

Towards an Open Source Toolkit for Ubiquitous Device Authentication

PerSec 2007

19. March 2007, New York, NY, US

Rene Mayrhofer

Lancaster University, UK

The problem

Wireless communication is insecure

- Especially problematic for spontaneous interaction: **no a priori information** about communication partners available

⇒ User needs to establish **shared secret** between devices

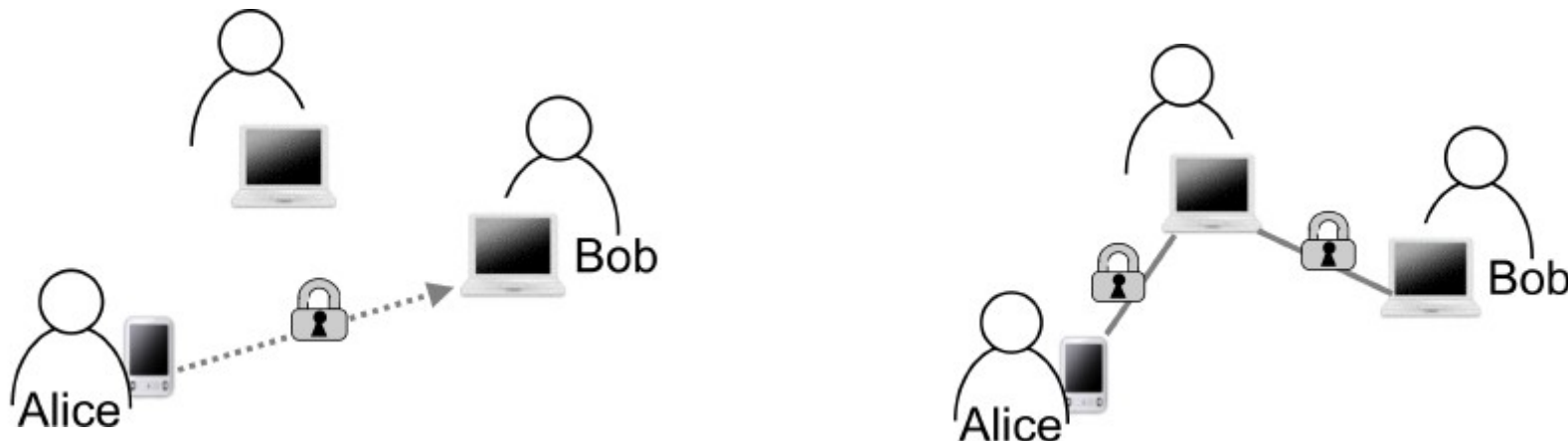
Example: mobile phone + Bluetooth headset



Why is it a problem?

Secret key exchange over wireless channels

- Can use Diffie-Hellman (DH) for key agreement
- Problem of Man-in-the-Middle (MITM) attacks:



⇒ Secret keys need to be **authenticated securely, intuitively and efficiently**

One possible solution

Solution for devices without conventional UI: **implicit context authentication**

- Authenticate devices when they are in the same context
Anonymous device-to-device authentication instead of identification
- Measure physical properties that are **verifiable** by the user

Proposed methods:

- Concepts: Stajano “resurrecting duckling”, Kindberg et al. “constrained channels”, Balfanz et al. “location-limited channels”, Hoepman “ephemeral pairing”, ...
- Implementations: e.g. “Seeing-is-Believing”, “Loud and Clear”, “LoKey”, “Spatial References”, “Network-in-a-Box”, “Shake well before use”, ...

