DOI: http://dx.doi.org/10.5769/C2011006

Evaluating the security of the Brazilian ePassport

Brute force attack on the BAC protocol

Ivo de Carvalho Peixinho[1], and Auto Tavares da Camara Junior[2]
(1) Brazi lian Federal Police, Brasília/DF, Brazil, peixinho.icp@dpf.gov.br
(2) Brazilian Federal Police, Brasília/DF, Brazil, junior.atcj@dpf.gov.br

***Abstract* — This work evaluates the feasibility of performing a brute force attack on a Brazilian electronic passport in an attempt to find the correct key for the BAC protocol and to be able to read the contents, including biometric information. It is assumed that the attacker has some standard tools, like a common reader, a portable computer and *open source software* for reading electronic passports. The attack model uses a *skimming* technique, where the passport is not in use and is near the attacker for enough time to be able to perform the attack.**

Key-words — Security; ePassport; BAC; brute force;

1. Introduction

Electronic passports (ePassports) are the latest evolution on passports, which are used to identify travelers going to other countries. They were first introduced in 1998 in Malaysia [1], but at that time there was little concern about security. The ePassports have an electronic *tag* that uses RFID [4] technology (ISO 14443), enabling it to be read wirelessly at a small distance.

In 2004 [2], the International Civil Aviation Organization (ICAO) issued a series of recommendations and protocol specifications to countries that wished to implement ePassports. These specifications are grouped on a document called Doc 9303 – Machine Readable Travel Documents [3]. Although the document has many protocols designed for security, most of them are optional and may not be implemented by issuing countries. According to ICAO, an electronic passport must contain a contactless electronic reading device with internal storage space of at least 32 kilobits of size, coded accordingly to the Logical Data Structure (LDS) defined on the 9303 Document, with a minimum of the MRZ data on the first group of data and a facial image of the bearer on the second group. The passports that have these characteristics must have a specific symbol on the cover that identifies them as ePassports, as seen on Figure 1.

Figure 1 - ePassport identifier symbol

The Brazilian government started to issue RFID enabled passports in late 2010, as part of a project to update Brazilian passport, called PROMASP 2. In this project, many security features were added, including the RFID tag that was made standard by ICAO. As a matter of fact, the project implemented all of the optional security measures defined by the 9303 document, and it uses two kinds of biometry: photo in hi-res (protected by the BAC protocol), and fingerprints (protected by the EAC standard).

In this paper, the BAC (Basic Access Control) protocol implementation of Brazilian passport will be evaluated. This protocol is the first security barrier for an attacker to read the contents of the passport, so it's very likely that it will be the first to be attacked. The Brazilian ePassport is shown on Figure 2.

Figure 2 - Brasilian ePassport

2. Related Work

Liu et al [6] did a similar work evaluating the security of the Germany and Netherlands ePassport. In this work, the attack model was to eavesdrop on a legitimate connection and use a parallel FPGA computer to break the encrypted data. Although the work claims an average time of 14 hours for finding the MRZ, it assumes that the age of the bearer can be guessed, further reducing the entropy. It also needs some special hardware (120 FPGAs operating in parallel on a 64 bit data bus), and proximity to the reader, which can be tricky in some cases (ex: immigration inspection point).

Nithyanand [1], made a survey on the evolution of the security protocols on ePassports, according to the ICAO

specifications, since its first generation in 2004, until the proposal of the 2.1 version of the EAC protocol.

Initially the survey addresses ePassport technologies, including biometrics, RFID and Public Key Infrastructure (PKI). On RFID technologies, the work details the issues on the adoption of this technology, specially the detection, tracking, eavesdropping and cloning of RFID tags. Also using some signal propagation formulas, the work states that the typical distance for a correct reading of an ePassport is approximately 10 centimeters. This small distance makes it difficult for an attacker to perform an unauthorized reading, because it needs to be very close to the victim.

