Cloud Security in the context of Fog and Edge Computing

Milosch Meriac
meriac.com
@FoolsDelight

ABOUT
ME

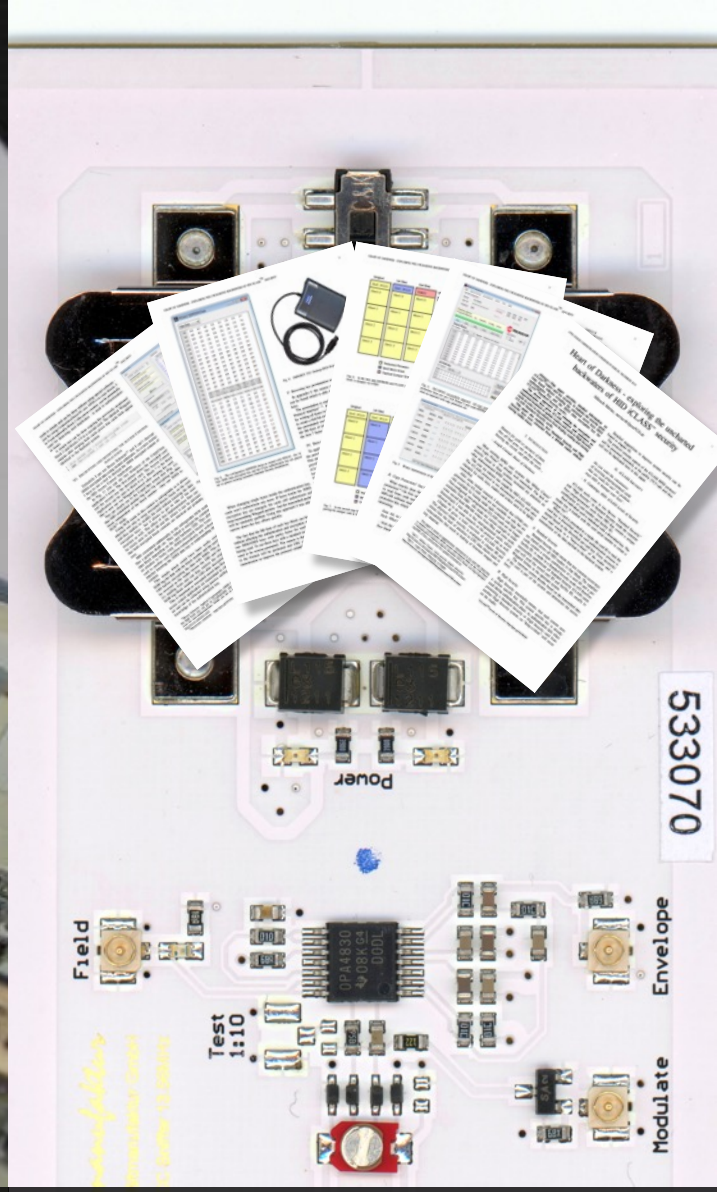
My Open Software & Hardware Projects

SHARED FUN IS TWICE THE FUN

I created a range of open hardware designs and software tools around RF(ID)/BLE security research and electronic art projects. You can find a more information on my work at meriac.com

OpenPCD.org

Open 13.56 MHz
RFID NFC Reader



iClass Security

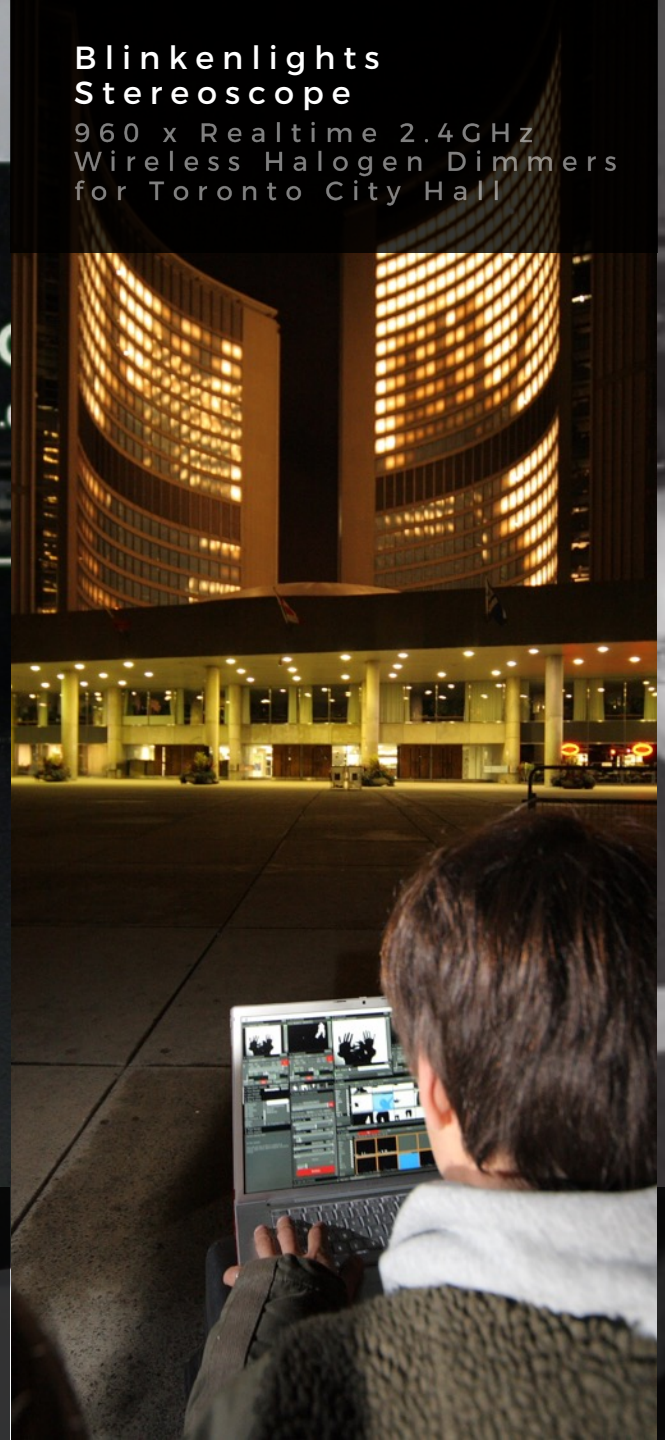
Open 13.56 MHz
RFID Sniffer Design
with HID iClass
decoder