Don't re-invent the primitives

Currently:

- Interesting proposals to solve the authentication problem
- Using different terminology, different underlying concepts
- Implementations specific to the approach, and sometimes to a single demonstration application
- No re-usability of protocols, cryptographic primitives, sensor data handling, user interfaces, etc.
- Hard to reproduce published results

To foster research in the area:

- Have a repository of authentication techniques, methods, and protocols
 - Provide tested and re-usable primitives for creating new protocols
 - Make proposals and protocols comparable and interchangeable
 - Provide real-world sensory data sets for reproducibility and for testing new approaches
- ⇒ allow to focus on new and interesting applications that use these primitives

Take OpenSSL as an example

Frameworks

- Framework defines complete structure of (parts of) applications with hooks
- Rapid application development if framework fits the application

Toolkits

- Toolkit provides a library of components that can be used independently and combined with each other
 - Application can – but does not need to – use defined structure
- ⇒ “Cherry-picking” approach

Prominent example: OpenSSL

- Popular crypto toolkit for ANSI C (available on most *NIX, Windows, embedded systems)
- SSL/TLS support on top of the cryptographic primitives, and support application
- Distributed as distinct libraries that build upon each other and executables

How should it do it?

- **lightweight**: in terms of CPU, RAM, storage, bandwidth, power consumption, user attention, etc.
- **self-contained**: include dependencies if reasonable
- **simple to use**: APIs, default values, components as black boxes for application developers
- **extensible**: as few “must implement” interfaces as possible
- **vertical**: components for all layers, from sensing and communication up to UI
- **interoperable**: ASCII protocol messages
- **secure**...
- **event based**
- **free** (as in speech) to use, also in commercial/closed source projects

Where do we want to use it?

Standard desktop/laptop/server platforms:

- Java
- .NET

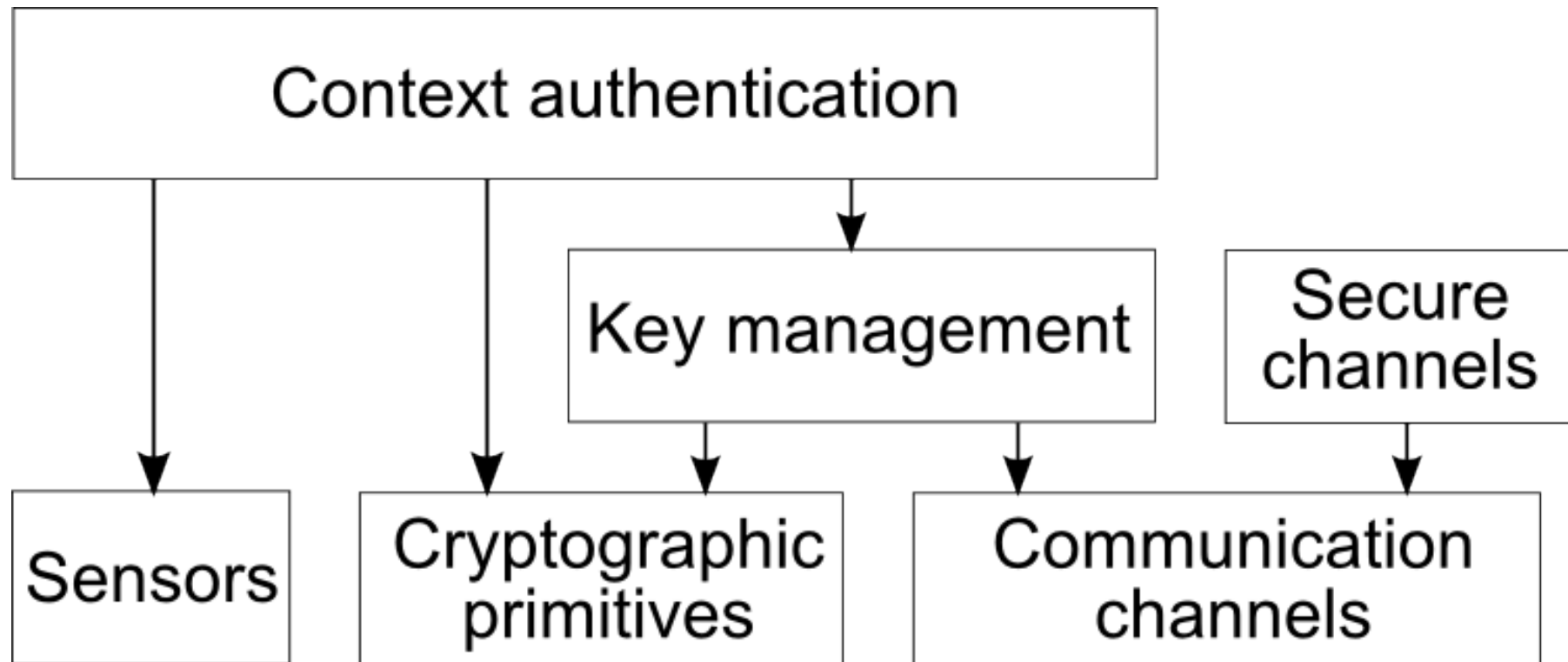
Embedded mobile phone/smart device platforms:

- Java (J2ME)
- C++ (Symbian)

Small device/sensor node platforms:

- TinyOS

Overview of the toolkit



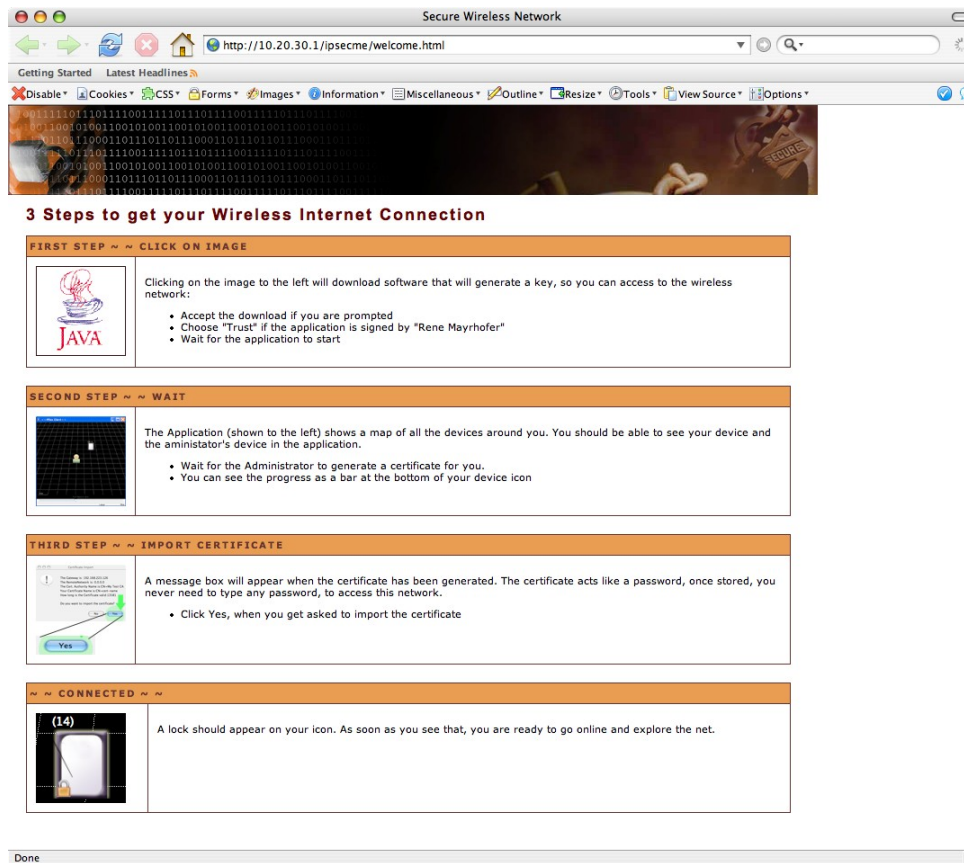
Components in the current release

- **Cryptographic primitives**: ciphers, hashes (JCE and Bouncycastle with wrappers), DH with default parameters and utility methods, interlock*, on-the-fly creation of X.509 CAs and certificates
- **Communication channels**: threaded TCP and Bluetooth RFCOMM servers using same interface (transparently interchangeable), UDP multicast, Bluetooth background discovery and peer management (opportunistic authentication)
- **Key management protocols**: DH-over-streams (TCP or RFCOMM), Candidate Key Protocol
- **Sensors and feature extractors**: ASCII line reader with various implementations for accelerometers, simple statistics, time series aggregation, activity detection/segmentation, FFT, quantizer
- **Context authentication protocols**: spatial references, shared motion (shaking)
- **Secure channels**: IPSec tunnel and transport (Linux, MacOS/X, Windows)

