# Efficient and Secure ECC on Embedded Devices

André Weimerskirch, ECC 2005, Copenhagen

escrypt
Embedded Security

escrypt GmbH
Lise-Meitner-Allee 4
44801 Bochum

t: +49(0)234 43 870 209
f: +49(0)234 43 870 211

# Acknowledgements

**Special thanks to**

- Christof Paar
- Sandeep Kumar
- Marko Wolf

# Contents



escrypt
Embedded Security

# Introduction

- ECC good choice for constrained (embedded) devices
- Plenty of literature about ECC arithmetic and side-channel resistance available
- Literature usually for PCs, stand-alone systems and smart cards
- Requirements for embedded devices often different

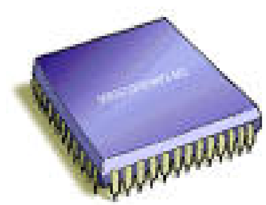
# Contents

# What are Embedded Systems?

“A computer that doesn‘t look like a computer”, or

a “processor hidden in a product”

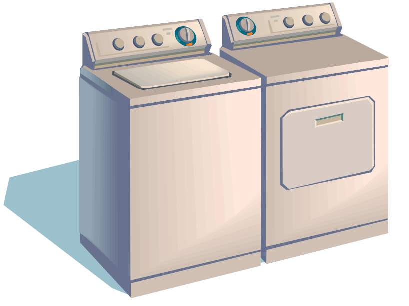
+



= Embedded System

# Characteristics of Embedded Systems

Single purpose device

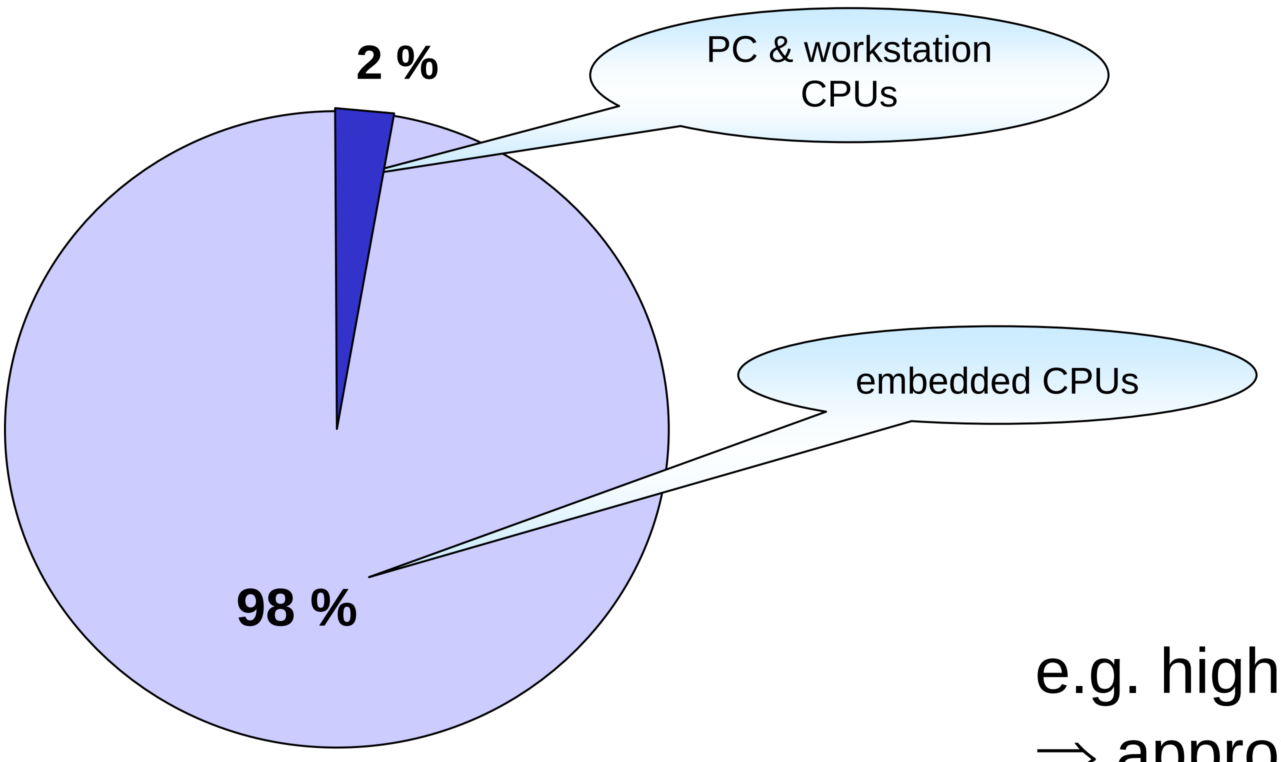


- Not general purpose like PC
- Interacts with the world
- No (or primitive) user interface

