Cryptography for Embedded Systems

Elisabeth Oswald

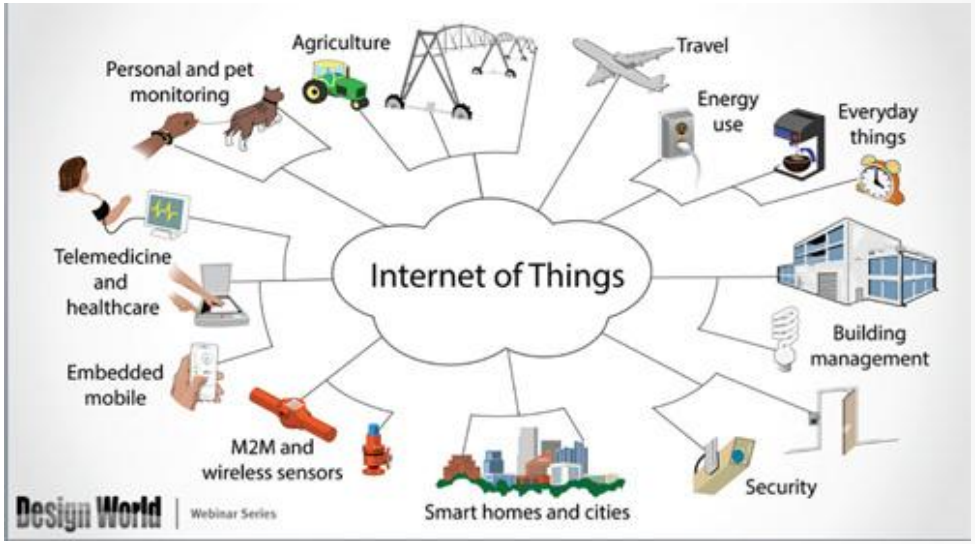
Reader, University of Bristol

Outline

1. **Embedded devices**
   History, role and importance, use of cryptography
2. **Security challenges**
   Nothing is ever easy ….
3. **Future directions**
   Guess what: it is still not easy

Embedded Devices

... Ghz processor, graphics, sound, wireless, bluetooth, SIM, GPS, …



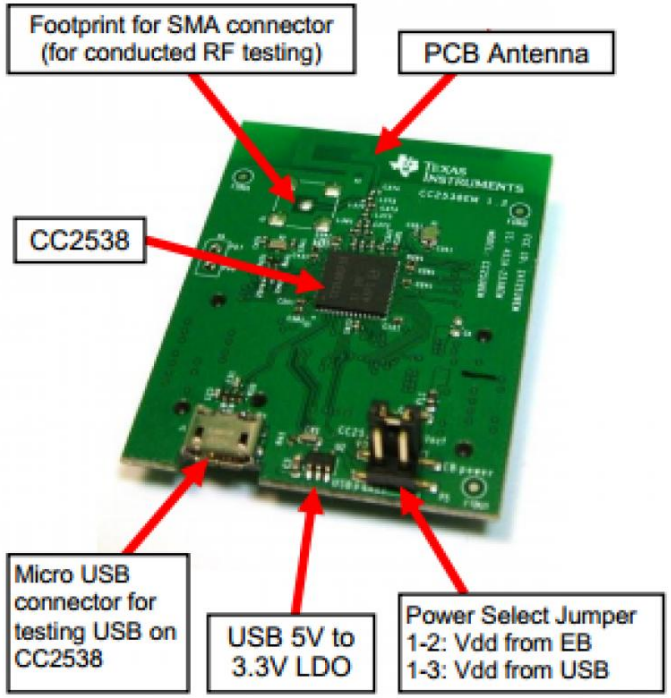
Pictures taken from various online resources.

Key Characteristics of Embedded Devices

Micro and other processors, FPGAs
Can be fast, powerful but must be energy efficient

Connectivity
Often multiple standards are supported.

Sensors
Temperature, movement, etc.



Embedded OS
Nowadays is multi-purpose, multi-tasking/threading

Security
TLS, DTLS, IEEE 802.15.4, EMVCO, PayTV standards

Real time
Sensor nodes, cyber physical systems

Security Challenges: Implementation Attacks

Mostly via cat and mouse game, well documented in the Pay-TV application area:

- F-card to H-card: no crypto to poor crypto,
- Countered by the 'infinite loop', counteracted by the 'un looper' (glitching boards)
- Countered by frequent updates that are necessary descramble content, hackers up their game and provide 'life patches' (i.e. within 15 mins)!
- Finally, content providers demand serious investment into **side channel and tamper resistant** smart card. (Now the set top box is the prime target.)