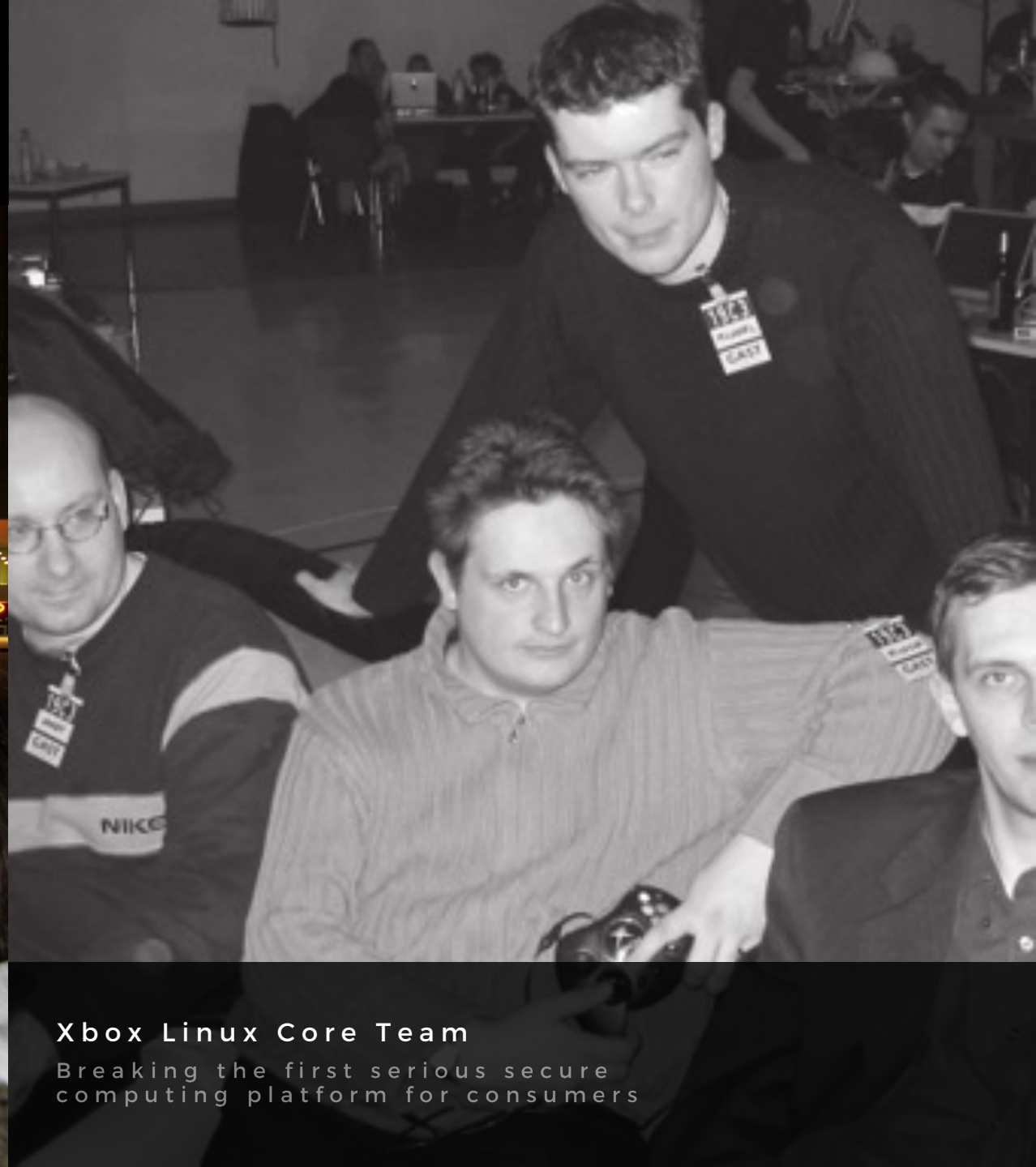
OpenBeacon.org

Realtime 2.4GHz Localization & Human Interaction Analysis, see also SocioPatterns.org

