN} is sent to the issuer as the last digit in the "Track x discretionary data". Thus, an issuer has a means to detect this type of attack. Nevertheless, not all issuers have such checks in place.

7.3 Maximizing n_{UN}

The current Mag-Stripe protocol seems to be very limited in the size of the unpredictable number. The origin of this limitation seems to be the limited number of characters/digits of the discretionary data of tracks 1 and 2. Therefore, it seems to be difficult (or even impossible) to use the full range of a 4-byte unpredictable number while staying backwards compatible. Nevertheless, our tests revealed that some issuers use a very low value for n_{UN}. With $n_{\text{UN}} = 3$, our application needs about a minute to pre-compute all UN+ATC+CVC3 sets. Incrementing n_{UN} by one would increase the pre-computation time by a factor of 10. Thus, for $n_{\text{UN}} = 4$ the pre-computation would already take approximately 10 minutes. Therefore, we suggest that issuers use the maximum possible size for the unpredictable number that fits into the discretionary data of tracks 1 and 2.

8 Conclusion

In this paper we discussed a new attack scenario on contactless EMV payment cards that permits to create functional clones of such cards. Our attack uses contactless skimming to transfer data from the original card to the clone. EMV's contactless Mag-Stripe mode implements dynamic authorization codes based on random challenges sent by the point-of-sale terminal to prevent skimming. However, using an NFC-enabled smartphone and a typical credit/debit card it only takes about a minute to pre-play enough authorization codes to answer any possible challenge sent by a point-of-sale terminal. Even though, many new credit cards and terminals use the improved contactless EMV mode protocol, we found that such cards can be downgraded to the vulnerable contactless Mag-Stripe mode protocol.

We created a smartcard application that implements a basic contactless Mag-Stripe card and that can be filled with data collected from genuine credit cards. We also created an Android application that extracts the data from genuine credit cards and transfers this information onto our card clone. We verified the attack scenario by creating clones of various credit/debit cards and by successfully paying with these clones at real point-of-sale terminals. While our tests were performed under lab conditions, we used real POS terminals backed by an acquirer operating in Austria that did not differ from those used in regular roll-outs around Austria.

An analysis of the protocols involved in these payment transactions revealed that there are already several mechanisms available to partially mitigate such attacks. However, these counter measures are currently not implemented by the issuers of our cards.

We reported our findings to MasterCard. They quickly acknowledged these vulnerabilities. Nevertheless, they pointed out that their protocols and requirement documents already provide sophisticated countermeasures. Thus, it seems to be left to each card issuer to implement these measures.

Acknowledgments

This work is part of the project "High Speed RFID" within the EU program "Regionale Wettbewerbsfähigkeit OÖ 2007–2013 (Regio 13)" funded by the European regional development fund (ERDF) and the Province of Upper Austria (Land Oberösterreich).

References

[1] Anderson, R. Position Statement in RFID S&P Panel: RFID and the Middleman. In *Financial Cryptography and Data Security*, vol. 4886/2007 of *LNCS*. Springer Berlin Heidelberg, 2007, pp. 46–49.

[2] Bond, M., Choudary, O., Murdoch, S. J., Skorobogatov, S., and Anderson, R. Chip and Skim: cloning EMV cards with the pre-play attack. Computing Research Repository (CoRR), arXiv:1209.2531 [cs.CY], Sept. 2012.

[3] Conway, J. H. *On Numbers and Games*. Academic Press, 1976.

[4] Count Zero. Card-O-Rama: Magnetic Stripe Technology and Beyond. In *Phrack*, vol. 37. Nov. 1992.

[5] Desmedt, Y., Goutier, C., and Bengio, S. Special Uses and Abuses of the Fiat-Shamir Passport Protocol (extended abstract). In *Advances in Cryptology — CRYPTO '87*, vol. 293/2006 of *LNCS*. Springer Berlin Heidelberg, 1988, pp. 21–39.

[6] Dr. Mabuse. Magneetkaarten. In *Hack-Tic*, vol. 8. 1990.

[7] Dr. Mabuse. Magneetkaarten deel II. In *Hack-Tic*, vol. 9/10. 1990.

[8] EMVCo. *EMV Contactless Specifications for Payment Systems – Book C-2: Kernel 2 Specification*, Mar. 2011.

[9] EMVCo. *EMV Contactless Specifications for Payment Systems (Books A–D)*, Mar. 2011.

