Tool-supported Composition of Software Modules for Safe and Secure Wireless Sensor Networks

Steffen Peter

IHP
Im Technologiepark 25
15236 Frankfurt (Oder)
Germany
Outline

- Introduction IHP & Related Projects
- Motivation
- Configuration Tool Approach
- Module Selection
- Security Assessment
- Conclusion
New Institute & Cleanroom
IHP in a Nutshell

The Institute

- Founded in 1991; successor institution to the former institute of the East German Academy with extensive experience in silicon microelectronics
- 200 employees from 16 countries
- Member of the Gottfried Wilhelm Leibniz Society (WGL)

Mission

- Strengthen the competitive position of the German and European microelectronic and communication research
- Act as an innovation center, leading research results towards prototypes
- Enhance the attractiveness of the region as location for high technology

Facilities

- Complete innovation chain from materials to systems, including class-1 cleanroom, 0.13 µm capable pilotline

Competencies

- Systems for wireless communication
- RF circuit design
- Extension of silicon CMOS technologies
- Materials for microelectronic technology

Strategy

- Create value through innovation
- Focus on solutions for wireless & broadband communications
- Development of forward-looking technologies and system-level prototypes
- Strategic partnerships
UbiSec&Sens Project (2006-2008) Overview

Application areas

- **Agriculture** weak security
- **Automobile**
- **Homeland** strong security

Middleware

- API
- Query language
- Selection and config. of security means

Basic & complex services

- flexible routing in-networking
- secure routing
- concealed data aggreg.
- Key pre-distribution
- sec. distr. data storage
- discrepancy monitoring
- plausible and resilient
- authentication re-recognition

Efficient impl. of crypto means.

UbiSec&Sens Toolbox
UbiSec&Sens Vineyard Scenario (2008)

The setting
- Commercially run vineyard “Weingut Georg Naegele” in Neustadt, Germany
- Deployment in operational part of the vineyard, no special arrangements
- Requirements collected together with the proprietors

Key requirements
- Diverse sensing capabilities together with geographic coverage
- Resilience to faulty data and component failures
- Storage of data to the network as well as remote access either synchronously or asynchronously
- Long deployment times; self-organization

Encountered problems: (like in LOFAR-Argo Project)
- Insufficient software engineering
- Incompatible modules
- Unforeseen side effects between modules

Additional problems:
- Harvester does not like sensor nodes
- Metal container influence wireless communication
Sensor nodes
Realflex project (2008-2010)

Water works  Biogas facility  Roboter cell

Standards, existing architektur  large distance, local intelligence  Small latency, dependability

wireless architecture for industrial automation
WSAN4CIP project (2009-2011)

Scheme of water mains

ca. 1800m

ca. 5800m

ca. 4800m

ca. 2500m
Projekt: FeuerWhere
Timeline of Sensor Network Research

2000 -
• Studies, basic research, small demonstrations

2006 -
• Prototypical real world applications
  - Developed by specialized expert groups
  - Very expensive (several person years)

???
• Broad spectrum of reusable application frameworks
  - Programmable by domain experts
  - Easy and cheap software and hardware setup
Motivation

Huge future market for Wireless Sensor Networks:
• environmental and structural health monitoring
• military and homeland security applications
• control in offices and private households
→ Strong security, safety and privacy requirements

Problem: How to realize and manage this security?

Sensor Nodes have severely scarce resources:
• Energy
• Processing power
• Memory

Problem: Trade-Off: Security ↔ Performance
Motivation (2)

There are a lot of solutions for isolated problems:
- Secure and dependable routing
- Good encryption
- Dependable and secure distributed data storage

Problem: How to exploit all good solutions in one system?

Implementation of sensor nodes is a lot manual work
- Error-prone
- Not objective
- Needs a lot of time → Expensive

Problem: Secure WSN solutions have to be economically reasonable
How to design secure and dependable WSNs?

- **Today:**
  - develop actual application first
  - Attach some security stuff later

  → no satisfying security & safety
  → Lot of processing overhead

- **Required:**
  - Development of integrated safe and secure application

  → Less protocol overhead
  → But development overhead
Example for possibly wrong design

Design unprotected application

Secure implementation with encryption in MAC
Goals / Vision

Bunch of Solutions

Application

Sensor node HW

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

 Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA

Req.

