E-Passport: Cracking Basic Access Control Keys with COPACOBANA

Yifei Liu, Timo Kasper, Kerstin Lemke-Rust and Christof Paar

Communication Security Group

Ruhr University Bochum, Germany

http://www.crypto.rub.de

Horst-Görtz Institut für IT Sicherheit

presentation at
SHARCS 2007
Vienna, Austria

September 10, 2007

Outline

1. Introduction to the E-Passport
2. The Attack Scenario
3. Basic Access Control (BAC) Protocol
4. Complexity Analysis of Key Space
5. Introduction to COPACOBANA Hardware
6. Implementation of the BAC key-search
7. Practical Results
8. Conclusion

Introduction to the E-Passport

- Standardized by the ICAO*
- Contactless chip stores data
- Basic Access Control
- Encryption of interchanged data
- Passive Authentication
- Extended Access Control

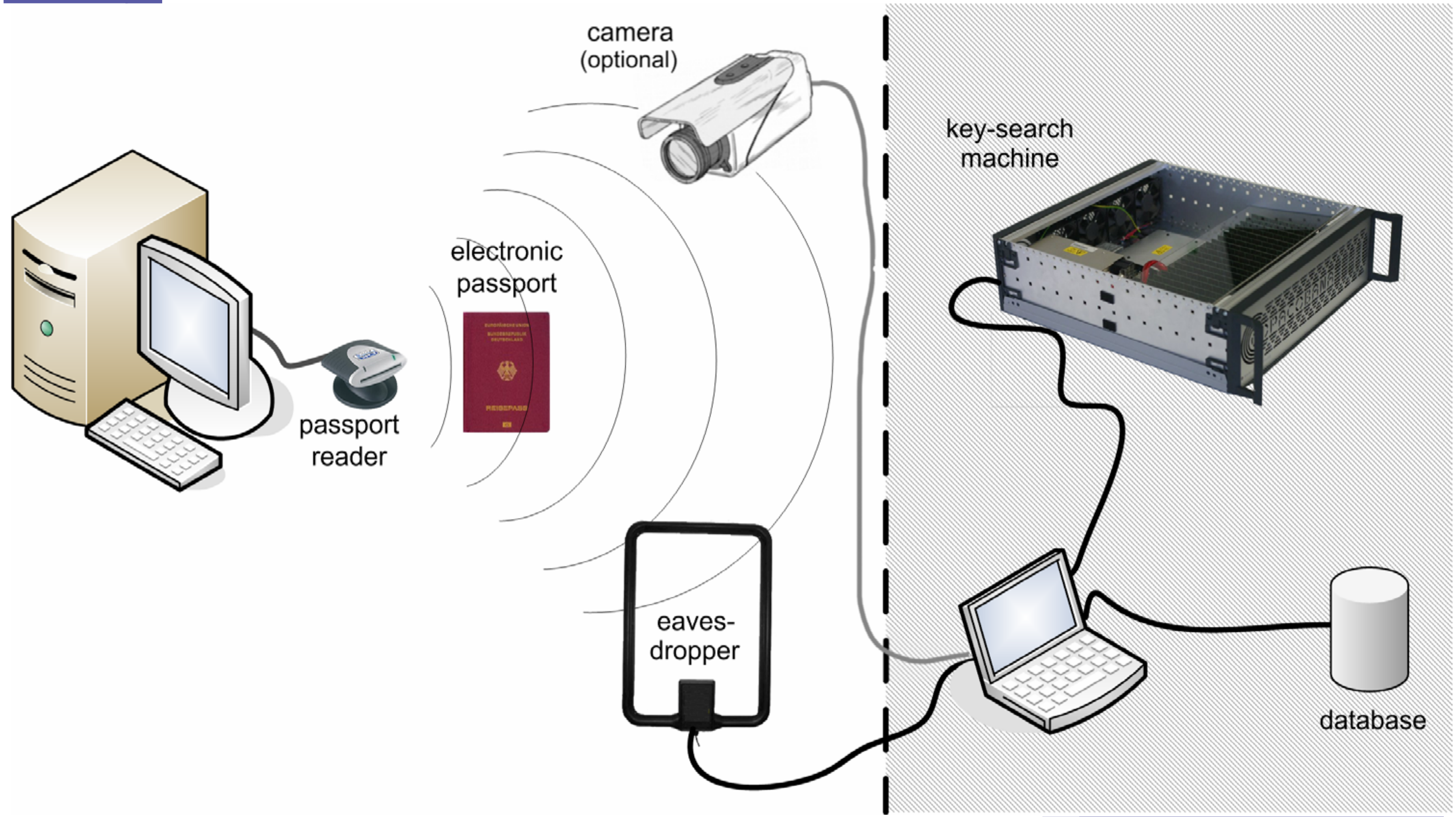
*) ICAO = International Civil Aviation Organisation

Security Mechanisms of the Epass

- Standarized by the ICAO
- Contactless chip stores data
- Basic Access Control
- Encryption of interchanged data
- Passive Authentication
- Extended Access Control

→ prevents unauthorized access to personal data **?**

The Attack Scenario



Basic Access Control (BAC)

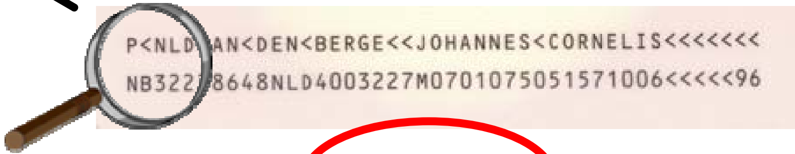
Horst-Görtz Institut für IT Sicherheit

K_{ENC}

K_{MAC}

Reader (IFD)

optically scan printed data



E-Pass (ICC)