[10] Francis, L., Hancke, G. P., Mayes, K. E., and Markantonakis, K. Potential misuse of NFC enabled mobile phones with embedded security elements as contactless attack platforms. In *Proceedings of the 1st International Workshop on RFID Security and Cryptography (RISC'09)*. IEEE, London, UK, Nov. 2009.

[11] Francis, L., Hancke, G. P., Mayes, K. E., and Markantonakis, K. On the security issues of NFC enabled mobile phones. In *Int. J. Internet Technology and Secured Transactions*, vol. 2(3/4). 2010, pp. 336–356.

[12] Francis, L., Hancke, G. P., Mayes, K. E., and Markantonakis, K. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In *Radio Frequency Identification: Security and Privacy Issues*, vol. 6370/2010 of *LNCS*. Springer Berlin Heidelberg, 2010, pp. 35–49.

[13] Francis, L., Hancke, G. P., Mayes, K. E., and Markantonakis, K. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. Cryptology ePrint Archive, Report 2011/618, 2011.

[14] Gallagher, S. Automated robbery: how card skimmers (still) steal millions from banks. *Ars Technica* (June 2012). `http://arstechnica.com/security/2012/06/automated-robbery-how-card-skimmers-still-steal-millions-from-banks/`.

[15] Hancke, G. P. A Practical Relay Attack on ISO 14443 Proximity Cards. `http://www.rfidblog.org.uk/hancke-rfidrelay.pdf`, Jan. 2005.

[16] Haselsteiner, E., and Breitfuss, K. Security in Near Field Communication (NFC) – Strengths and Weaknesses. In *Workshop on RFID Security 2006 (RFIDsec 06)*. Graz, Austria, July 2006.

[17] Heydt-Benjamin, T. S., Bailey, D. V., Fu, K., Juels, A., and O'Hare, T. Vulnerabilities in First-Generation RFID-enabled Credit Cards. In *Financial Cryptography and Data Security*, vol. 4886/2007 of *LNCS*. Springer Berlin Heidelberg, 2007, pp. 2–14.

[18] Little, A. Overseas credit card scam exposed. *BBC News* (Mar. 2009). `http://news.bbc.co.uk/2/hi/uk_news/7953401.stm`.

[19] MasterCard Worldwide. *MasterCard Rules*, July 2011.

[20] MasterCard Worldwide. *PayPass – M/Chip Requirements*, Dec. 2011.

[21] Murdoch, S. J., Drimer, S., Anderson, R., and Bond, M. Chip and PIN is Broken. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, Oakland, CA, USA, May 2010, pp. 433–446.

[22] Paget, K. Credit Card Fraud – The Contactless Generation. Presentation at ShmooCon 2012, 2012. `http://www.shmoocon.org/2012/presentations/Paget_shmoocon2012-credit-cards.pdf`.

[23] Roland, M. Applying recent secure element relay attack scenarios to the real world: Google Wallet Relay Attack. Computing Research Repository (CoRR), arXiv:1209.0875 [cs.CR], Sept. 2012.

[24] Roland, M. *Security Issues in Mobile NFC Devices*. PhD thesis, Johannes Kepler University Linz, Department of Computational Perception, Jan. 2013.

[25] Roland, M., Langer, J., and Scharinger, J. Applying Relay Attacks to Google Wallet. In *Proceedings of the Fifth International Workshop on Near Field Communication (NFC 2013)*. IEEE, Zurich, Switzerland, Feb. 2013.

[26] Svigals, J. The Long Life and Imminent Death of the Mag-Stripe Card. In *IEEE Spectrum*, vol. 49(6). June 2012.

Appendix A Mag-Stripe Mode

A typical EMV Contactless Kernel 2 Mag-Stripe mode contactless credit card transaction consists of the following smartcard command sequence [23–25]:

1. POS → Card: SELECT PPSE:
   `00 A4 0400 0E 325041592E5359532E4444463031 00`

2. Card → POS: The card responds with a list of supported EMV payment applications:
   `6F 23` (FCI template)
   `84 0E 325041592E5359532E4444463031`
   (DF name: "2PAY.SYS.DDF01")
   `A5 11` (Proprietary information, BER-TLV)
   `BF0C 0E` (FCI issuer discretionary data)
   `61 0C` (Application template)
   `4F 07 A000000004 1010`
   (Application identifier: MasterCard credit/debit card)
   `87 01 01`
   (Application priority indicator)
   `9000` (Status: Success)