Request

Configuration and Management Module

1. Req. vs features of modules
2. Interoperability of modules
3. Security of combination

Tailor made security architecture

System

Application

Sec. robust data storage
CDA_alg2
Resilient data aggregation alg_1
Sec. rout_1
Sec. MAC_1
Secure local
AES
ECC
OS

Sensor node HW

AES

ECC

RSA

DES

Sec. random generator
Sec. localization
Sec. robust data storage
Sec. MAC_2
Sec. MAC_1
Sec. routing_1
Sec. routing_2
Resilient data aggregation alg_1
CDA_alg2
CDA_alg3
Resilient data aggregation alg_2
TEA
The configKIT Approach

- **Hardware Repository (XML)**
- **Software Repository (XML)**

**Module Selection**
- Resolve Dependencies
- Semantic Evaluation
- Estimation of
  - Code size
  - Security properties
  - Energy assessment

**List of potential solutions satisfying requirements**

**Blueprint for system integration**

**User Interface, Application designer**
- Hardware selection
- Application description

**refine**
configKIT for Application Designers

- Selection of hardware
- Selection of required functions
- Definition of security properties
- Each change of inputs immediately updates the result
  → Fast and easy refinement process
- Proposed software configuration
- Including assessment of security level
- Prediction of footprint

The configKIT Approach – Setup of Repositories

Module Selection
- Resolve Dependencies
- Semantic Evaluation
- Estimation of
  - Code size
  - Security properties
  - Energy assessment

List of potential solutions satisfying requirements

Blueprint for system integration

User Interface, Application designer

Hardware selection
Application description

refine

Hardware Repository (XML)
Software Repository (XML)
Challenge (1)

How to present the bunch of modules in an objective form?
Structure of Software Repository

Nodes:
1. Actual Module
2. Interfaces they use and provide
3. Abstract Modules and Interfaces

Connections
4. Uses
5. Provides
6. isA

- AsymCrypt
- ECCIf
- Asymmetric Cryptography
- ECC
- ECCArith
Structure of Software Repository

Other modules can use interfaces without knowing them

User can select functions in informal way

Key Exchange

AsymCrypt

ECC-IF

ECC

ECC Arith

Asymmetric Cryptography

RSA
Setup of Software Repository

- structural view
- dependency graph
- selection of sub-components

Setting of parameters, dependencies, requirements

Re-usable XML, part of software repository
Example Module Description

<SoftwareComponentType
    Description="Elliptic Curve Digital Signature Algorithm"
    IsStatic="true" Name="ECDSA" Version="0.1">
    <EnergyConsumption>0</EnergyConsumption>
    <CodeMemorySize>1468</CodeMemorySize>
    <DataMemorySize>540</DataMemorySize>
    <PersistentMemorySize>0</PersistentMemorySize>
    <Provides>
        <SoftwareInterfaceType Alias="ECDSA" SoftwareInterfaceTypeId=""/>
    </Provides>
    <Uses>
        <SoftwareInterfaceType Alias="ECC" SoftwareInterfaceTypeId=""/>
    </Uses>
    <SecurityParameter Name="Integrity" Value="4"/>
    <SecurityParameter Name="Concealment" Value="2"/>
    <SecurityParameter Name="Robustness" Value="1"/>
</SoftwareComponentType>
Setup of Hardware Repository

- Structural view/selection of sub-components
- Parametrisation
- Re-usable XML, part of hardware repository
Example Hardware Description

<SensorType>
  <Sensor>
    <Name>SHT11</Name>
    <SensorTypeID>10011</SensorTypeID>
    <Description>Humidity accuracy: +-3.0%RH  Temperature accuracy +-0.4°C</Description>
    <DataType>unit8</DataType>
    <Unit>%RH / °C</Unit>
  </Sensor>
  <SensorAttribute>
    <TimeToSample unit="ms">10</TimeToSample>
    <EnergyConsumptionPerSample unit="mJ">10</EnergyConsumptionPerSample>
    <SamplingRate unit="Hz">10</SamplingRate>
  </SensorAttribute>
</SensorType>
The configKIT Approach – Input for Application Designer

- User Interface, Application designer
  - Hardware selection
  - Application description

Module Selection
- Resolve Dependencies
- Semantic Evaluation
- Estimation of
  - Code size
  - Security properties
  - Energy assessment

- List of potential solutions satisfying requirements

- Blueprint for system integration

- Hardware Repository (XML)
- Software Repository (XML)

refine
Challenge (2) – App description & Module Selection

Selection of the set of modules satisfying the application’s needs

Application:
- Need for Public Key Cryptography
- 20 operations per hour
- Life time > 6 months
- Hardware: MicaZ

Security Requirements:
- Data secure for 20 years

Diagram:
- Asymmetric Cryptography
- ECEG
- RSA
- ECC
- ECEG
- Random Number Gen.
- Real RNG
- Pseudo RNG
- Classic
- Stream cipher
Selection: Current State

• Selection of Hardware

• Selection of software modules
  functional description or explicit modules

+ required parameters
What’s the application anyway?