K_{ENC}

K_{MAC}

RND_{ICC}

$RND_{ICC} \in \{0,1\}^{64}$

$RND_{IFD} \in \{0,1\}^{64}$, $K_{IFD} \in \{0,1\}^{128}$

plaintext

$E_{IFD} := ENC_{K\_ENC} (RND_{IFD} \,||\, RND_{ICC} \,||\, K_{IFD})$

ciphertext

$M_{IFD} := MAC_{K\_MAC} (E_{IFD})$

$E_{IFD} \,||\, M_{IFD}$

decrypt and verify $E_{IFD} \,||\, M_{IFD}$

plaintext

$K_{ICC} \in \{0,1\}^{128}$

ciphertext = $msb_8 (E_{ICC})$

$E_{ICC} := ENC_{K\_ENC} (RND_{ICC} \,||\, RND_{IFD} \,||\, K_{ICC})$

$M_{ICC} := MAC_{K\_MAC} (C_{ICC})$

$E_{ICC} \,||\, M_{ICC}$

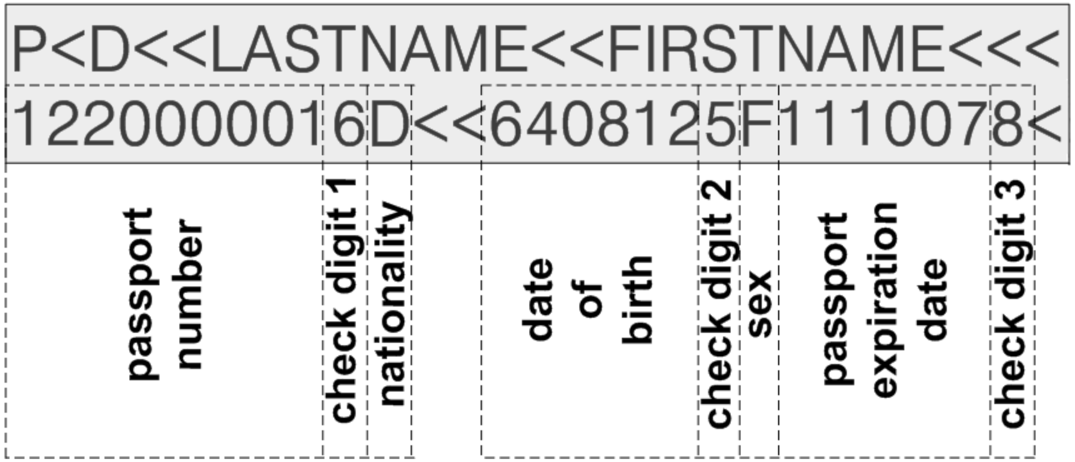
$KS_{Seed} := K_{IFD} \oplus K_{ICC}$

decrypt and verify $E_{ICC} \,||\, M_{ICC}$

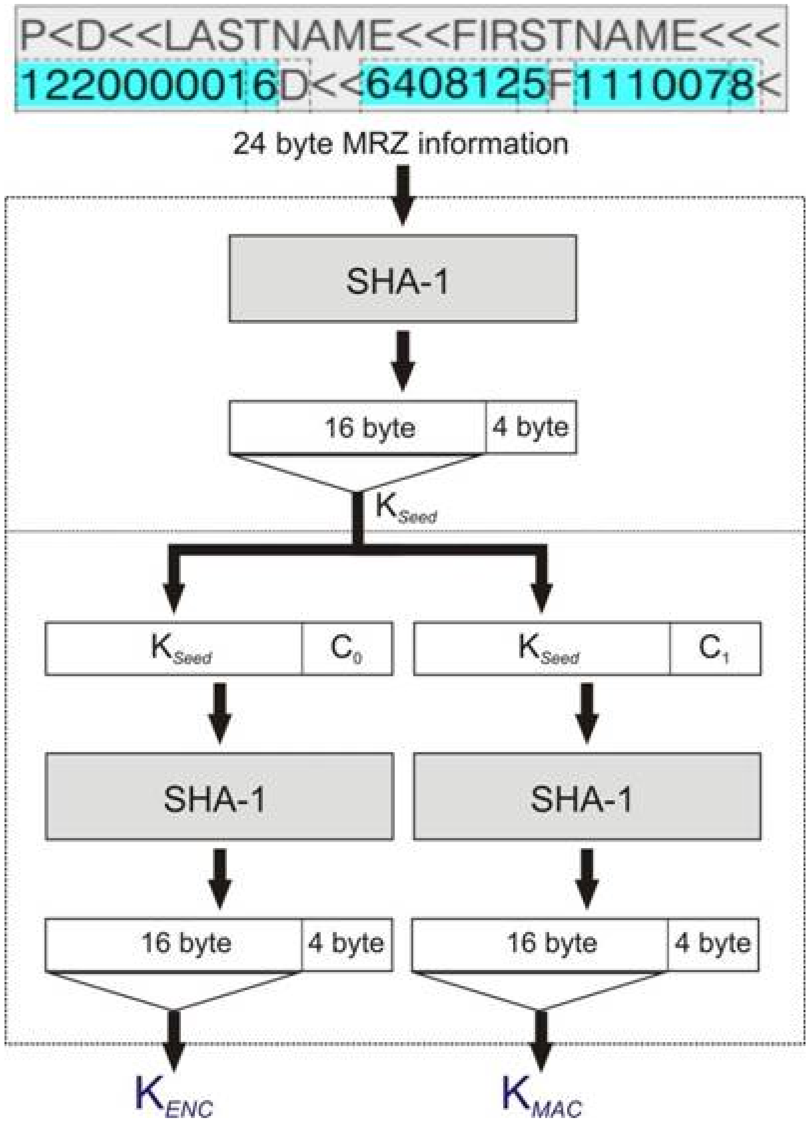
$KS_{Seed} := K_{IFD} \oplus K_{ICC}$

Derivation of BAC Keys

Information of the MRZ (Machine Readable Zone) is used for key derivation:



Derivation of BAC Keys

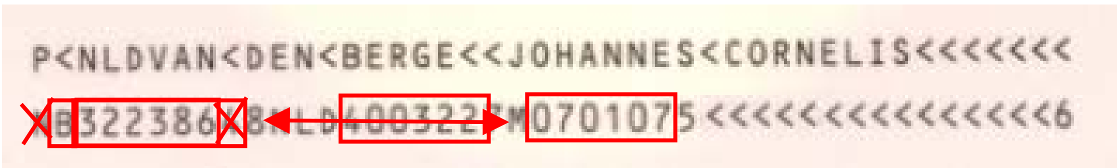


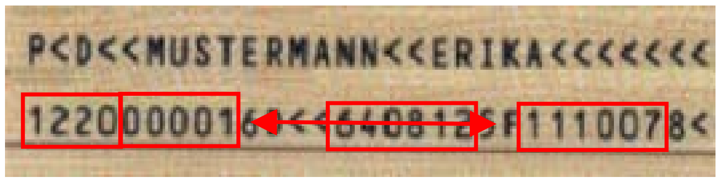
- C_0 = 00 00 00 01 for K_{ENC}
- C_1 = 00 00 00 02 for K_{MAC}

Complexity Analysis of Key Space

Special (public) parameters for issuing passports:

	The Netherlands	Germany
Start:	August 26, 2006	November 1, 2005
Validity:	5 years	10 years
Working days T_{work} until June 1, 2007	196	413
Passport owners:	approx. 9 million	approx. 20 million
Passports issued per working day:	approx. 7000	approx. 8000
Numbering:	fixed `N´, 1 alphanumeric digit, 6 numeric digits, 1 digit checksum	4 numeric digits for local authority, 5 numeric digits serial number





Complexity Analysis of Key Space

Adversary‘s knowledge on the system:

1. public knowledge
2. stochastic dependency between *passport number* and *expiry date**
3. complete database of BAC keys

Knowledge on the passport holder:

1. issuing state
2. photo of passport holder
3. date of birth
4. site of eavesdropping (only relevant for Germany)

*) in Germany: for each local authority

Complexity Analysis of Key Space

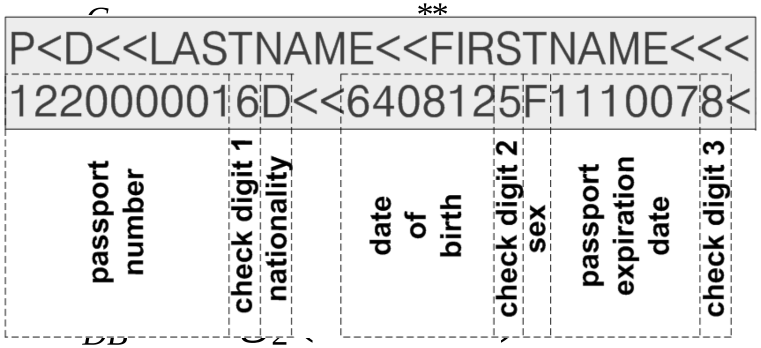
Horst-Görtz Institut für IT Sicherheit

Example Scenario:

• public knowledge, stochastic dependency between *passport number* and *expiry date**, age of passport holder with margin of 10 years, and issuing state known