Later, the work details the security protocols present on the first ePassport generation: *Passive Authentication* (PA), *Basic Access Control* (BAC) and *Active Authentication* (AA). Following, it shows the changes on the second generation, specially the *Extended Access Control* (EAC), and its protocols: *Chip Authentication* (CA) and *Terminal Authentication* (TA). These protocols were created to establish an additional security level for protection of secondary biometrics (fingerprints and iris), and to allow the mutual authentication of both the terminal and the ePassport. Finally, the third generation proposed in 2008 introduces a new protocol, named *Password Authenticated Connection Establishment* (PACE). It was introduced with the objective of replacing BAC, along with an updated version to the EAC standard.

At the end, the article shows the vulnerabilities known on ePassports:

1. Attacks on RFID technology: cloning, eavesdropping, and *skimming* [5]. In short, it is possible to obtain some information that prevents cloning, using physical attacks.
2. Attacks on the first generation protocols: some protocols, like BAC and AA are not mandatory, so some countries may not adopt them. This option makes unauthorized reading and RFID cloning trivial. BAC also has some low entropy problems on its keys. The information used to derive the cryptography keys shorten the original entropy from 56 bits (3DES) to something between 25 and 35 bits (depends on the country implementation), which may be cracked with a brute force attack. Finally, the specs does not have access control rules, meaning that a successful read would gather all the information on the ePassport, even if it is not needed.
3. Attacks on the second-generation protocols: the EAC standard, although more secure still needs BAC to work, so it inherits all its vulnerabilities. ePassports, as being passive, do not have a correct time information, so it may be difficult for them to verify the validity of a terminal certificate, which may be used by an attacker that has an expired certificate. EAC is also vulnerable to a denial of service attack, where a malicious terminal sends lots of invalid certificates, making the RFID tag to malfunction.
4. Attacks on the third generation protocols: the work states that the third generation solved all the previous problems, except the verification of the validity of the digital certificates problem. In this standard, the BAC protocol is replaced by an improved one, named PACE. Not many countries at the moment are using the third generation standards.

3. The BAC Protocol

The BAC protocol is one of the security protocols that are described by the 9303 documents. Its main purpose is to prevent *skimming* [5], which is the unauthorized reading of an ePassport, and *eavesdropping*, which is intercepting a valid communication between a passport and a reader. When a passport uses BAC, the reader cannot directly read the tag contents. Instead, it must prove that it is authorized to read the chip, using a challenge-response protocol [3]. The protocol establishes that the reader must derive a couple of Basic Access Keys, denoted K_{ENC} and K_{MAC}. These keys are derived from information present on the data page of the passport, specifically the Machine Readable Zone (MRZ).

The MRZ is a series of characters on the data page of the passport that uses a special font and convention, which allows an inspection reader to scan the characters and use Optical Character Recognition (OCR) to retrieve the actual data.

Figure 3 - Data Page of an ePassport

On Figure 3 we can see the MRZ at the bottom of the data page. It has a defined pattern, so the reader can automatically translate and extract the subject's data. The character "<" is a filling character, so the reader can separate the different fields. The first line of the MRZ contains the following information:

- P – Identification of the document as a passport
- UTO – Issuing Country Code
- ERIKSSON – Primary ID (Last Name)
- ANNA MARIA – Secondary ID (Given Names)

Similarly, the second line of the MRZ has additional information:

- L898902C – Document Number
- 3 – Check digit for Doc. Number
- UTO – Nationality
- 690806 – Birth Date (6 Aug 1969)
- 1 – Check digit for Birth Date
- F – Sex (Female)
- 940623 – Expiration Date (23 Jun 1994)
- 6 – Check digit for Exp. Date
- ZE184226B – Optional Field
- 1 – Check digit for Opt. Field
- 4 – Overall check digit

For the generation of the BAC keys, only the second line of the MRZ is used. The keys are derived from the document number, date of birth and date of expiry (including check digits), using a SHA-1 hash of this information. On the above example, the MRZ info used for BAC key calculation would be:

- L898902C<369080619406236

This hash is used as a seed to compute the actual keys. The actual protocol involves generating random numbers, performing a challenge-response authentication using the generated keys and the 3DES symmetric cipher, and deriving session keys for encrypted communication between the reader and the passport. The actual steps of the BAC protocol are detailed below:

1. The terminal reads the MRZ information from the data page of the passport, using Optical Character Recognition (OCR), and extracts the information needed for generating BAC keys: passport number, birth date and expiry date, including the check digits. Using this info it derives K_{SEED}, using the result of an SHA-1 hash on the MRZ data and using the 126 most significant bits.
2. The previous result is joined with a 32-bit counter. The counter is set with 00000001, which indicates we are generating an encryption key. A new SHA-1 hash is calculated, and the result is split in two keys, K_a and K_b, using the first 8 bytes on the first key and the following 8 byes on the second. The remaining of the hash bits is discarded.
3. The resulting keys are used on the 3DES algorithm, which uses 56 bits of key length and 8bits of parity for each key. Using the generated keys, the parity bits are adjusted to correctly reflect the parity calculation. These keys made the set K_{ENC}, which are used for encrypting.
4. The same process on items 2 and 3 are repeated, this time with 00000002 on the counter, which indicates that we are generating a Message Authentication Code (MAC) key. These keys made the set K_{MAC}, which are used for integrity.
5. The terminal requests a random 8-byte number from the ePassport, named RND.ICC. This request is made with a specific command named GET CHALLENGE.
6. The terminal generates two random numbers. One with 8 bytes named RND.IFD and other with 16 bytes named K_{IFD}.
7. The terminal calculates S = RND.IFD || RND.ICC || K.IFD. The operator || indicates joining the values.
8. Result S is encrypted with K_{ENC}, using 3DES. This result is named E_IFD.
9. Integrity data is calculated encrypting E_IFD with K_{MAC}, resulting in M_IFD.
10. The joined results (E_IFD || M_IFD) are sent to the ePassport, using the MUTUAL AUTHENTICATE command.
11. The ePassport deciphers the received data and verifies the integrity data. The result is split and the received RND.ICC is compared with the initially generated for authentication measures. The passport has the K_{ENC} and K_{MAC} keys stored internally, so it does not need to calculate them.
12. The passport generates a 16-byte random number named K_{ICC}. K_{IFD}, sent by the terminal and K_{ICC} are joined, using XOR bitwise operation, resulting on a new K_{SEED}. The session keys KS_{ENC} and KS_{MAC} are now generated using the same method described previously.
13. The passport generates an initial value for a sequential counter using the 4 least significant bytes of RND. ICC and RND.IFD. This result is named SSC.
14. The passport joins RND.ICC, RND.IFD and K_{ICC}. The result is named R.
15. The passport encrypts R with K_{ENC}, using 3DES. The result is named E_ICC.
16. The passport encrypts E_ICC with K_{MAC}, resulting M_ICC.
17. The passport sends E_ICC and M_ICC as answer to the terminal.
18. The terminal decrypts and verifies the received data, and compares RND.IFD with the originally generated, completing the mutual authentication.
19. The terminal does a XOR of K_{IFD} and K_{ICC}, generating the same K_{SEED} generated by the passport.
20. The KS_{ENC} and KS_{MAC} are generated using the same method used by the passport.