3. POS → Card: SELECT credit/debit card application:
   `00 A4 0400 07 A000000004 1010 00`

4. Card → POS: The card responds with application details:
   `6F 17` (FCI template)
   `84 07 A000000004 1010` (DF name)
   `A5 0C` (Proprietary information, BER-TLV)
   `50 0A 4D617374657243617264`
   (Application label: "MasterCard")
   `9000` (Status: Success)

5. POS → Card: The POS requests the credit card application's processing options (GET_PROCESSING_OPTIONS command):
   `80 A8 0000 02 8300 00`

6. Card → POS: The credit card applet responds with the application interchange profile and one or more application file locators:
   `77 0A` (Response message template)
   `82 02 0000`
   (Application interchange profile: Mag-Stripe mode only, online transactions only, no cardholder verification, etc.)
   `94 04 08 01 01 00`
   (Application file locator: Mag-Stripe data file, short file ID = 1, first record = 1, last record = 1)
   `9000` (Status: Success)

7. POS → Card: The POS reads (READ_RECORDS command) the Mag-Stripe data from record 1 of the data file with the short file ID 1:
   `00 B2 010C 00`

8. Card → POS: The credit card applet responds with the Mag-Stripe version, track 1 and track 2 information:
   `70 75`
   (Non inter-industry nested data object template)
   `9F6C 02 0001`
   (Mag-Stripe application version: Version 1)
   `9F62 06 000000000038`
   (Track 1 bit map for CVC3: The bits set in this bit map mark the positions within the track 1 discretionary data where the POS terminal should embed the obtained track 1 CVC3. Consequently, only three digits of the track 1 CVC3 are used.)
   `9F63 06 00000000FE00`
   (Track 1 bit map for UN and ATC: The bits set in this bit map mark the positions within the track 1 discretionary data where the POS terminal should embed the unpredictable number and the application transaction counter. Consequently, a total of 7 digits of UN and ATC can be embedded.)
   `56 34` (Track 1 data)
   `42` (ISO/IEC 7813 structure "B" format)
   `3533xxxx xxxxxxxx xxxxxxxx xxxxxxxx`
   (PAN "53xx xxxx xxxx xxxx")
   `5E` (Field separator "^")
   `202F` (empty cardholder name "␣/")
   `5E` (Field separator "^")
   `31323132` (Expiry date "12"/"12")
   `313231` (Service code "121")
   `31393138 38323231 30303030 30303032 32313030 30303030`
   (Track 1 discretionary data)
   `9F64 01 04`
   (Track 1 number of ATC digits: The application transaction counter has 4 digits. As UN and ATC have 7 digits in total, the UN must be a 3 digit number.)
   `9F65 02 0038`
   (Track 2 bit map for CVC3: The bits set in this bit map mark the positions within the track 2 discretionary data where the POS terminal should embed the obtained track 2 CVC3. Consequently, only three digits of the track 2 CVC3 are used.)
   `9F66 02 1FC0`
   (Track 2 bit map for UN and ATC: The bits set in this bit map mark the positions within the track 2 discretionary data where the POS terminal should embed the unpredictable number and the application transaction counter. Consequently, a total of 7 digits of UN and ATC can be embedded.)
   `9F6B 13` (Track 2 data)
   `53xx xxxx xxxx xxxx` (PAN)
   `D` (Field separator)
   `1212` (Expiry date)
   `121` (Service code)
   `0000000000000`
   (Track 2 discretionary data)
   `F` (Padding)
   `9F67 01 04`
   (Track 2 number of ATC digits: The application transaction counter has 4 digits. As UN and ATC have 7 digits in total, the UN must be a 3 digit number.)
   `9000` (Status: Success)

9. POS → Card: The POS instructs the card to compute the cryptographic checksum for a given unpredictable number `nnnnnnnn` (COMPUTE_CRYPTOGRAPHIC_CHECKSUM command):
   `80 2A 8E80 04 nnnnnnnn 00`

10. Card → POS: The credit card applet responds with the application transaction counter (`xxxx`) and with the dynamically generated CVC3 for track 1 (`yyyy`) and track 2 (`zzzz`):
    `77 0F` (Response message template)
    `9F61 02 zzzz` (CVC3 Track 2)
    `9F60 02 yyyy` (CVC3 Track 1)
    `9F36 02 xxxx` (ATC)
    `9000` (Status: Success)

View publication stats