Entropy for Germany

$$H^{G} = H^{G}_{PN} + H^{G}_{DB} + H^{G}_{DE}$$



$$H^{G}_{DE} \approx \delta$$

$$H^{G} \approx 33.3 + \delta$$

Entropy for the Netherlands

$$H^{NL} = H^{NL}_{PN} + H^{NL}_{DB} + H^{NL}_{DE}$$

$$H^{NL}_{PN} = \log_2(T_{work}^{**} \times 7000) \approx 20.4$$

$$H^{NL}_{DB} \approx \log_2(10 \times 365) \approx 11.8$$

$$H^{NL}_{DE} = \delta$$

$$H^{NL} \approx 32.2 + \delta$$

*) in Germany: for each local authority

**) working days since start of system until June 1, 2007

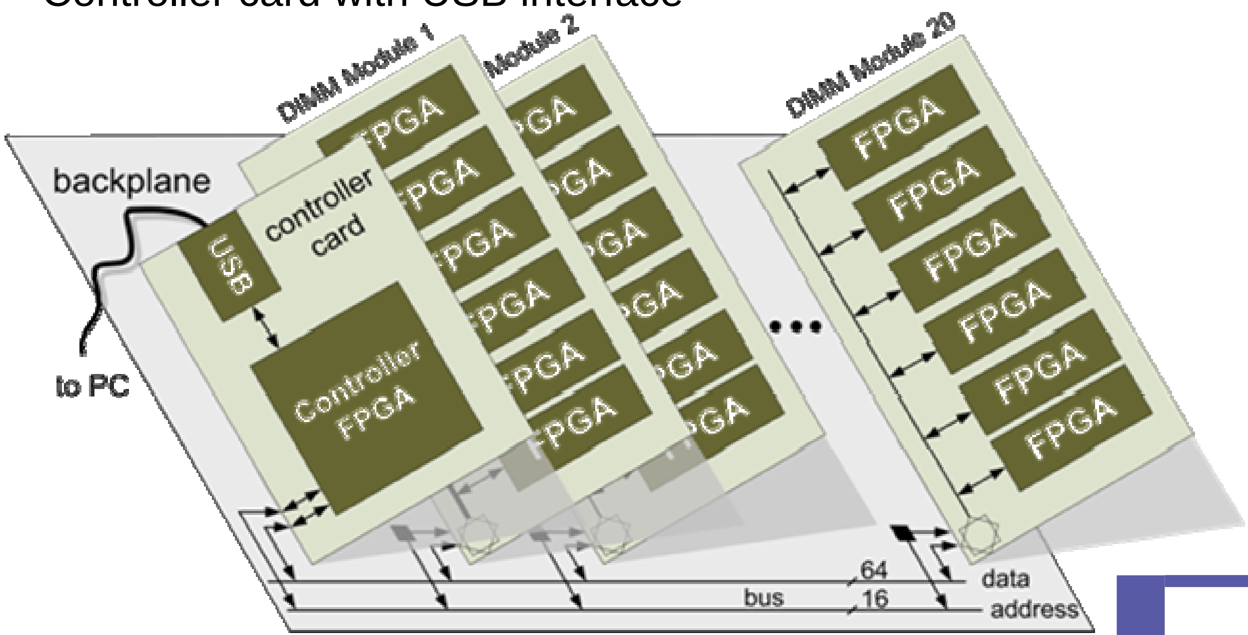
Major Flaw in the BAC Scheme

Low entropy of the BAC keys in present numbering schemes:

1. key space of the passport number:
   fixed digits, check digit, mainly numeric characters
   → could be nine alphanumeric characters
2. stochastic dependency between passport number and the expiry date
   → don‘t assign passport numbers serially
3. dependency on publicly available data (date of birth)
   → don‘t use publicly available data

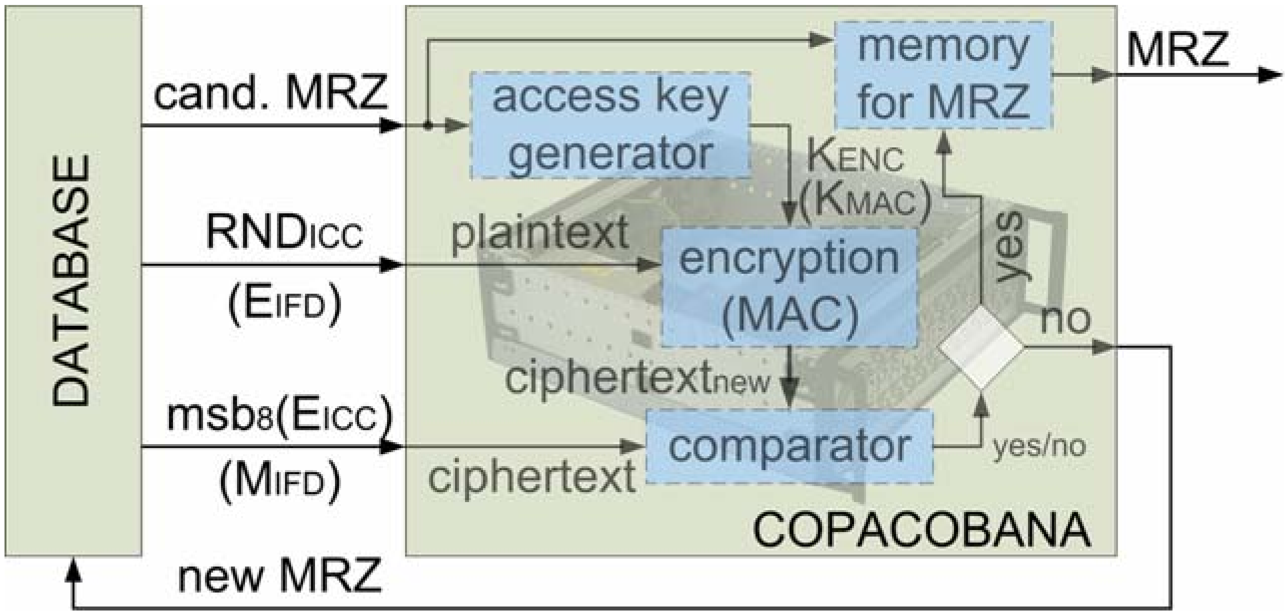
COPACOBANA: A Brief Overview

- Cost-Optimized PArallel COde Breaker
- an FPGA-based machine for DES cracking
- Parallel architecture built out of 120 Xilinx Spartan3 XC3S1000 FPGAs
- Modular design:
  - Backplane with FPGA modules (each with 6 low-cost FPGAs)
  - Controller card with USB interface