Table 1 - Crypto operations on the BAC protocol

Passaport	Terminal
Generation of an 8 bit random (RND.ICC)	SHA-1 of MRZ info
Deciphering of E_IFD and M_IFD (3DES)	SHA-1 of K_{SEED} + counter
Verification of RND.IFD	Adjusting of 3DES keys partity
	SHA-1 of K_{SEED} + counter
	Generation of 8 bit random (RND.IFD) and K_{IFD} (16 bits)
	Ciphering of RND.IFD||RND. ICC||K.IFD (E_IFD – 3DES)
	Ciphering of E_IFD (M_IFD – 3DES MAC)

21. The SSC counter is generated using the same method used by the passport.

The following table shows the basic crypto operations performed on BAC by the terminal and the ePassport on a single trial of the BAC protocol with the wrong key. Looking at the protocol steps, we can see that the step 11 would fail and the subsequent steps probably will not take place.

Obviously the ePassport has much less performance doing the cryptographic operations than the terminal, due to its limited size, passive stance and low power.

4. BAC Entropy Vulnerability

Many researchers have stated that the BAC protocol have some vulnerabilities regarding the entropy of the BAC keys [6]. In fact, the information used for deriving the keys has lower combinations than you would expect from a modern random key. Today's symmetric cryptography standards use keys of 128 bits to 256 bits of size, such as the AES algorithm [7]. The passport number is usually numeric and sequential because it is easier to implement this way. Some countries may even shorten the amount of digits by fixing some of them (to denote a particular series, for example). The birth date has only two digits for the year, the date only ranges from 1-31, and the month from 1-12. Considering that the passport is still valid, the expiration date can be only at most x years on the future, as x being the number of years that a passport remains valid (usually 5 or 10 years). The check digits can be calculated from the data, so they are irrelevant on calculating the entropy of the keys.

If the resulting entropy of the BAC keys is low enough, it can enable an attacker to try to find the correct keys using a brute force approach: it uses a computer to try all combinations of possible MRZ data, generates the corresponding keys and try to authenticate on the passport. If the attacker is successful, it has performed a *skimming* attack, as stated earlier. Other kind of attack would be an *eavesdropping* attack, where an attacker uses a reader with *sniffing* capabilities to collect the traffic between the reader and the passport. Using this traffic, the attacker can try to *crack* the encryption keys, using a dedicated computer with great processing power. It is required, however, that to do the *eavesdropping* attack the adversary needs to be very near to an inspection point and collect the information at the exact moment that a passport is being read. The ISO 14443 standard states that the maximum distance for a passport to be read is 10cm [4], but some researchers claim that they have achieved a limit of 4 meters [6]. Most of the inspection places keep the waiting people at a distance from the reader, so eavesdropping some valid communication may be risky, and the attacker may need some special equipment to be able to eavesdrop at greater distances (like a special antenna).

On the other hand, a *skimming* attack can be performed on the waiting line, if the victim has a passport on a close spot, like its back pocket. The attacker can be the next in line and get close enough to get a reading distance on the passport and try a brute force attack. Although more easy to perform than an eavesdropping attack, each authentication attempt takes some time, because of the protocols involved and the speed of communication between the reader and the passport. We can compare this approach with brute forcing passwords on an HTML form on a web page, where the attacker needs to wait for the result before trying a new password. The advantage of eavesdropping communication is that we can make the brute force attack faster, trying to decrypt directly E_IFD, which is faster than waiting for a challenge response protocol between a reader and a passport. But as stated earlier, an eavesdropping attack is more risky and may need some special equipment.

The objective of this work, ultimately, is to verify the amount of time needed for an attacker to perform a *skimming* attack, with an attempt to brute force the keys for the BAC protocol, using a cheap reader and a portable computer. In the next section we will discuss the attack model.

5. Attack Model

Considering the ease to perform, the attack model selected is a *skimming* attack, where the attacker is in some line (ex: inspection line, security line, restaurant line, etc), close enough to the target, so it can attempt to read his or hers passport. The attacker knows that the victim has a Brazilian passport, and knows some public information about the issuing of Brazilian passports, like how many years they are valid and how the passport numbers are formatted. This information is somewhat public, so it would be easy for an attacker to find out. Any other information about the victim is unknown like age or date of issue of the passport.