Utilizing Log4j, JUnit, Ant build system including J2ME builds

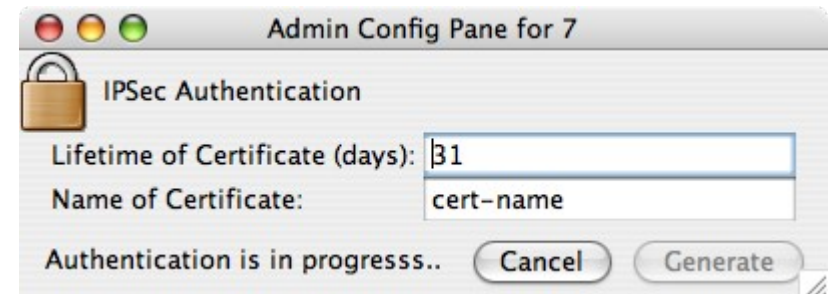
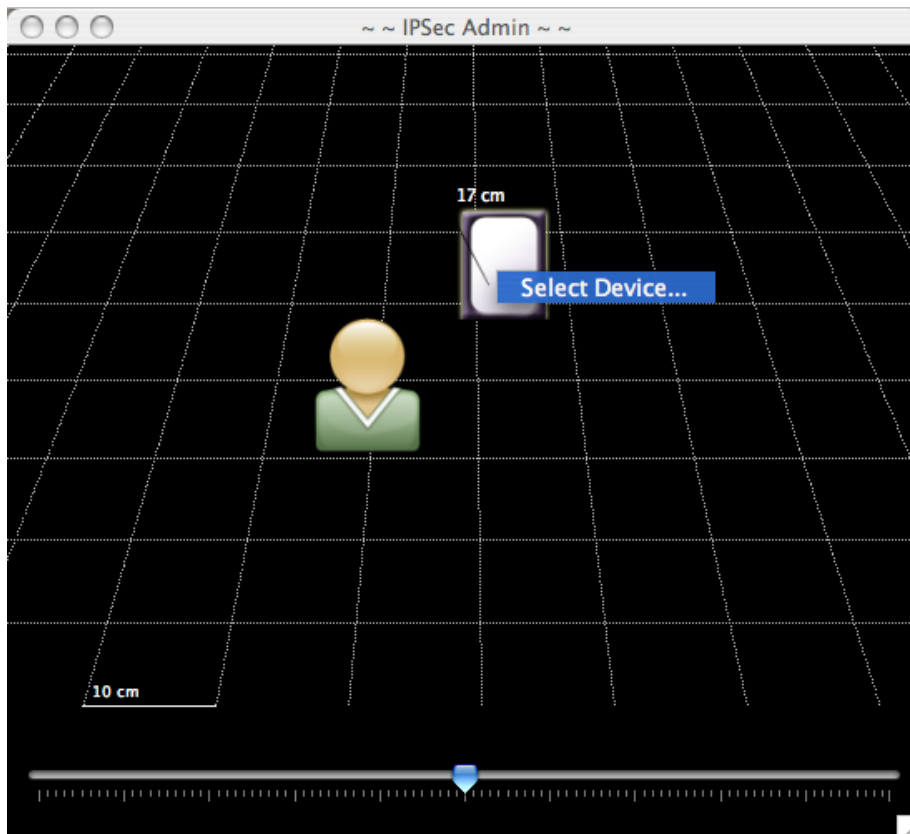
Demonstration applications using OpenUAT

IPSecME (IPSec Made Easy): creating IPSec connections using a spatial authentication proxy