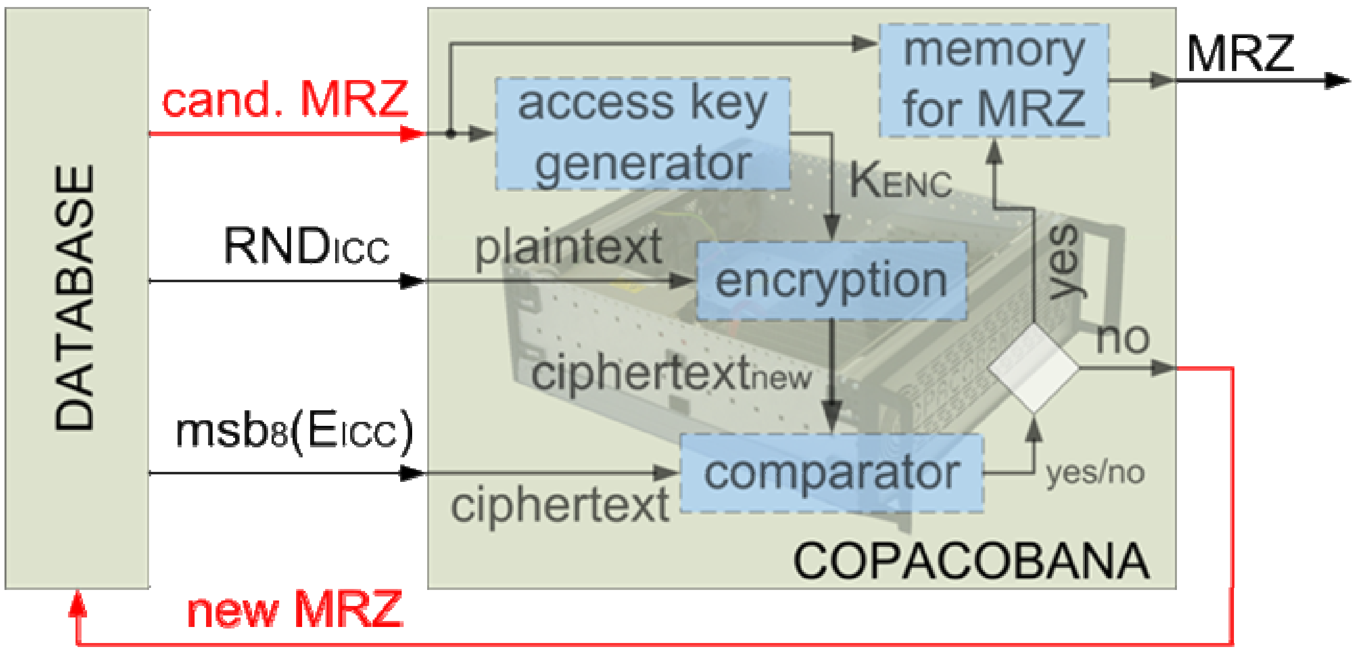
The First Thought about the Design

- Two approaches for key search:
  1. $msb_8(E_{ICC}) = E_{K\_ENC}(RND_{ICC})$
  2. $MAC_{K\_MAC}(E_{IFD}) = M_{IFD}$



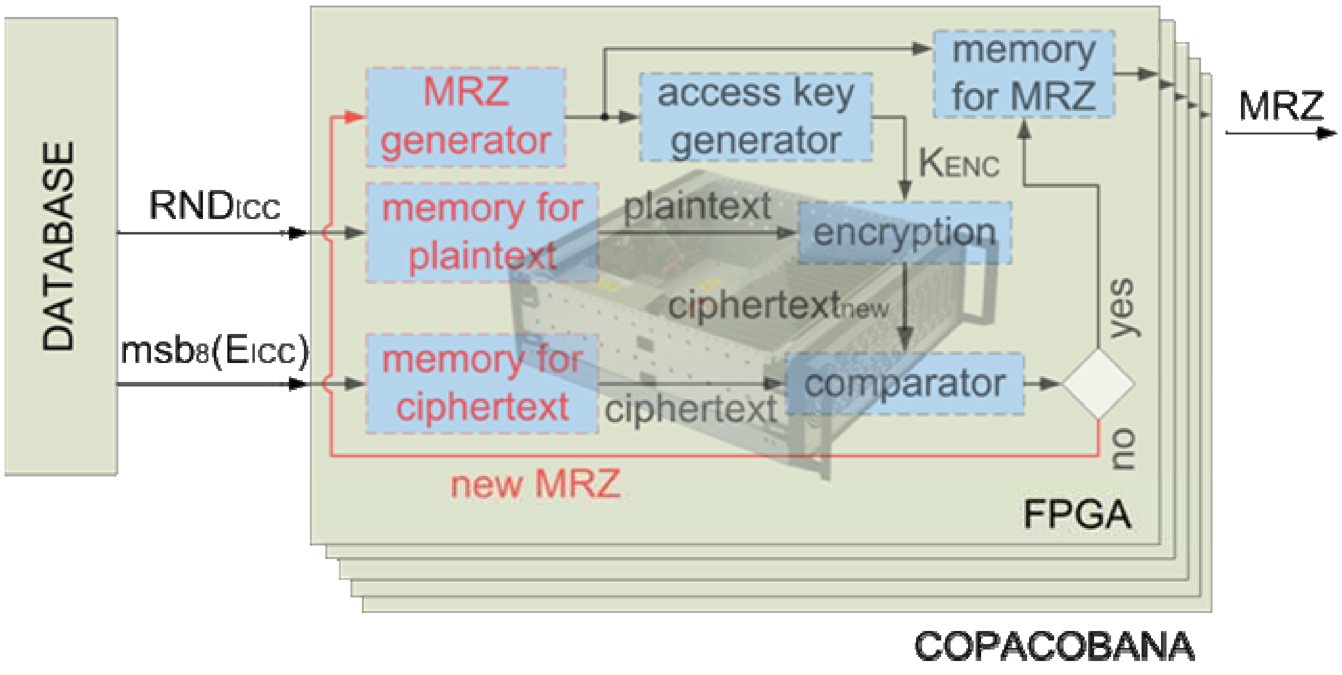
Problems and Solutions

- **Problem**: The bottleneck of the architecture of the COPACOBANA is the communication via the buses and to the Host PC.