Blinkenlights Stereoscope

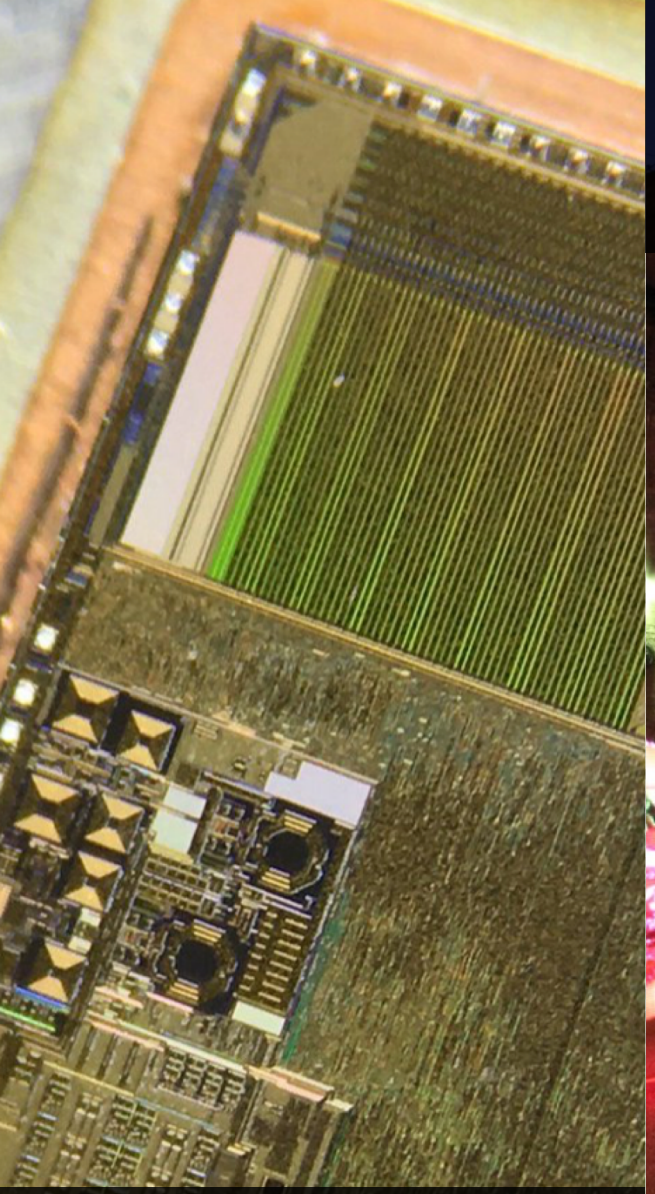
960 x Realtime 2.4GHz Wireless Halogen Dimmers for Toronto City Hall





Xbox Linux Core Team

Breaking the first serious secure computing platform for consumers



Chip Security

Reverse engineering and researching safer chip de-capping for HW attacks

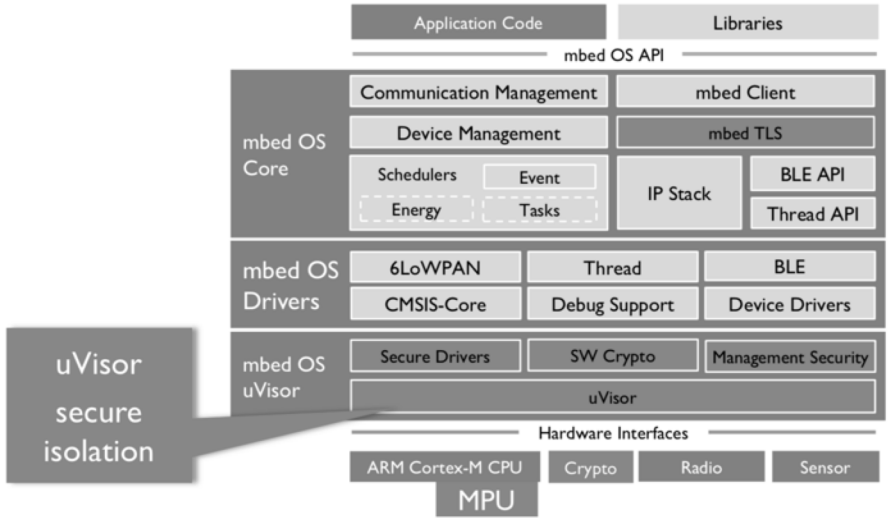
Blinkenstick.org

Light Painting using LPD8806 based RGB strips

Arm Mbed uVisor Security

mbed OS

- mbed OS is a modular, secure, efficient, open source OS for IoT
- Connects to mbed Device Connector



Communication Security

Management Security

Device Security

ARM

Arm Ltd

Led mbed OS uVisor code compartmentalization project for microcontrollers. Uses hardware isolation features of Arm v7M/v8M for isolated execution.

Topics Covered?

QUICK OVERVIEW

1.

Cloud & Edge Security

Definition of Terms

Reality vs. Expectation

Combining Cloud & Edge Security

2.

Fog Computing Software Security

Time Kills

Resource Limitations

Flat Memory Models

Resource Limitations

3.

Fog Computing Hardware Security

Memory Security

Extraction of Secrets

Randomness

Storage Security

With **Edge Computing**, processing is **done by devices themselves**.

With **Fog Computing**, collected data is **pushed to a separate device for processing**.