Application is just another software Module as part of the Software Repository
The configKIT Approach – Module Selection

Module Selection
- Resolve Dependencies
- Semantic Evaluation
- Estimation of
  - Code size
  - Security properties
  - Energy assessment

List of potential solutions satisfying requirements

Blueprint for system integration

User Interface, Application designer

Hardware selection

Application description

Hardware Repository (XML)

Software Repository (XML)
Example (1)

Signature algorithm

Asymmetric Cryptography

ECDSA

RSA-Sig

Cryptographic Hash

SHA-1

SHA-1 general

RSA

ECC

RSA 1024 bit general

ECC 160 bit general

ECC 160 bit mica

ECC 224 bit MSP430

ECC 233 bit Hw-accel

RSA-Sig

Cryptographic Hash

SHA-1

SHA-1 general
Example (2) – Signature algorithm on MSP430

Signature algorithm

Asymmetric Cryptography

ECDSA

RSA-Sig

Cryptographic Hash

SHA-1

SHA-1 general

RSA

ECC

RSA 1024 bit general

ECC 160 bit general

ECC 160 bit mica

ECC 224 bit Msp430

ECC 233 bit Hw-accel

Step 1: mark all selected modules, remove not suited
Step 2: look for dependencies, implementations
Until we have a set of implementations and no open dependencies
Optimizations

• Original algorithm is NP complete
  → Optimizations required

Approach:
• Remember decisions
  - Do not follow unbeficial sub-trees more than once

→ Problems:
  - Re-convergences (no actual tree)
  - how make sure unbeficial sub-tree is not better for another pre-selection?
How to solve the model-reality-gap

- **Simplify reality**
  - Unified interfaces
  - Backbone operating system/message controller/middleware

- **Increase complexity of models**
Assessment Process

• Memory:
  - Addition of single modules
  - Problem: size(A+B) != size(A)+size(B)
    → Simple addition is rather an approximation (upper bound)

• Energy:
  - currently qualitatively (good, medium, bad)
    → Allows comparison of similar protocols
    → Not yet satisfying
  - Required: prediction of actual energy consumption (uJ/op)
    → A lot of issues!

• Security & Dependability:
Challenge (3) – Proof of Security

Which flow is the best and why?

- Dependability?
- Concealment?

1. Read Sensor → Encryption (eg. CaMyTs) → Aggregation → Replication
2. Read Sensor → Encryption (eg. CaMyTs) → Replication → Aggregation
3. Read Sensor → Replication → Encryption (eg. CaMyTs) → Aggregation
Proof of Security (2) – How it is currently done

• straightforward logic: (probabilistic security?)
  secure module + not secure module = ??

• For concealment:
  secure + not secure = not secure (the weakest module)

• For robustness (replication):
  secure + not secure = secure (one replication is ok)

• For integrity:
  secure + not secure = secure (one proof of integrity is ok)

→ NOT REALLY SUFFICIENT → how to do it better?
Security Assessment

• Security metric:
  0 = no security/does not matter
  1 = weak
  2 = good
  3 = strong

• Propagation from bottom to top of system

• Security properties propagate through the dependency graph

• Currently support of three security properties:
  - Secrecy
  - Integrity
  - Robustness

sec = f_E(sec_C, sec_D)
= f_E(f_C(sec_A, sec_B), sec_D)
## Security Metric

<table>
<thead>
<tr>
<th>Security class</th>
<th>Attacker</th>
<th>Attacker tools</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No security</td>
<td>attack can be succeed 'by accident'</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>curious hacker</td>
<td>common tools</td>
<td>&lt; 10,000$</td>
</tr>
<tr>
<td>2</td>
<td>organized attacker (academic, crime)</td>
<td>special tools</td>
<td>&lt; 100,000$</td>
</tr>
<tr>
<td>3</td>
<td>large organized attacker (crime, government)</td>
<td>highly specialized tools, laboratory</td>
<td>&gt; 100,000$</td>
</tr>
</tbody>
</table>

An algorithm belongs to class c if it resists all attacks from attacker groups smaller than c.
Similar metrics for other properties?

- Dependability / Safety?
- Maintainability?
- Energy consumption?
- Memory consumption?
The configKIT Approach – further work

Module Selection
- Resolve Dependencies
- Semantic Evaluation
- Estimation of
  - Code size
  - Security properties
  - Energy assessment

List of potential solutions satisfying requirements

- Proof of Security properties
- Automatic Integration

Hardware selection
Application description

User Interface, Application designer

Refine

Hardware Repository (XML)
Software Repository (XML)
Challenge (4) – Automatic Integration

Module Selection

WSN Security Compiler

Image for Sensor Nodes

Image for Cluster heads

Template/Framework For Application

Module Source Code
Conclusions

• Wireless Sensor (and Actuator) Networks are needed!

• Design of software is too difficult and expensive

• What’s missing is a unified middleware or engineering approach

• configKIT approach can help

• Done:
  - way of module description
  - Selection algorithm

• ToDo:
  - Find better metrics for estimation of properties
  - Find a way to verify security and safety properties
  - Automatic integration
Thank You

Questions?

peter@ihp-microelectronics.com