Point is: Content providers demanded strong security. Similarly, security evolved in other systems, e.g. GSM/UMTS (required 3 generations to become robust).
Automotive security still in its infancy, smart meters, and anything IoT.
But who will be 'demanding' strong security there?

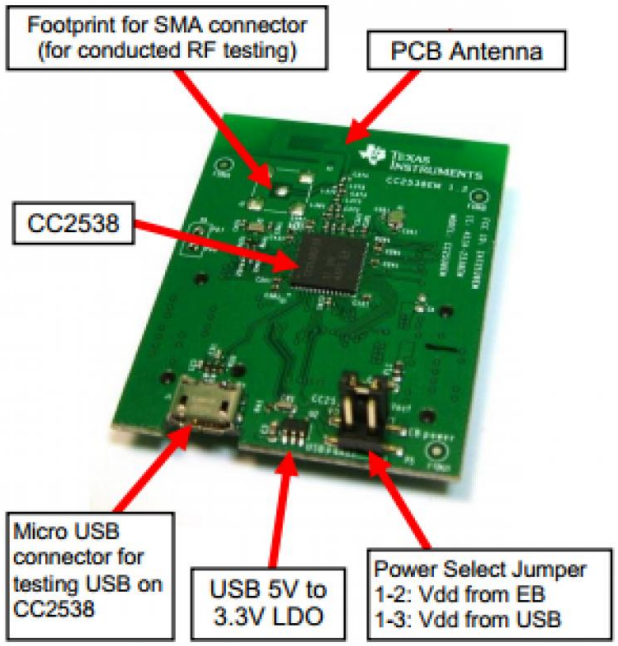
Security Challenges/Failures: e.g. CC2538

32-bit ARM CPU

Has AES in hardware, as well as ECC but not documented (until recently), PRNG support

Contiki OS

Implements DTLS with a fatal flaw: PRNG produces biased 16 bit output



Reasons

Standards not clear enough for non-domain experts, developers not sufficiently aware, insufficient support for crypto in hardware, lack of memory in the community, no security evaluation standards.

Not to mention that they hardware crypto is neither leakage nor tamper resistant.