Quick introduction into IoT cloud service security & trust models

HOW THINGS ARE & HOW THEY SHOULD BE

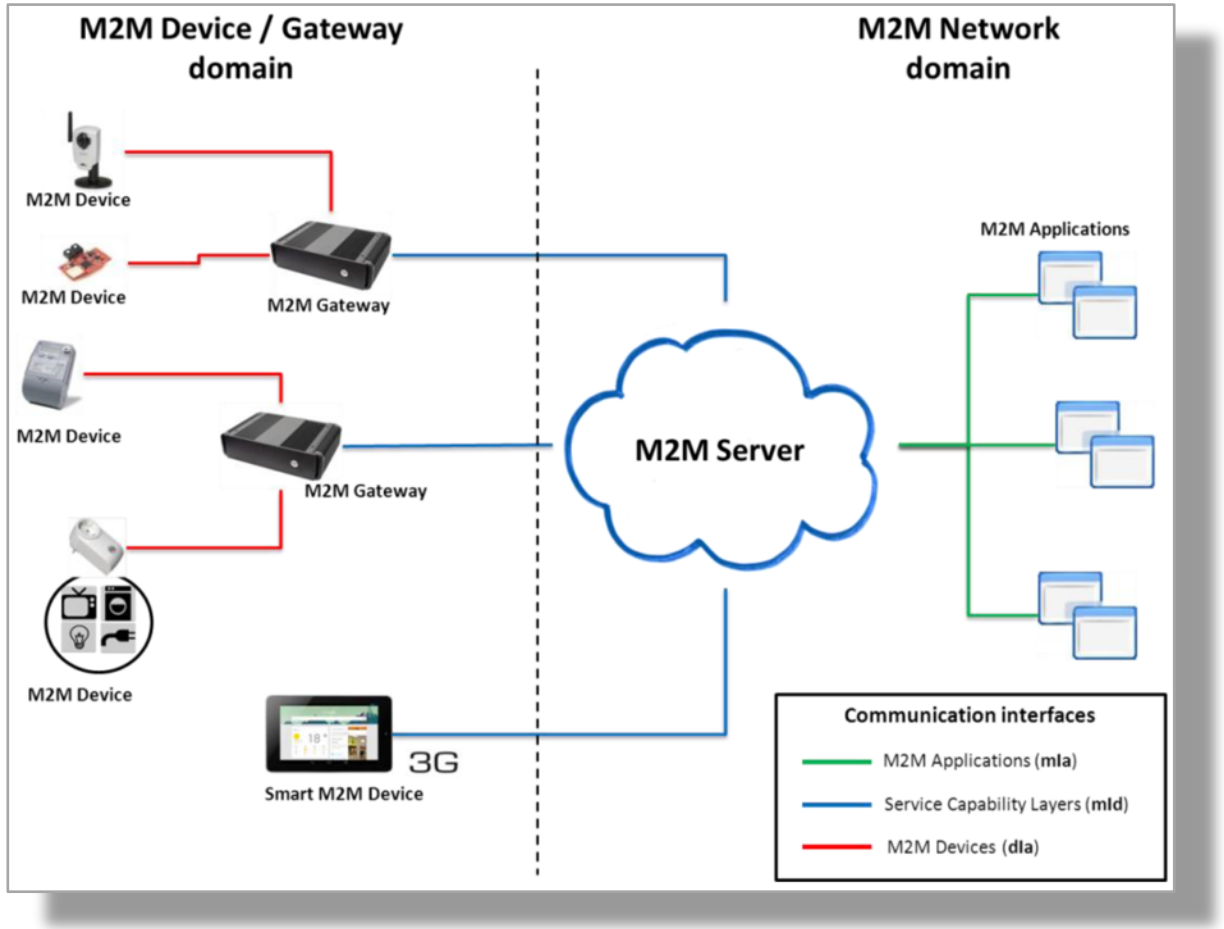
Reality vs. Expectation

THE WORLD COULD BE NICE, ACTUALLY

Historically the Internet of emerged from traditional m2m (machine to machine) automation.

The architectures back then were commonly hub & spoke models – where the hubs were industrial automation system and the spokes connected sensors and actors.

The hubs had ultimate trust in such systems.



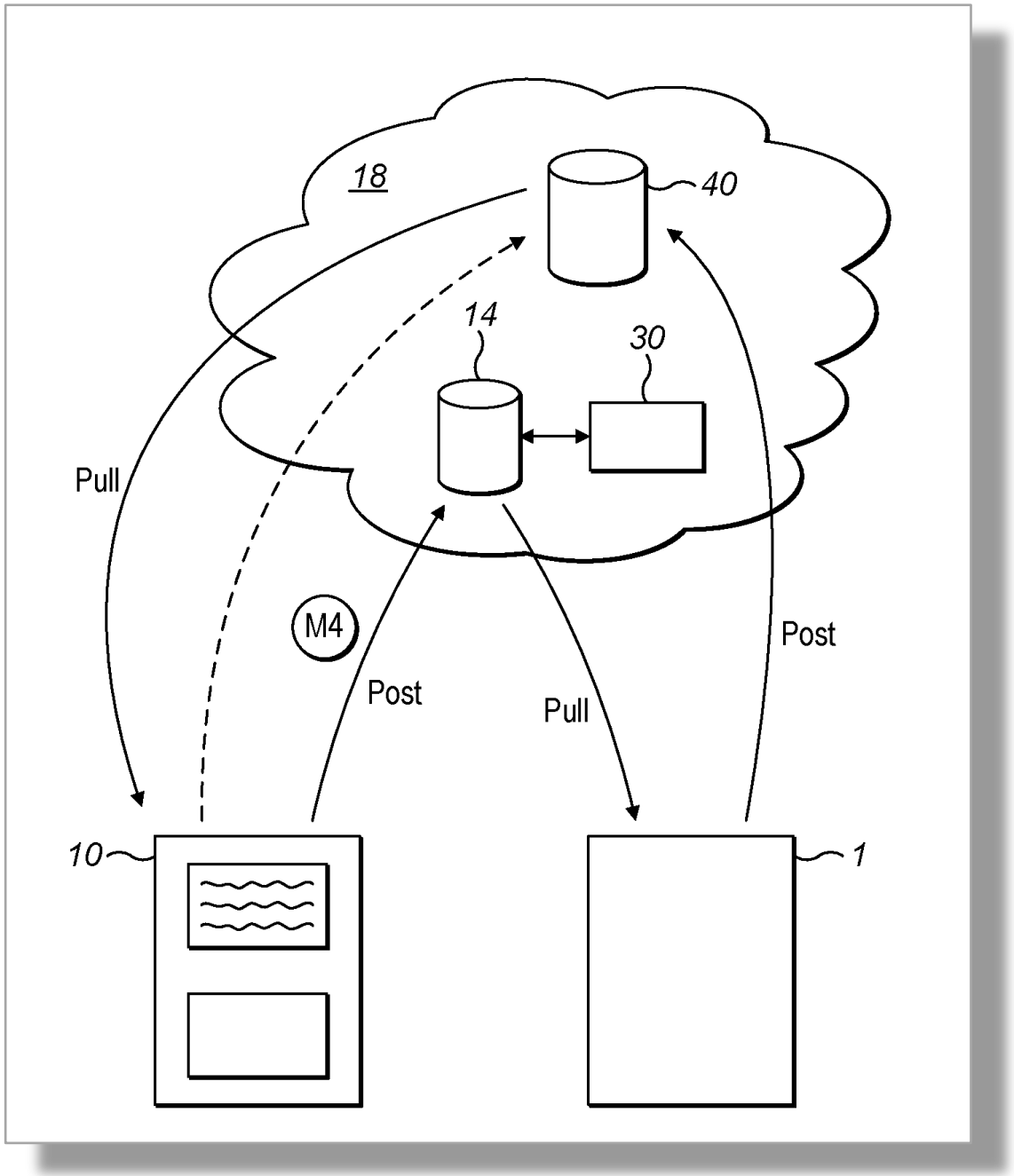
http://www.mdpi.com/1999-5903/6/2/261/htm