The attacker also has access to some hardware like a portable reader and a portable computer, and some software like *open source* tools for reading ePassports. Using this hardware and software, the attacker can build a brute force tool to make authentication attempts on a passport, using valid combinations of MRZ data.

6. The Brazilian ePassport

As stated before, the Brazilian ePassport uses all the optional security protocols defined by ICAO, including BAC. The issuing of passports in Brazil is in the responsibility of the Brazilian Federal Police (DPF). The Brazilian passports started to include RFID chips on late 2010, and actually there are around 8.000 passports being issued every day, as stated by Federal Police Officials.

When the ePassport was first introduced, a new series of passport identification numbers was defined. The passport numbers would then all start with the characters "FD" followed by a 6 digit sequential number. On 12 Aug 2011, the "FD" series was finished and a new series, starting with "FE" was initiated.

The passports issued by Federal Police have a valid time of 5 (five) years. All other information on the Brazilian passport complies with the ICAO standard. Improvements were also made on the physical document of the passport, but these are beyond the scope of this paper.

7. BAC key entropy on brazilian passport

Using the information from the previous section and some mathematics, we can find out how many combinations of keys can a valid Brazilian passport have:

- Passport number: "FD" or "FE" followed by 6 digits: $2 * 10^6$
- Birth date year: from 00 to 99 (100 combinations)
- Birth date month: from 01 to 12 (12 combinations)
- Birth date day: from 01 to 31 (31 combinations)
- Expiry date year (considering an attack on 2011): from 11 to 16 (6 combinations)
- Expiry date month: from 01 to 12 (12 combinations)
- Expiry date day: from 01 to 31 (31 combinations)

The dates could be optimized, considering months with 30 days and February with 28 (or 29) days, but these optimizations would reduce the entropy very little, so they were not considered. Using this combination information, we can find out the resulting combinations possible for a Brazilian passport:

$$X = 2 * 10^6 * 100 * 12 * 31 * 6 * 12 * 31 \quad (1)$$

$$X =\sim 1.66 * 10^{14} \quad (2)$$

If we compare this number of combinations with a key represented in bits, we would have:

$$2^Y =\sim 1.66 * 10^{14} \quad (3)$$

$$Y =\sim \log_2 (1.66 * 10^{14}) \quad (4)$$

$$Y =\sim 47 \quad (5)$$

From these calculations we conclude that the entropy of Brazilian Passport is equivalent to a key of 47 bits. This result seems small, considering that the current standard for symmetric encryption (AES) [7] has a key size range from 128 to 256 bits, but is higher than the value described by [1]. To be certain if an attack is feasible, nevertheless, the amount of time of an unsuccessful attempt to open the passport using BAC needs to be measured.

8. Experiment on reading times for BAC

Using a standard reader, several attempts were made to read a Brazilian passport, using a popular *open source* RFID reading tool called *RFIDiot* [8], with the goal of calculating the mean time for an unsuccessful attempt to read a passport. This tool is made in *Python* language, and accepts arbitrary parameters as MRZ input.

The reader used was an OMNIKEY 5321 USB, as seen on Figure 4.

Figure 4 - OMNIKEY 5321 USB Reader

The reader was connected to a portable computer, with enough processing power (Intel 2.53Ghz Core 2 Duo CPU with 8 GB RAM). A *shell script* was made to try to read a standard Brazilian ePassport 100 times, using RFIDiot with incorrect MRZ information, and the execution time was obtained using the *time* UNIX command [9]. The results of the experiment are summarized on Table 1.