http://www.theregister.co.uk/2010/01/15/smart_meter_crypto_flaw/

Future Directions

1

Verifiable implementations

Compiler support, language support, leakage simulators, …

2

Meaningful Leakage evaluations

Need to account for increase in computing power as part of evaluations

3

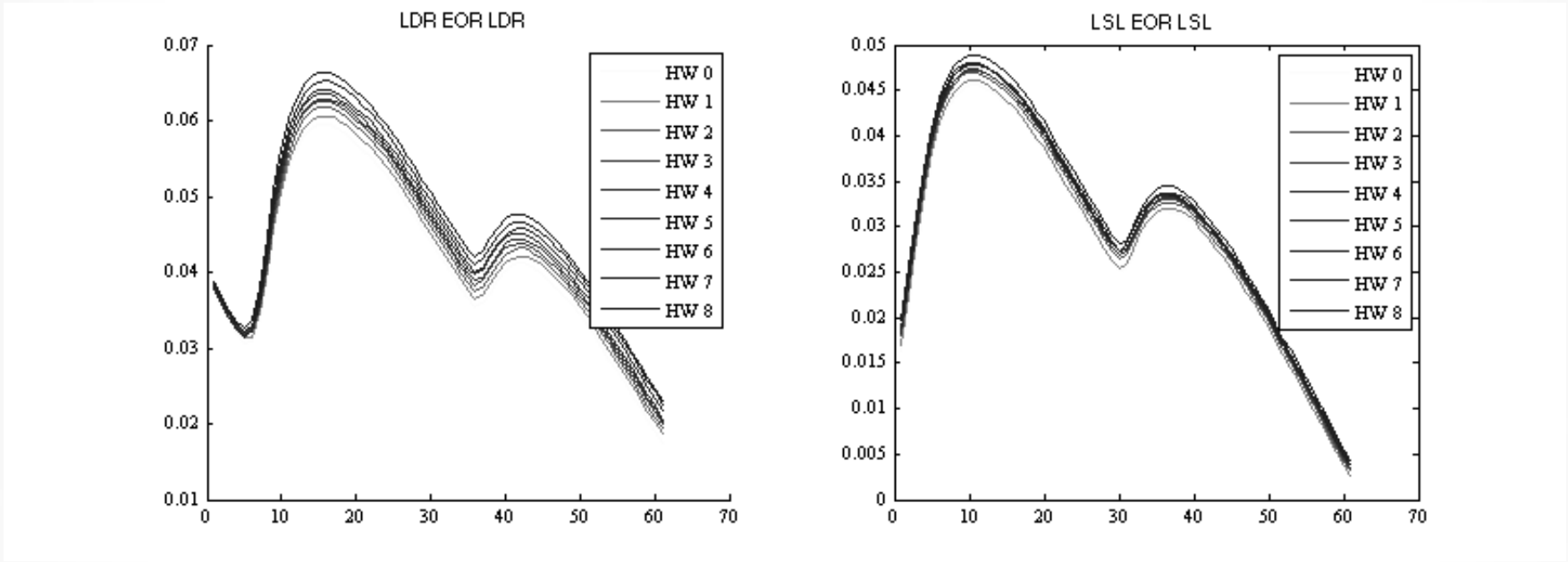
Leakage resilience per design

Primitives that tolerate some leakage under reasonable assumptions

Verifiable Implementations, 1

Need to (formally) verify the correctness (functionally, cryptographically, leakage) of implementations (aka code):

- Advances made by Dupressoir et al. w.r.t. verifying implementations of masking schemes: but all with simple leakage models
- But instructions leak depending on what happens before and after!



Verifiable Implementations, 2

Need to (formally) verify the correctness (functionally, cryptographically, leakage) of implementations (aka code):

- Advances made by Dupressoir et al. w.r.t. verifying implementations of masking schemes: but all with simple leakage models

Meet ELMO: Emulating Leaks for the M0!

- A statistical modeling approach capturing instruction-level leakage models for arbitrary instruction sequences (i.e. processor states) working with 32-bit data
  - Test the contribution of explanatory variables with an F-test and keep only those 'that matter'
- For the ARM Cortex-M0 using instruction-triplets leads to excellent models
  - Shown via comparative leakage detection tests on real vs. ELMO data.
- Using ELMO one can verify leakage without access to a testing lab!

Meaningful Leakage Evaluations, 1

BeagleBone Black

Attack Environment

Hardware:

- ARM Cortex-A8 1 GHz CPU (High clock rate)
- ARM NEON SIMD (High degree of parallelism)
- TI proprietary cryptographic hardware (RNG, SHA-1, AES)

Software:

- Debian Wheezy (3.15) (Full unmodified Linux distribution)
- OpenSSL 1.0.1*j* (Bulk encryption)

Summary of Attack Results

Implementation	Hardware	Trigger	Acquisitions	Data
T-tables	ARM core	GPIO-based	3000	46 kB
T-tables	ARM core	Network-based	100	400 kB
Hardware	Co-processor	DMA-based	50 000 ~~500 000~~	<1GB ~~7GB~~
Bit-sliced	NEON core	GPIO-based	5000	625 kB

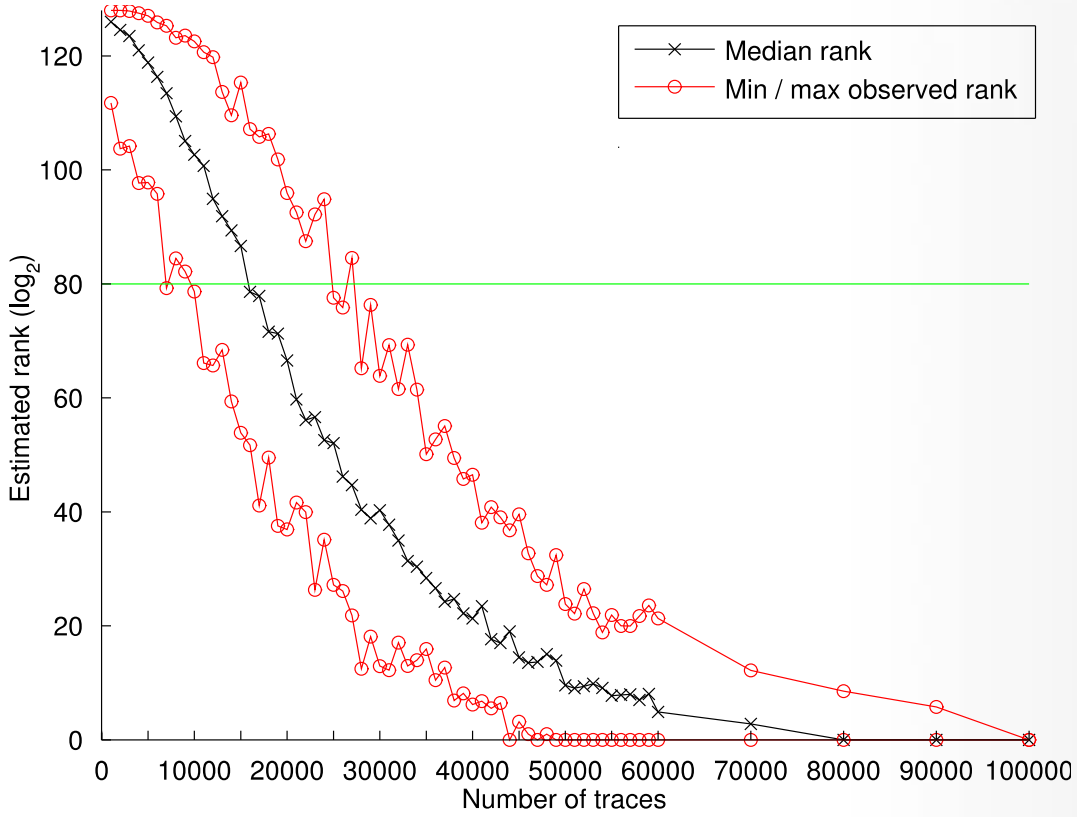
But is this the best strategy or are these typical results?