Reality vs. Expectation

THE WORLD COULD BE NICE, ACTUALLY

In todays IoT systems that former local server just moved into remote datacenters – together with all its friends like databases, processing nodes and file servers (aka CDN). To distract people from that, these blocks got fancier names.

Because developers are “efficient" and delivery schedules short – the ultimate trust into the former hub also moved along to cloud services.



THERE IS NO CLOUD:
It's just someone
else's computer

THERE IS NO CLOUD:
It's just someone else's computer

Feature Requests:

- Use Edge & Fog computing first.
- Run remaining cloud services at minimal trust.
- Verify security and trust at local nodes & cloud.
- Least-privilege principle for network & code.

Why is Fog Computing Security so hard?

MICROCONTROLLERS ARE DIFFERENT

INEVITABILITY

Security + Time = Comedy

IF A PRODUCT IS SUCCESSFUL, IT WILL BE HACKED

Who can't suppress snickering on seemingly dumb security bugs in other peoples product? It's easy and delightful to declare developers of failed products as incompetent.
As old stuff has old bugs – we won't run out of entertainment anytime soon.

DEVICE LIFETIME

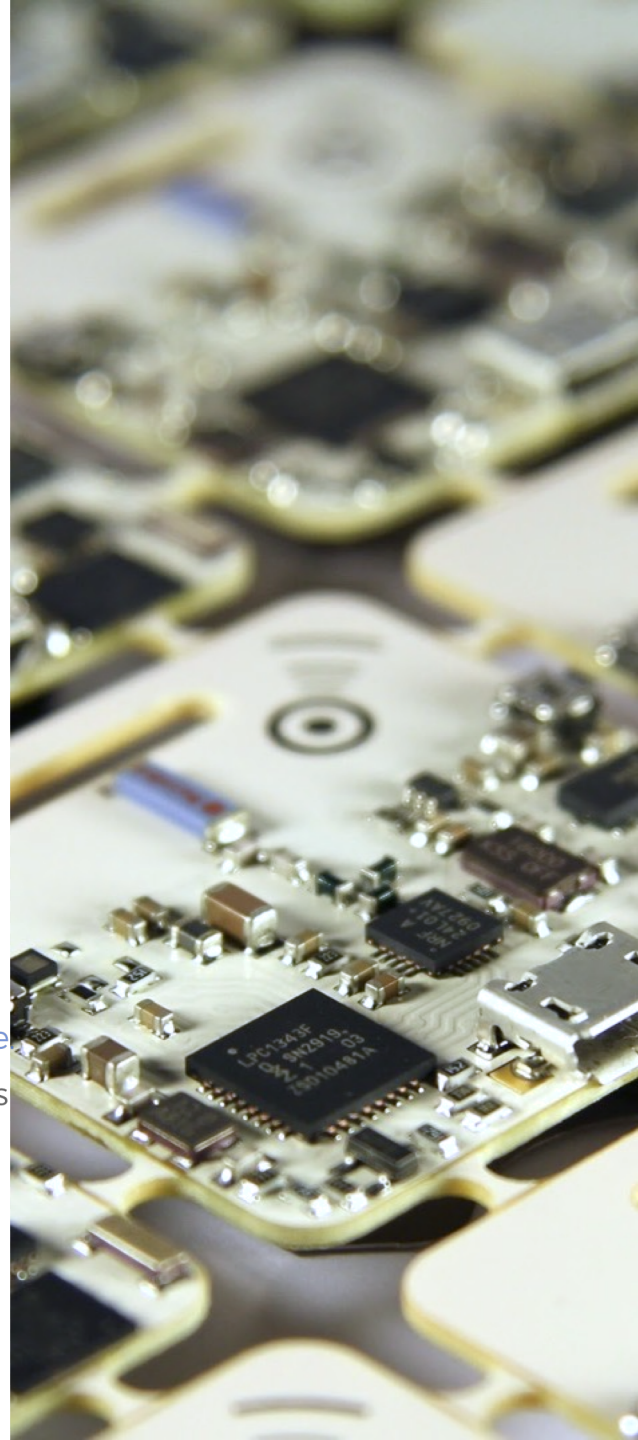
The security of a system is dynamic over its lifetime. Lifetimes of home automation nodes can be 10+ years.