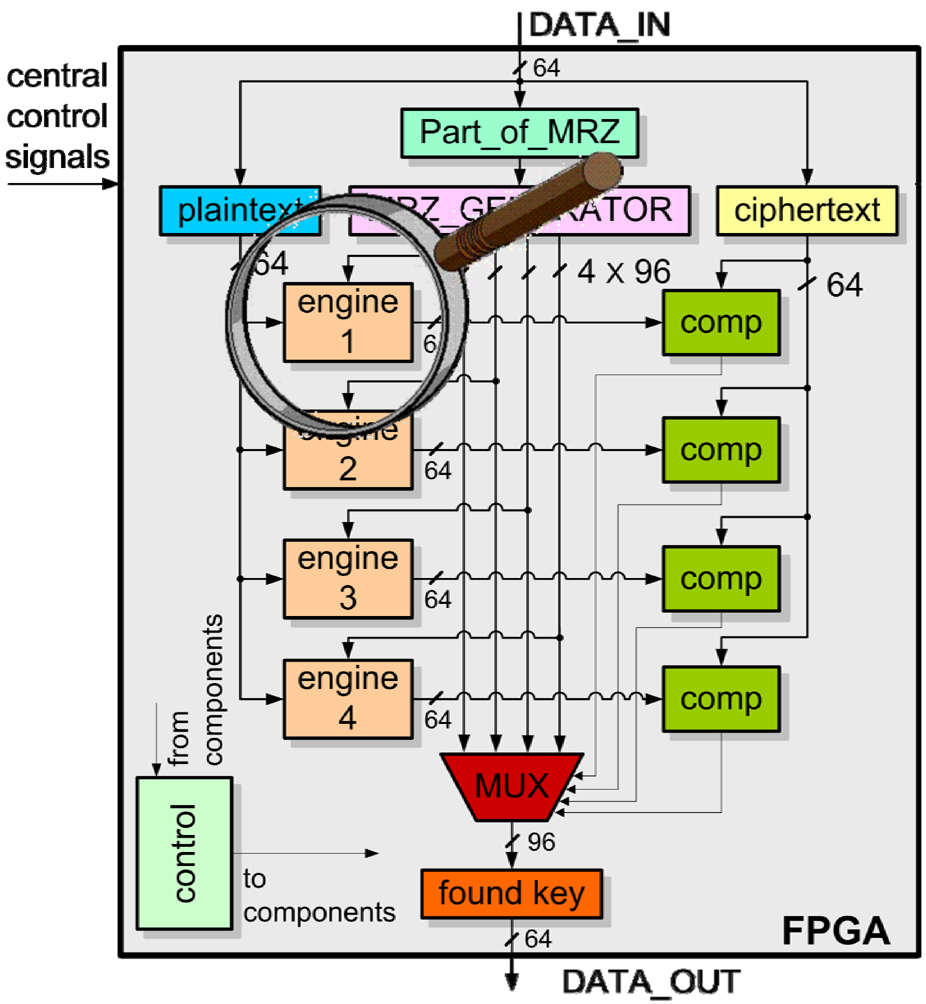


Problems and Solutions

- **Problem**: The bottleneck of the architecture of the COPACOBANA is the communication via the buses and to the Host PC
- **Solution**:
  - Every FPGA possesses a MRZ generator to support the key derivation
  - Special memories will be established to store the plaintext and ciphertext



Layout of a single FPGA



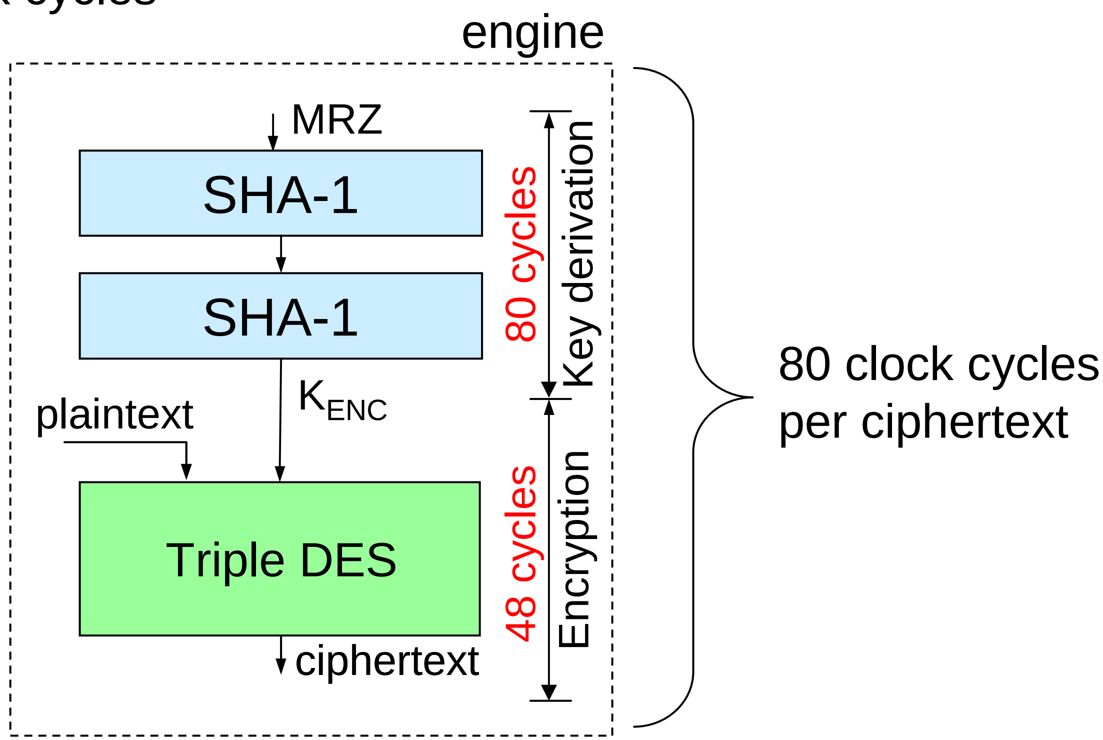
- **Part_of_MRZ**:
  - fixed for every FPGA, e.g., expiry day or birthday.
- **Plaintext**: RND_{ICC}
- **Ciphertext**: $msb_8(E_{ICC})$
- **MRZ_Generator**:
  - producing 4 MRZs/clock
- **Engine$_i$**:
  - Deriving K_{ENC}
  - Encrypting the plaintext into ciphertext
- **Comp:**
  - $Ciphertext_{new}$ = Ciphertext ?