Table 2 - ePassport BAC Reading times

	Min	Max	Mean
Unsuccessful attempt	0.922s	0.952s	0.938s
Successful attempt	3.268s	3.329s	3.293s

As expected, the successful attempt times are bigger than the unsuccessful ones. This can be explained because when BAC is performed correctly, the software reads the contents of the passport, which is significantly more information than the BAC protocol itself. Also the number of steps on the BAC protocol is higher with a correct key. The mean time for an unsuccessful attempt on reading a passport is 0,938 seconds. Combining this value with the entropy of the Brazilian passport we can find how much time we need to try all the possible keys:

$$T = 2^{47} * 0.938 \quad (1)$$

$$T= 132011764077297.664 \quad (2)$$

T is stated in seconds. Converting it to years:

$$T =\sim 4244205.38 \text{ years}$$

We are calculating the mean time for a successful attempt, so we need to consider half of the previous value:

$$T_2 =\sim 2122102.69 \text{ years}$$

9. Conclusions

With the experiment and the entropy values of the Brazilian passport, we concluded that the mean time needed for finding the correct BAC key using RFIDiot, a portable computer and the supplied reader would be around 2 million years, which is unfeasible.

If the attacker had some more information about the victim, such as an estimate of the age (it can be done analyzing directly the victim or a photograph), the entropy could be further reduced and also the time needed to find the key. Other data such as the relation between the passport number and the date of expiry (can be estimated knowing the number of passports issued per day if the numbering of passports is sequential) and the reduction of possible expiry dates (if the passport is only issued on week days, for example) can be used to further reduce the entropy, but we think that it would not be enough to reach a feasible time (something between hours and minutes). In the other hand, with the further issuing of passports, new series will appear, which may increase the combinations of passport numbers.

The time needed for making an attempt is high (almost one second), so a faster reader may be needed to improve the cracking performance. This may also not be enough, because the ePassport needs to do some cryptographic operations for BAC to work, which may be slow considering that it is a passive, low power device.

Some improvements could be made to enhance the security of the Brazilian ePassport: the passport numbering scheme could be random and use alphanumeric characters. This would increase significantly the BAC key entropy. The passport could have some blocking scheme that prevents it from be read if it is not open. Also, the inspection systems could implement some jamming technology to prevent eavesdropping from malicious attackers.

The study obviously has some limitations. The tests were performed using only one specific reader. Other readers from other manufacturers may have different read times. The study also relies on an *open source* tool, which may not be optimized for speed. A specialized tool, developed to access the hardware directly may speed up the process, or even some dedicated hardware that is optimized for speed.

For future work, a practical implementation of a BAC brute force attacker is planned, optimized for trying several BAC keys. It is also planned testing different readers, including specialized ones, if available. Testing of an eavesdropping attack is also planned, verifying the real maximum distance that an attacker can record the conversation between the reader and the passport, and the time needed for a typical modern computer to find the correct key.

References

[1] Nithyanand. R., A Survey on the Evolution of Cryptographic Protocols in ePassports." University of California – Irvine, 2009.

[2] ICAO. Machine Readable Travel Documents. Avaliable in: http://mrtd.icao.int/. Accessed in 07/15/2011

[3] ICAO. Machine Readable Travel Documents. Doc 9303. Part 1 Volume 2. 6th edition. 2006.

[4] ISO/IEC 14443 Identification cards -- Contactless integrated circuit cards -- Proximity cards. 2009.

[5] A. Juels, D. Molnar and D. Wagner, Security and Privacy Issues in E-passports". IEEE Security and Privacy for Emerging Areas in Communications Networks, 2005.

[6] Y. Liu, T. Kasper, K. Lemke-Rust and C. Paar, "E-Passport: Cracking Basic Access Control Keys". OTM'07 Proceedings of the 2007 OTM confederated international conference on On the move to meaningful internet systems, Springer-Verlag Berlin, Heidelberg, 2007.

[7] J. Daemen, S. Borg and V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer-Verlag, 2002.

[8] RFIDIOT – RFID IO tools. Avaliable in: http://rfidiot.org/. Accessed in 07/15/2011.

[9] time(7) – Linux Programmer's manual. 2010. Avaliable in: http://www.kernel.org/doc/man-pages/online/pages/man7/time.7.html. Accessed in 07/15/2011.