ATTACKS SCALE WELL

The assumption of being hacked at some point requires a solid mitigation strategy for simple, reliable and inexpensive updates.
Do our defenses scale equally well as our attacker do?

YOU CAN'T STOP IT

Value of bugs is expressed by value = number_of_installations x device_value
Increased value and large deployments drive attackers - especially in the IoT.
Massively parallelized security researchers/attackers vs. limited product development budgets and product delivery time frames.



The ugly truth™ is that **defenders** must find all flaws – **attackers** only need to find one.

BREAKING A SYSTEM IS EASY.
FIXING A SYSTEM IS HARD.

The ugly truth™ is that **defenders** must find all flaws – **attackers** only need to find one.

BREAKING A SYSTEM IS EASY.
FIXING A SYSTEM IS HARD.

Feature Requests:

- Enable scalable maintenance.
- Design systems for graceful failure.
- Engineer for recovery, even under attack.

Security from the 80's for today's threats

MMU-LESS ARCHITECTURES

Flat memory models on microcontrollers enable privilege escalation & persistence of bugs.

LIMITED COMPUTING POWER & MEMORY

Vendors commonly use shortcuts to scale public key cryptography down to memory limitations of microcontrollers.

RANDOM NUMBER GENERATION

New wireless microcontrollers have TRNGs. Existing systems either don't have them or don't use them. PRNGs are hard to get right and make promising targets.

INTERNAL STORAGE

Reliable and tamper-proof storage of data is a core security requirement for most security systems. It's very hard to get internal storage right when assuming local attackers.

MMU-LESS MICROCONTROLLER ARCHITECTURES

17

Flat memory models

THE 80'S CALLED, THEY WANT THEIR SECURITY MODEL BACK

A flat address spaces as the result of lack of a memory management units (MMU) does not justify the absence of security. Many microcontrollers like ARM Cortex-M provide hardware memory protection units (MPU) or Arm TrustZone-M instead.

NO SEPARATION

Flat memory models and ignorance of MPUs prevent important security models like "least privilege".

ESCALATION

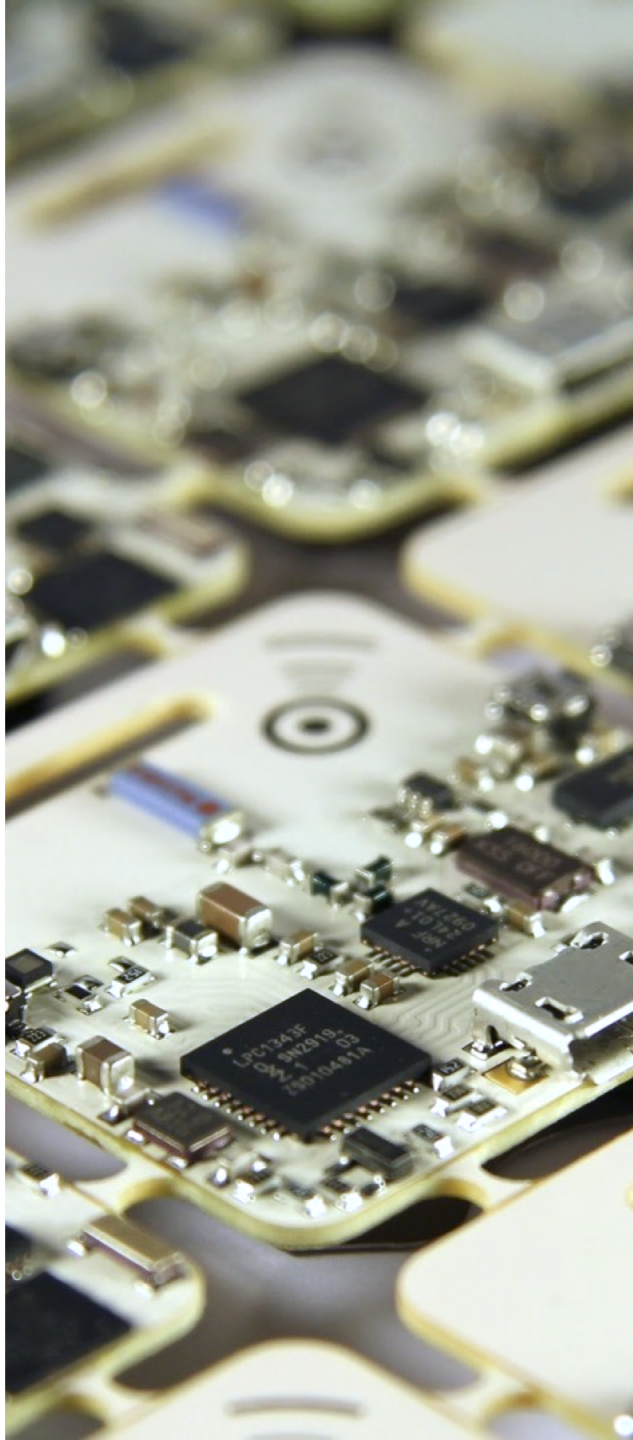
Flat memory models enable escalation & persistence of bugs through uncontrolled writing to flash memories.

VERIFICATION

Security verification impossible due to the immense attack surface and lack of secure interfaces between system components.