Design of one engine

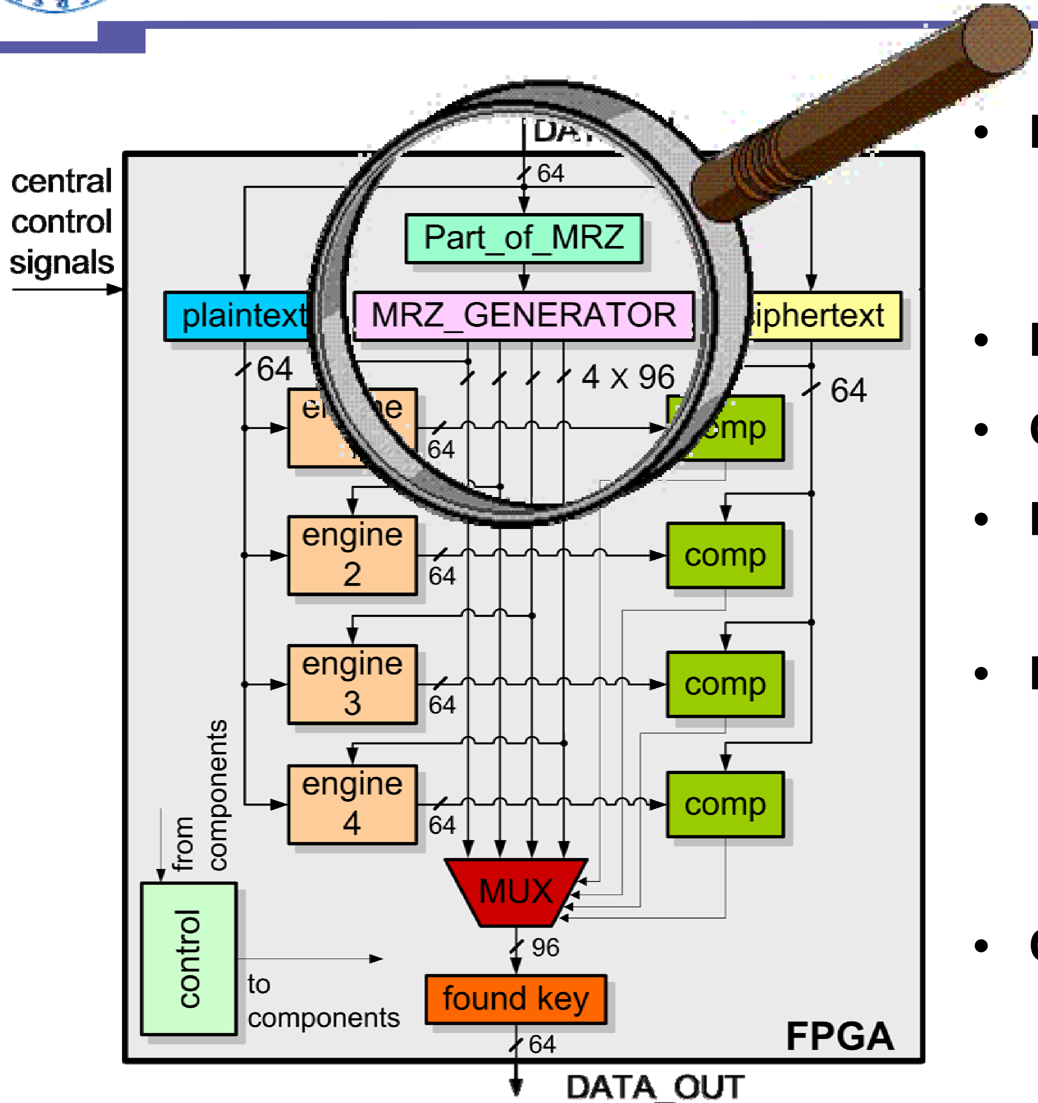
Solution:

→ Pipelined SHA-1: Needs 80 clock cycles per key candidate (SHA-1 is bottleneck)