# Software and Embedded Systems

- Software is important
  - Standard HW micro-controller
  - Adds „life“ to product
  - Can give different characteristics
  - Often relatively low-level languages (assembly, C)
- **Often no** SW updates (or inconvenient to perform)
  - code in ROM
  - lack of  online connection (washing machine, digger)
  - Memory / code size constrains

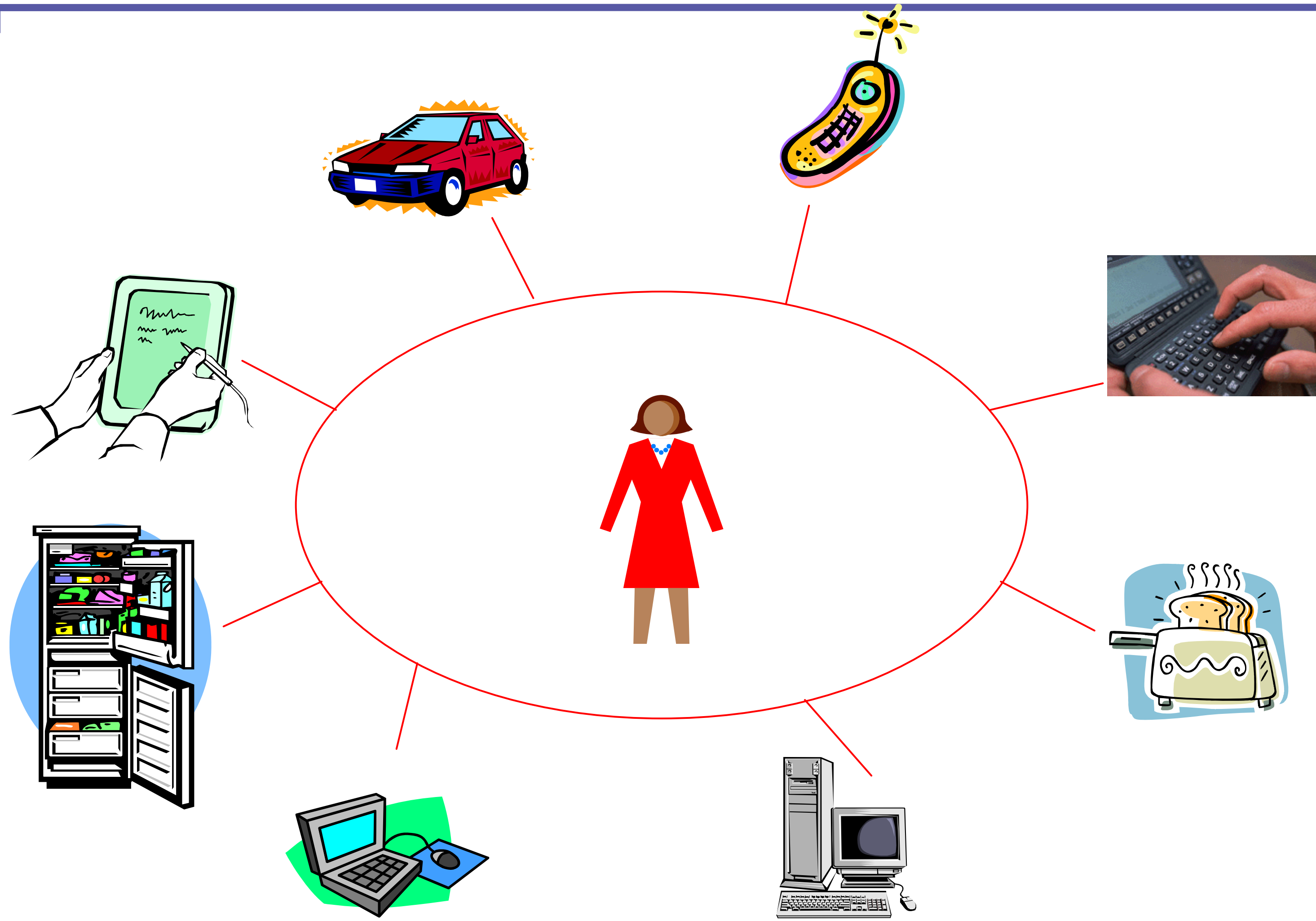
# Are Embedded Systems really Important?