LEAKAGE

All your bases are belong to us – thanks to leakage of device class-secrets like keys



LIMITED COMPUTING
POWER & MEMORY

18

Resources matter

ABSTRACTIONS ARE
EXPENSIVE

Point-solutions are easy to build for microcontrollers, but hard to maintain and to reason about. Abstractions are hard to build, but amortize through easier security verification across a large range of devices. For crypto API's this is understood now – but not yet for other system components.

PUBLIC KEY CRYPTO

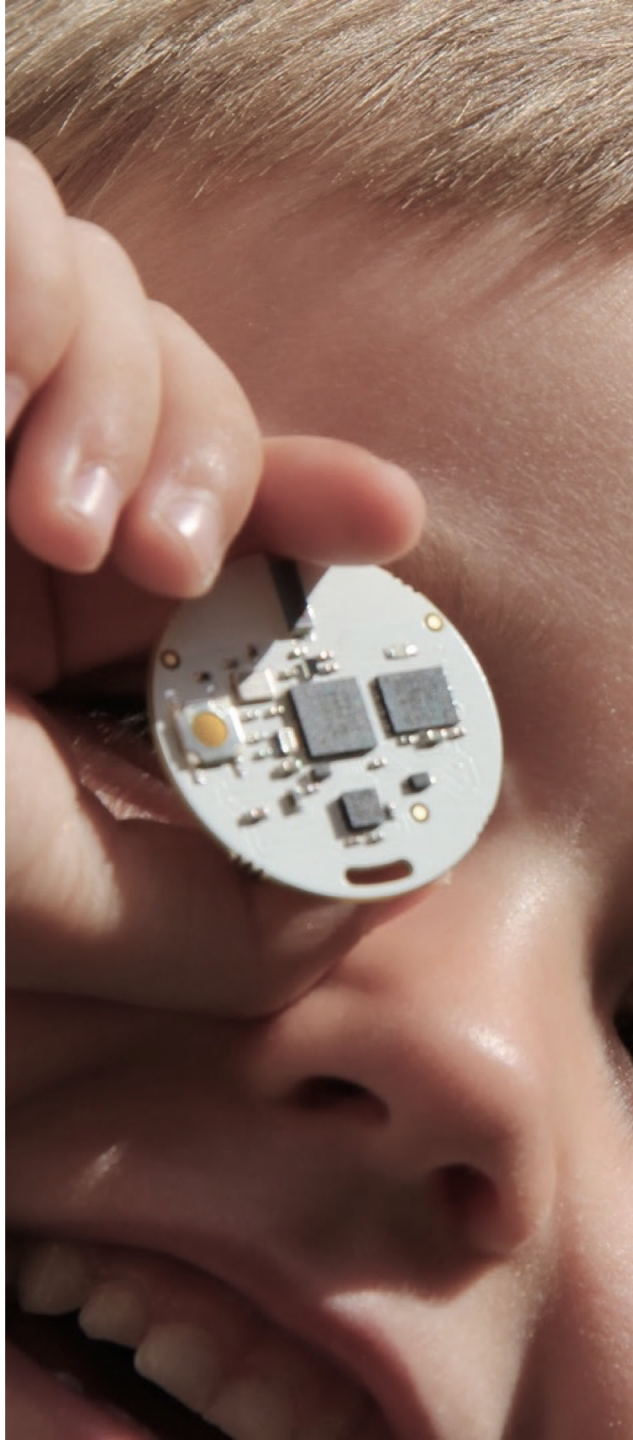
Public-key crypto is absent from many low end devices due to claims of memory and speed constraints.

SHORTCUTS

Vendors take compromises on security to reduce footprint. Think for example of commonplace class key usage versus supporting key provisioning and per-device secrets.

COMMUNICATION

Less abstraction requires more communication of limitations. This usually results in higher-level components having wrong assumptions about low level components. Think of flash configuration storage in presence of local attackers. Be scared.



No device is too small for being secured. There are just wrong engineering choices during design

DON’T BE THAT ENGINEER…

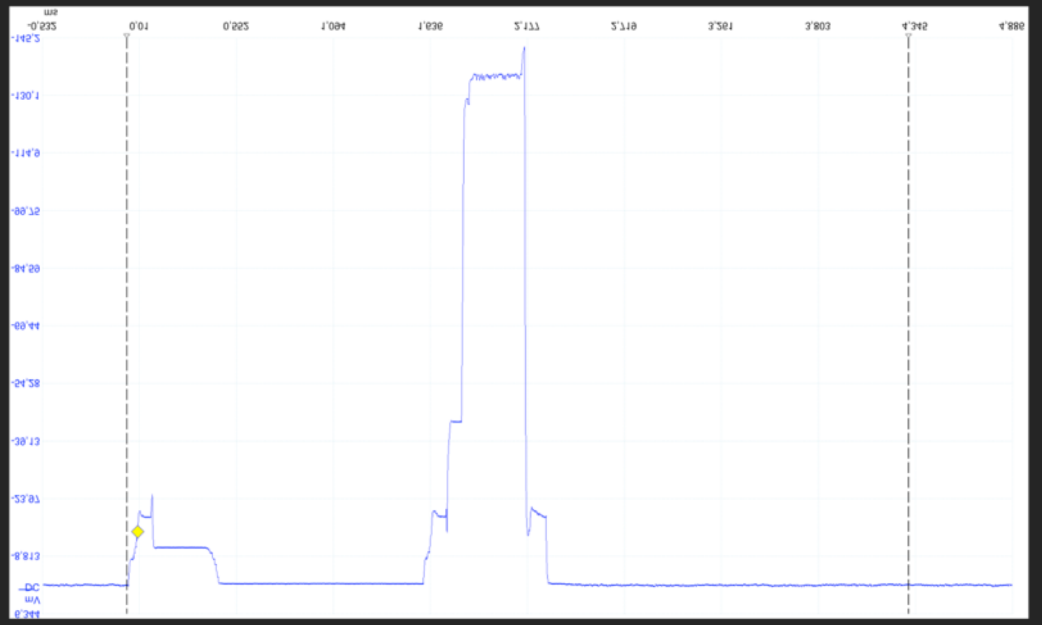
No device is too small for being secured. There are just wrong engineering choices during design