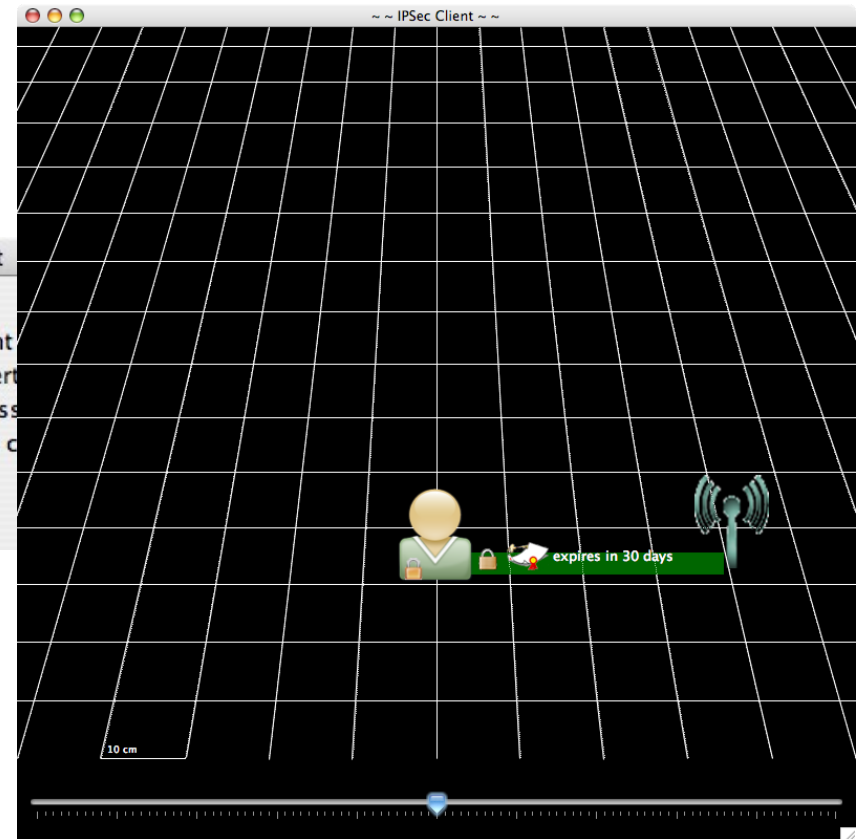
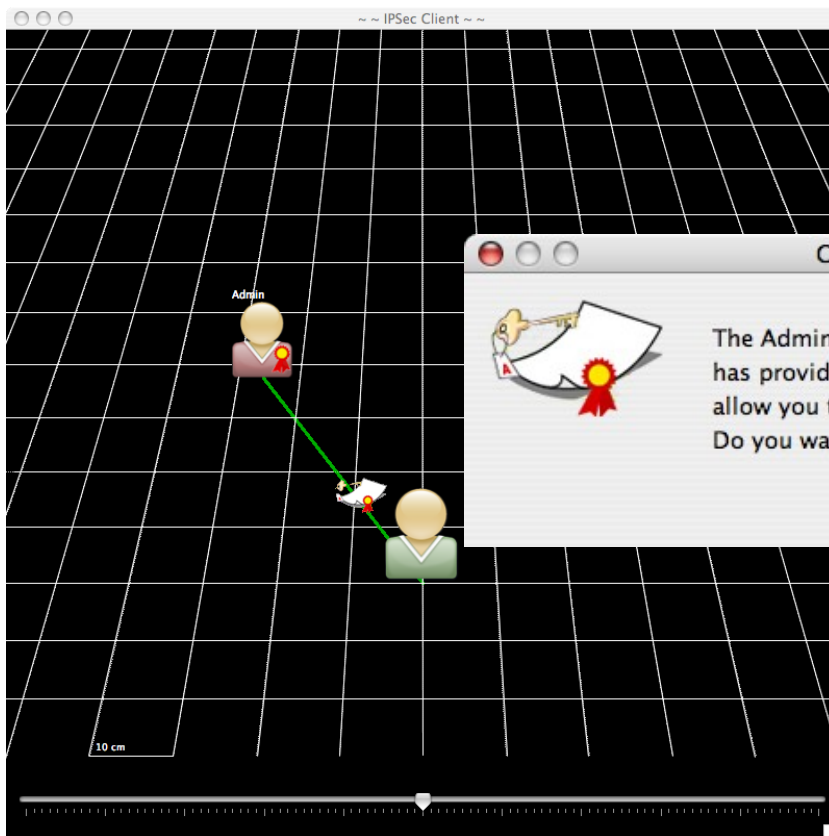
Demonstration applications using OpenUAT