Meaningful Leakage Evaluations, 2

Previous slide suggests 50k ‘traces’ for key rank 0 using a single target attack.

What are the characteristics of the rank distribution?

How low can you go w.r.t. key rank? Labynkyr is a multi-threaded implementation of fastest enumeration/verification algorithm (Asiacrypt 2015):



- On a single workstations (hexa-core Ivy Bridge-EP Intel Xeon E5-1650v2 CPU and 32 GiB of 1600 MHz DDR3 RAM), we enumerated/verified **2^(41.8) AES keys in 24 hours**
- 400 nodes of BlueCrystal (Sandy Bridge Intel Xeon E5-2670 CPUs, clocked at 2.6 GHz and with 4 GiB RAM available per core), we enumerated/verified **2^(47.99) AES keys in 32.42 hours** (could be bought in from Amazon at 680.82 USD).

Leakage Resilience per Design

- Probing model, random probing model, noisy probing model, noisy leakage model: simple circuit with Gaussian noise
  - Higher order masking, aka secret sharing, threshold implementations, very active area of research

- Leakage resilient cryptography: OCLI with lambda leakage (i.e. arbitrary function of well defined state leaks up to lambda bits)
  - Most constructions too expensive, unrealistically strong yet any straightforward implementation would still need countermeasures, but a very promising key update by Kiltz-Pietrzak that has spun off several primitive, loose connection with ‘fresh rekeying’

- Verifiable leakage (under physical assumptions): theoretical leakage simulator, with a a practical realisation, leading to typical real vs. ideal/simulated proofs
  - We showed that the only proposed simulator is irreparably flawed.

Conclusion

1

Cryptography requires careful standardisation and implementation

Cryptographic standards have to do more than just provide high-level specifications (i.e. they serve more than interoperability), they need to ensure that 'stupid' mistakes are less likely. Implementing cryptography has to evolve from being an 'art': we need verifiable implementations, that take information leakage into account.

2

Cryptography has to be pragmatic

Cryptography has to actually address security needs: e.g. authenticated encryption that is resilient to leakage and probing attacks in the real world, group key exchange/update with leakage/probing resilience, etc. All whilst being 'cheap' enough to work on 'small' devices.

Thank you for your attention

www.silent.cs.bris.ac.uk

(Useful) References

Longo, Martin, Mather, Oswald, Sach, Stam: How low can you go?. IACR ePrint Archive 2016: 609 (2016)

Martin, Mather, Oswald, Stam: Characterisation and Estimation of the Key Rank Distribution in the Context of Side Channel Evaluations. IACR ePrint Archive 2016: 491 (2016)

McCann, Whitnall, Oswald: ELMO: Emulating Leaks for the ARM Cortex-M0 without Access to a Side Channel Lab. IACR ePrint Archive 2016: 517 (2016)

Martin, O'Connell, Oswald, Stam: Counting Keys in Parallel After a Side Channel Attack. ASIACRYPT (2) 2015: 313-337

Martin, Oswald, Stam, Wójcik: A Leakage Resilient MAC. IMA Int. Conf. 2015: 295-310

Longo, Martin, Oswald, Page, Stam, Tunstall: Simulatable Leakage: Analysis, Pitfalls, and New Constructions. ASIACRYPT (1) 2014: 223-242

Whitnall, Oswald: Profiling DPA: Efficacy and Efficiency Trade-Offs. CHES 2013: 37-54

Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub: Verified Proofs of Higher-Order Masking. EUROCRYPT (1) 2015: 457-485

Kiltz, Pietrzak: Leakage Resilient ElGamal Encryption. ASIACRYPT 2010: 595-612

Medwed, Standaert, Großschädl, Regazzoni: Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. AFRICACRYPT 2010: 279-296