# CPUs sold in 2000

e.g. high-end BMW
⇒ approx. 80 CPUs

# Brave New Pervasive World



# Future


Smart Dust

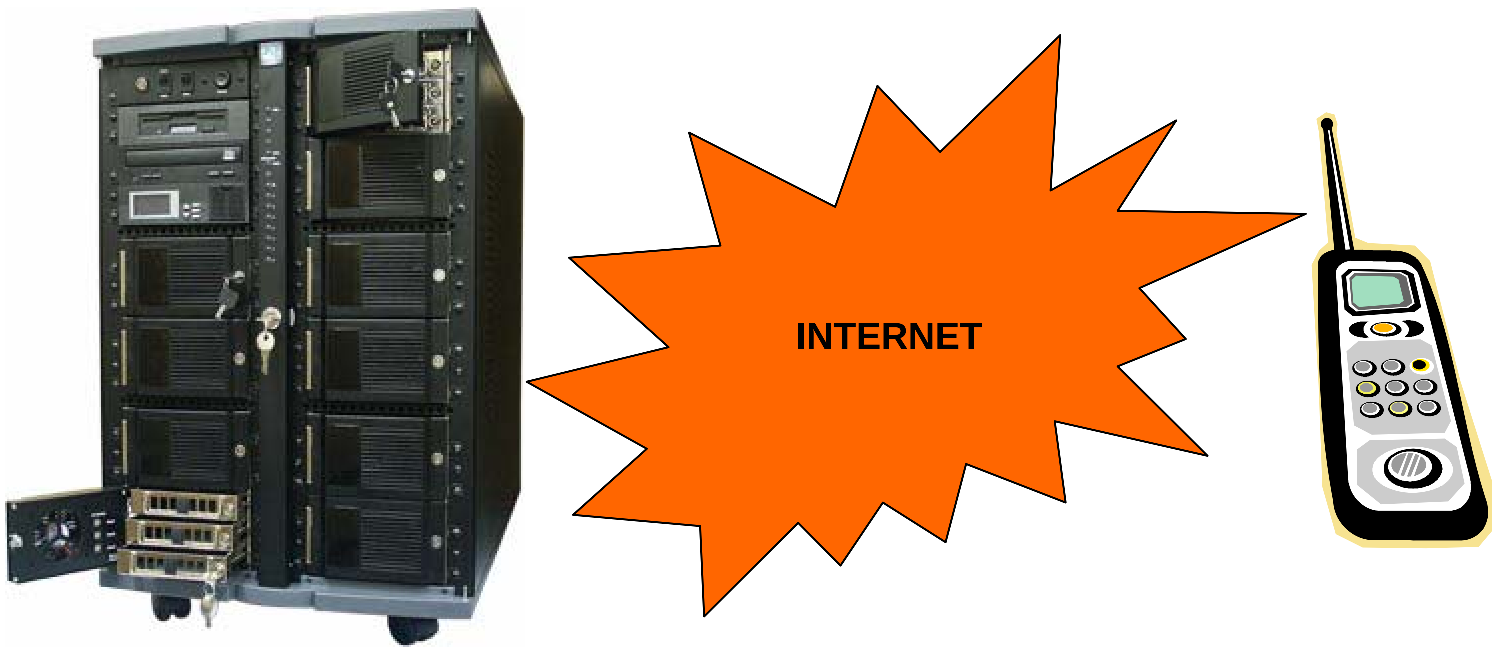
- Massively distributed microcontrollers
  - Wireless communication
  - Sensors
- Inexpensive enough to deploy by the hundreds

# Contents

# Applications of Embedded Devices

- Standard applications
  - Smart card applications
  - Inherent security applications
    - Identification
    - Payment
- Further applications
  - Applications where security is just a part of the embedded system
  - Applications where security is enabler for business models