IPSecME (IPSec Made Easy): creating IPSec connections using a spatial authentication proxy



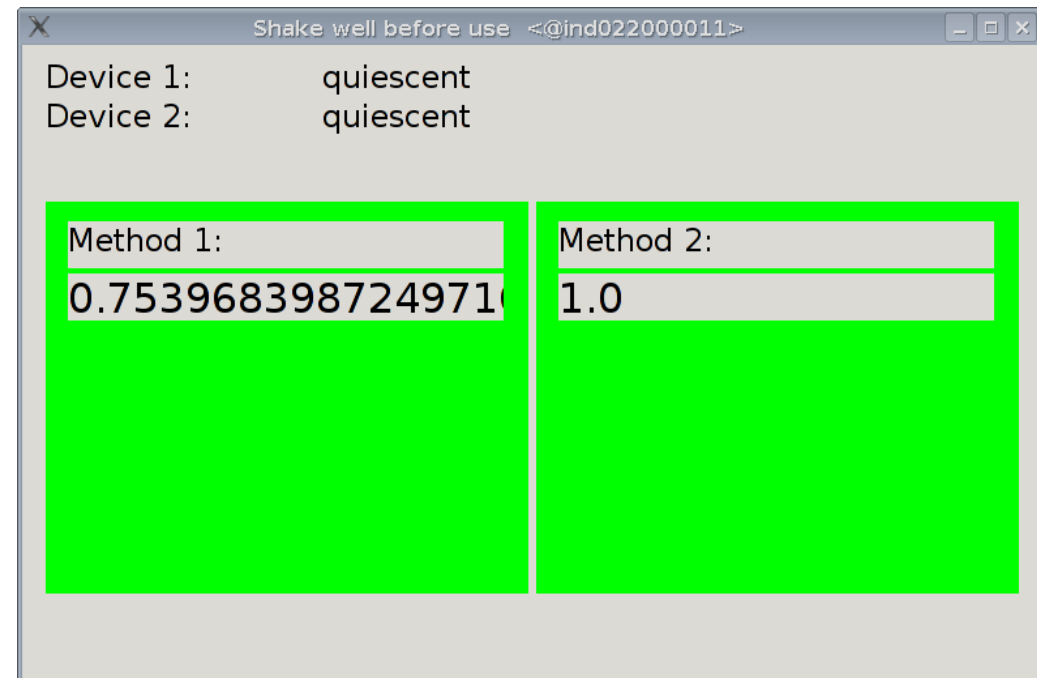
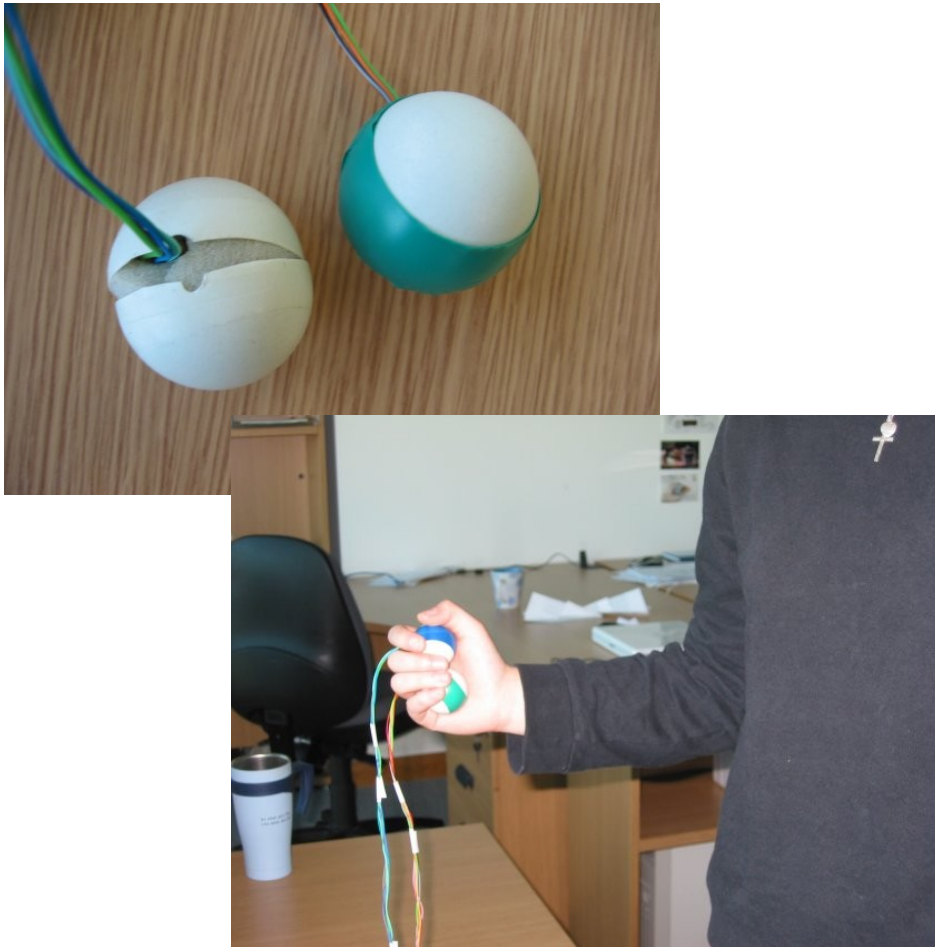
Demonstration applications using OpenUAT