→ 3DES needs 48 clock cycles



Layout of a single FPGA



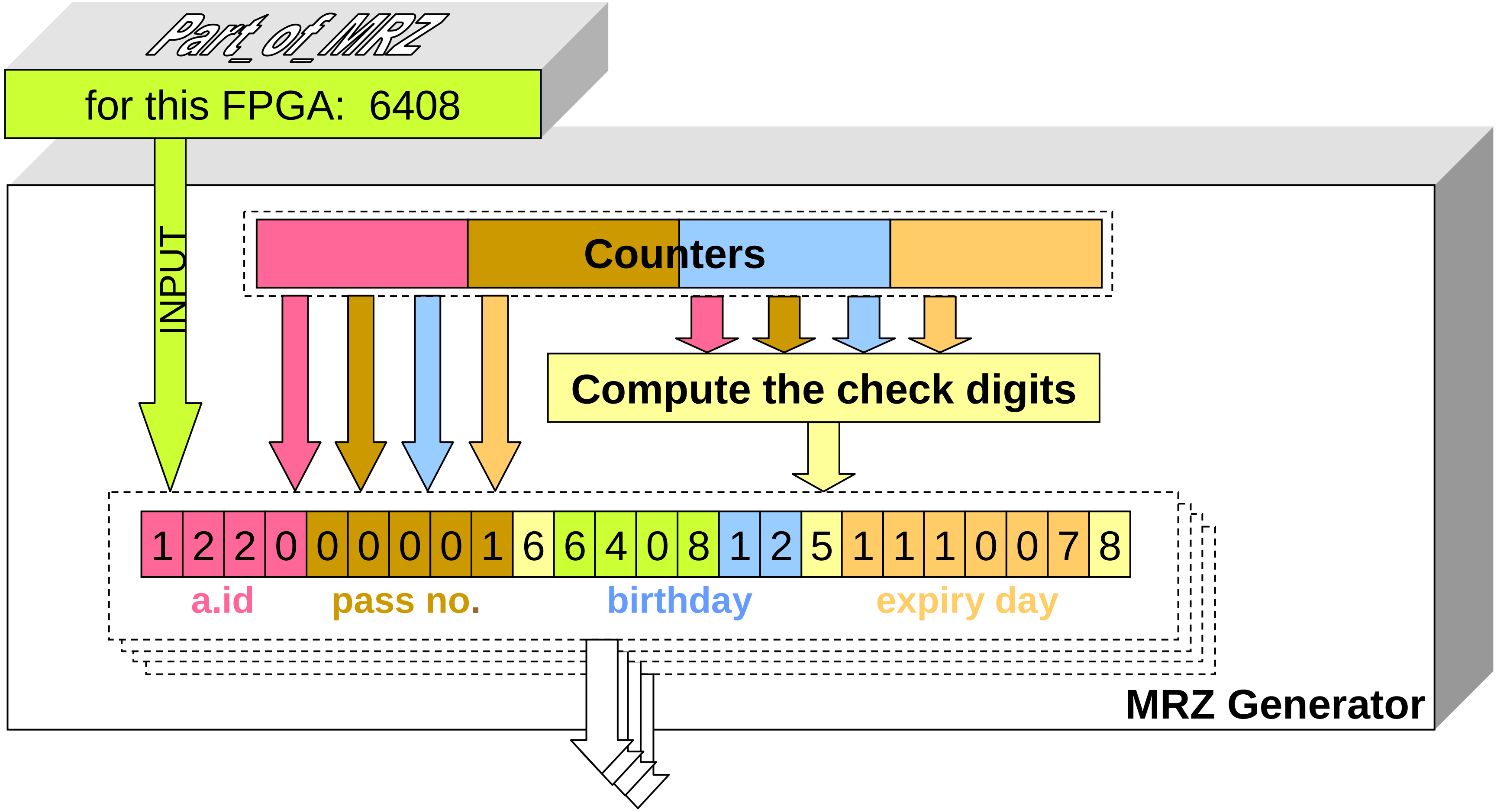
- **Part_of_MRZ**:
  - -fixed for every FPGA, e.g., expiry day or birthday.
- **Plaintext**: $\mathrm{RND}_{\mathrm{ICC}}$
- **Ciphertext**: $\mathrm{msb}_8(\mathrm{E}_{\mathrm{ICC}})$
- **MRZ_Generator**:
  - -producing 4 MRZs/clock
- **Engine$_i$**:
  - -Deriving K_{ENC}
  - -Encrypting the plaintext into ciphertext
- **Comp:**
  - -$\mathrm{Ciphertext}_{new}$ = Ciphertext ?

Design of the MRZ Generator

- **They are the only components depending on the concrete attack scenario (adversary, knowledge about passport holder,…)**
- **Part_of_MRZ**: fixed for every FPGA; but how?
  - One Idea: Age estimate: 10 years = 120 months = 120 FPGAs

  →Fixed part of MRZ is year and month of birth.
- **MRZ Generator**
  - Input: Part_of_MRZ
  - Output: Subspace of MRZ information

Implementation for Scenarios



- size of Part_of_MRZ field depends on application scenario
- as does the MRZ Generator

Practical Results

Efficiency of the implementation

Clock rate	40 MHz	
Speed of the Key search	1 FPGA	2 Mio. Keys/second $\approx 2^{20.93}$ Keys/second
	120 FPGAs	240 Mio. Keys/second $\approx 2^{27.84}$ Keys/second

		Germany	The Netherlands
Scenario:	Total amount of MRZ candidates	$1.06 * 10^{10}$	$4.9 * 10^{9}$
	Average time to find the MRZ	22 second	10.3 second

- public knowledge, stochastic dependency between *passport number* and *expiry date,* age of passport holder with margin of 10 years, and issuing state known

Conclusion

- Scenario for eavesdropping attacking BAC keys introduced
- Two approaches for two-way and one-way communication
- Complexity Analysis of BAC key space
  → Entropy of present schemes is too low
- Fast hardware implementation of the BAC key search
- Throughput: $2^{27.8}$ = 240 million BAC keys per second on COPACOBANA
- 2^{35} key candidates require 2 minutes and 23 seconds

➔ Key search machines are a real threat for privacy and security of electronic passport holders.

Thanks for your attention!

{yliu,tkasper,lemke,cpaar}@crypto.rub.de

presentation at
SHARCS 2007
Vienna, Austria

September 10, 2007