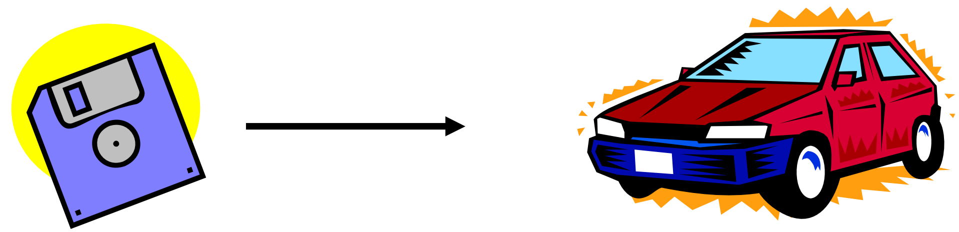
# Embedded SSL




- Provide authenticity and confidentiality

# Secure Download (Flashing)

"flashing" of embedded software: load program code into embedded device

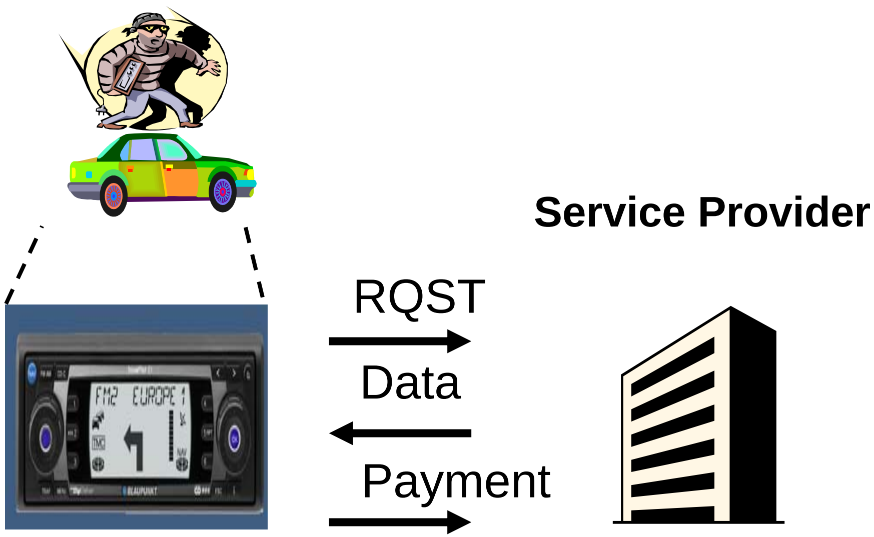


- Update software
- customization of cars
  - new products (SW tuning kits)
  - new business models ("20 HP more for the weekend for €19.99")

*But*: Unauthorized flashing poses major risk for safety and profits

⇒ Need authenticity!

# Digital Rights Management System




## Navigation Data

- Data on demand
  - e.g., two weeks of an Italy map
- Enables new business models
- But: user tries to break the rules

*Lessons learned*: Cryptographic protection (e.g. digital signature) is enabling technology for new business models

# Component Identification

- Car and component „recognize" each other

⇒ **Component is „chained" to car**

- **Security Objectives**
  - Protection of faked parts (Innovation protection, safety)
  - Theft protection
  - Protection against manipulation

⇒ Cryptography has real-world impact!

# Future Applications with Security Need

- Networked devices (GSM, 3G, WiFi):
  - Access control
  - Security & integrity of communication
  - Anonymity (e.g., privacy of location)
- Protection of digital content (navigation data, music, video, …)
- Software updates of all kind (via flashing, online, …)
- Theft protection
- Legal applications (speed control, warranty, …)
- …

# Contents

1. Introduction
2. Embedded Systems
3. Next Generation Crypto Applications
4. **ECC**
    - **Basics**
    - Requirements
    - Implementation
    - Comparison to RSA
5. Conclusions

# Basics

- Core operation
  - point multiplication k*P
- Security
  - Based on discrete logarithm problem (DLP) which is believed to be secure
- Main ECC schemes
  - Signature scheme (e.g. ECDSA)
  - Key agreement (e.g. ECDH, MQV)
- Benefits (mainly compared to RSA)
  - Fast signature generation
  - Small key sizes / small signature size
- But
  - Slow signature verification

# Assumptions

- Implementation
  - We consider only *software implementations* here
- Constrained resources
  - Memory: code size / RAM
  - CPU power
  - Power consumption
- Low-cost device with no or little physical security
  - No cryptographic co-processor
- Long life span of device (> 15 years)

# CPU Classification

## Rough classification of embedded processors

| *Class* | *speed : high-end Intel* |
|---|---|
| **Class 0**: few 1000 gates | ? |
| ***Class 1**: 8 bit $\mu$P, $\leq$10MHz* | $\approx$ *1: $10^3$* |
| ***Class 2**: 16 bit $\mu$P, $\leq$50MHz* | $\approx$ *1: $10^2$* |
| ***Class 3**: 32 bit $\mu$P, $\leq$100MHz* | $\approx$ *1: 10* |

# CPU Classification

***Class 1**: 8 bit $\mu$P, $\leq$ 10MHz*

- Symmetric algorithms possible at low data rates
- Asymmetric difficulty without co-processor

***Class 3**: 32 bit $\mu$P, $\leq$ 100MHz*

- Full range possible

*Note:* CPU might allow crypto application but code size might still be too large!