IPSecME (IPSec Made Easy): creating IPSec connections using a spatial authentication proxy



Demonstration applications using OpenUAT

Shake well before use: Authentication based on Accelerometer Data



Download

Documentation, applications, data sets:

<http://www.openuat.org>

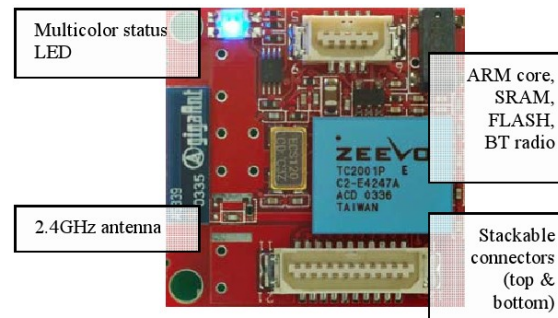
Source code, mailing list, bug tracker:

<http://sourceforge.net/projects/openuat>

Scaling it down

Current developments:

- Implementing the method on embedded devices
 - „Nokia 5500 Sport“ – includes 3D accelerometer with API
 - Intel iMote 1 with TinyOS – to emulate headset
- Bluetooth instead of TCP and UDP
 - different way of communication setup
 - no broadcast



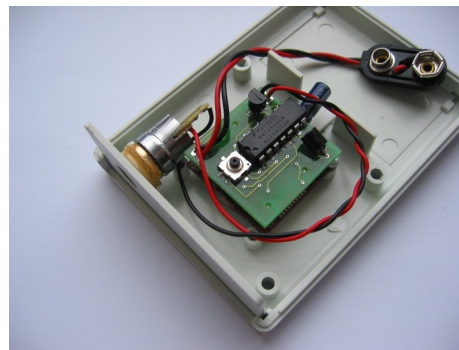
Additional protocols

Student projects for implementing:

- Extended variant of “**Seeing-is-Believing**” (with concepts from “Short Authenticated Strings”)
- “**Loud and Clear**”

both should be implemented for laptops and mobile phones

New protocol with a one-way semi-authentic channel (**visible laser**)



The Future: **You**

- Using the toolkit
- Integrating your own protocols
- Contributing real-world data sets
- Community benefits from reproducibility and re-usable components

“Portability is for people who cannot write new programs.”

Linus Torvalds, 1992-01-29, comp.os.minix

Linux is currently one of the most portable operating system kernels...

Thank you for your attention!

Slides: <http://www.mayrhofer.eu.org/presentations>
Later questions: rene@mayrhofer.eu.org

OpenPGP key: 0xC3C24BDE
7FE4 0DB5 61EC C645 B2F1 C847 ABB4 8F0D C3C2 4BDE