# Crypto Engineering

**Definition**

1. Efficient and
2. Secure

implementation

**Literature**

- Often, only speed matters *or* secure implementation
- Code size and cost rarely matter

# Contents

# Resources – Code and RAM Size

- Code / RAM size $\Leftrightarrow$ hardware cost
- Cryptographic methods often included afterwards $\Rightarrow$ minimal free memory left
- Low code and RAM size contradict fast running times
  - Use of pre-computed points
  - Fast implementation techniques

# Resources – Running Time

- Depends on application
  - Running time sometimes not important
    - Secure download at repair shop
  - Sometimes crucial
    - User interaction: < 1 sec.
    - Vehicle's engine start: < 50 ms

# Physical Security and Standards

## Physical Security

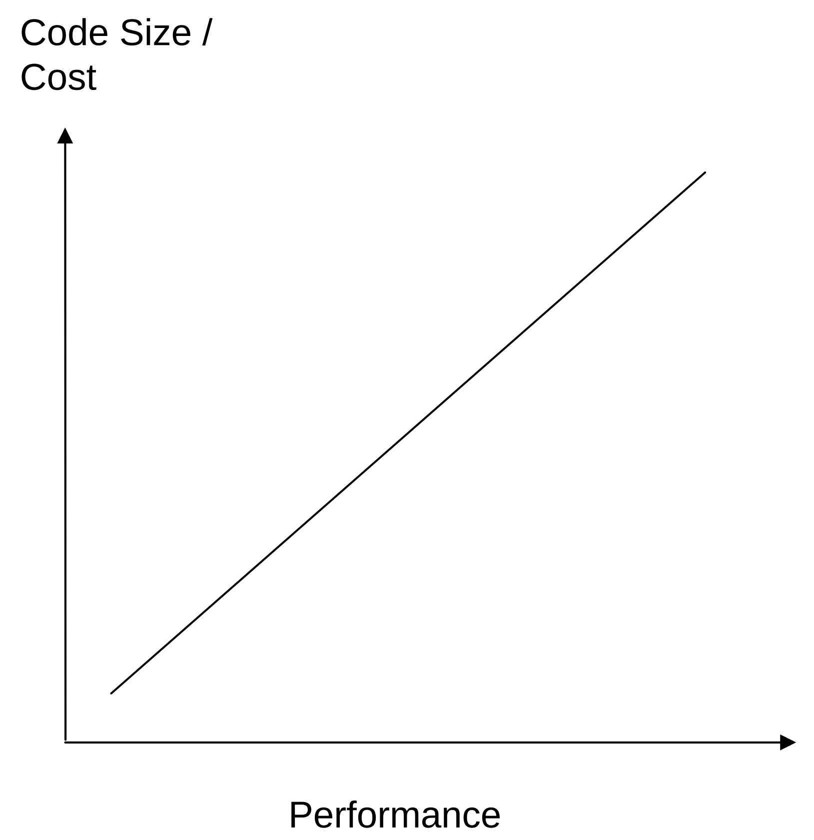
- Secure Implementation
  - Resistance to side-channel attacks
  - Flawless implementation
- Tamper resistant or evident

## Standards

- Often standardized curves such as NIST recommended curves are requested

# Contradictions

- large code /RAM size ⇔ high cost
- cryptographic processor ⇔ cost
- small code size ⇔ slow execution time
- side-channel resistance ⇔ slower execution time / larger code size
- Standard HW ⇔ no tamper resistance



# So what are the Requirements?

- Small code size: < 2-3 KB
- Small RAM size: < 200 Bytes
- Running time: < 1 sec.
- Standard curves (NIST)
- Tamper resistant implementation on non tamper resistant hardware ☺

# Examples

- Car industry
  - minimal code and RAM size
  - Often 16 bit micro-controller (sometimes even 8 bit)
  - E.g., secure downloading (ECDSA / RSA signature verification)

- Infotainment
  - e.g., vehicle navigation system
  - Optimized running times
  - Side-channel resistance
  - Usually 32-bit micro-controller
  - Code size negligible

# Examples

- Smart Card
  - Optimized running times
  - Small code size
  - Side-channel resistance
  - 8-bit micro-controller
  - *Note*: Co-Processor might be cheaper than additional EEPROM memory

# Contents

# Implementation - Literature

- Running time
  - Plenty of literature about fast ECC in SW
- Only little about small code size in SW
  - Straight forward engineering task up to a certain point
  - Non-trivial with further objectives
- Plenty about side-channel resistance in SW
  - at least SPA and DPA, but

  $\Rightarrow$ Some side-channels not well understood yet (e.g., RF)

  $\Rightarrow$ Often vulnerable to DFA
  - Only little in combination with other objectives such as running time and code size

# Implementation Overview

| | Minimal code size | Negligible code size |
|---|---|---|
| Speed optimized | ? / ? | ? / x |
| Speed not optimized | x / x | x / x |

Side-channel resistance / no resistance

# Implementation

1. Speed optimized / minimal code size / side channel resistance
   - Use hardware cryptographic co-processor
   - SW solution always trade-off
2. Speed optimized / standard code size / side channel resistance
   - Combine window methods / pre-computed tables with side-channel resistant methods
   - < 1 sec., 7 KB code-size

# Implementation

3. Speed not optimized / minimal code size / with or without side-channel resistance
   - High-level engineer's task rather than cryptographer's
   - Becomes more difficult with side-channel resistance
   - < 2-3 KB code-size
   - < 1 sec. running time

# Performance – Low Cost Controller

- Optimized speed / side-channel resistance / no cryptographic processor (8051 @ 33 MHz):
  - ECDSA signature generation:
    - 500 ms (ECC 160)
    - 750 ms (ECC 192)
  - Code Size: 7 KB (incl. pre-computed points)
  - Side-channel resistance
  - Incorporates pre-computed points with side-channel resistance

# Performance – Vehicle Platform

- Minimum code size / no cryptographic processor (16 and 32-bit, e.g. C166 and ARM7 @ 40 MHz):
  - ECDSA signature generation:
    - 300 ms / 1 sec. (ECC 160)
  - Code Size: 2-3 KB (no pre-computed points)
  - Side-channel resistance

escrypt
Embedded Security

# Tamper Resistance

- Requires special hardware
- But can raise engineering effort for mounting an attack
- Introduce some kind of obscurity (although against any schoolbook)
  - e.g., secret curve parameters, base point
  - Must follow same generation principles
  - Is not comparable to raising cryptographic security level!!!
- ***But*** weaknesses usually induced by implementation, not by cryptographic primitives

$\Rightarrow$ Incorporate in security design

  - Successful attack to single device must not scale
  - Attack should require hardware modifications

# Long Life Span

- Consider using 192 bit ECC instead of 160
  - Although embedded tends to have smaller key sizes
- Include update mechanism in system design
  - It is hard to ensure a secure system for a decade at industrial level
    - e.g., hardware might be vulnerable to new attacks
  - But it might be possible to correct it
    - Include update mechanism

# Contents

# Comparison

- ECC almost always wins
- RSA wins for
  - Signature verification
  - Code size

| | ECC | RSA |
|---|---|---|
| Signature verification | ↓ | ↑ |
| Signature generation | ↑ | ↓ |
| Key Agreement | ↑ | ↓ |
| Key / Signature Size | ↑ | ↓ |
| Code Size | ↓ | ↑ |

# Contents

1. Introduction
2. Embedded Systems
3. Next Generation Crypto Applications
4. ECC
    - Basics
    - Requirements
    - Implementation
    - Comparison to other Schemes
5. **Conclusions**

# Conclusions

- ECC has special requirements on embedded devices
- ECC for embedded devices enables new applications
  - e.g., new business models
- Secure and efficient implementation hard to achieve
  - Several competing objectives
  - Several side-channel issues not well understood yet
- ECC not always best choice
- But ECC works fine even on smallest embedded devices

# Thank you for your attention!

**André Weimerskirch**

escrypt GmbH

aweimerskirch@escrypt.com

escrypt
Embedded Security

escrypt GmbH
Lise-Meitner-Allee 4
44801 Bochum

t: +49(0)234 43 870 209
f: +49(0)234 43 870 211