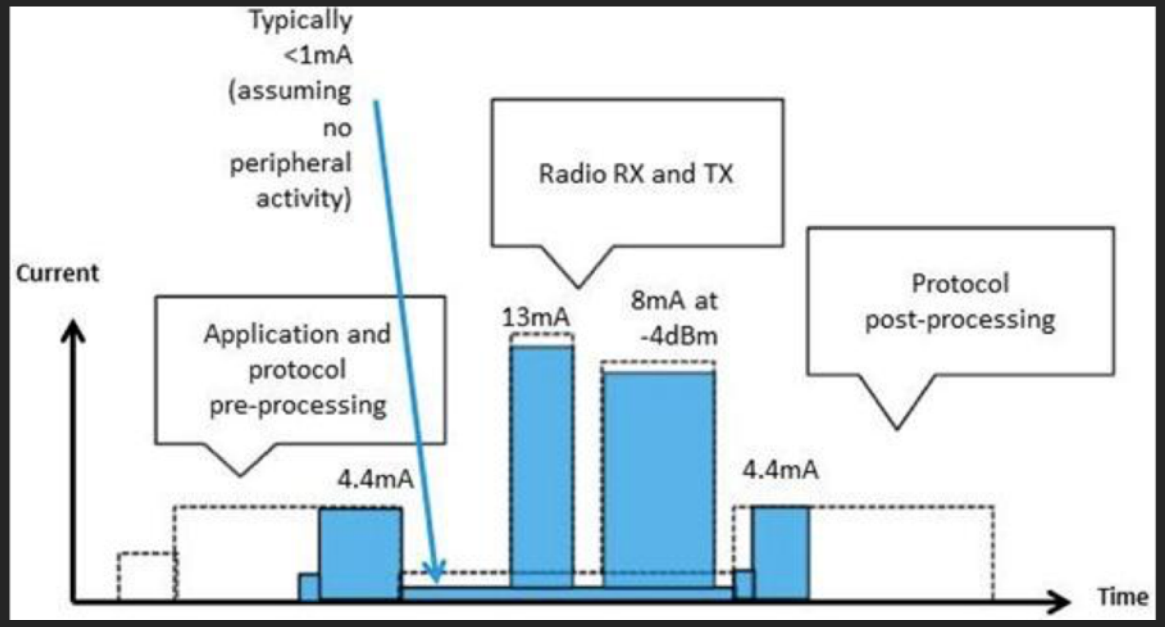
DON'T BE THAT ENGINEER…

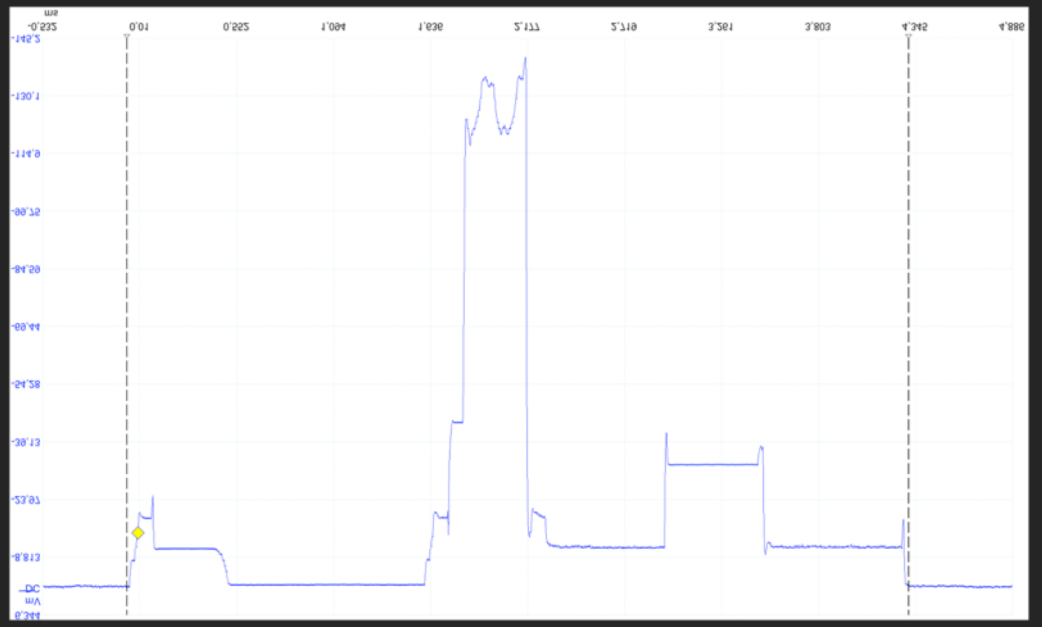
Feature Requests:

- Threat Modelling
- Compartments through spatial or temporal code isolation.
- Secure firmware updates and communication

Device **power consumption**: Transparency of device operations to attackers

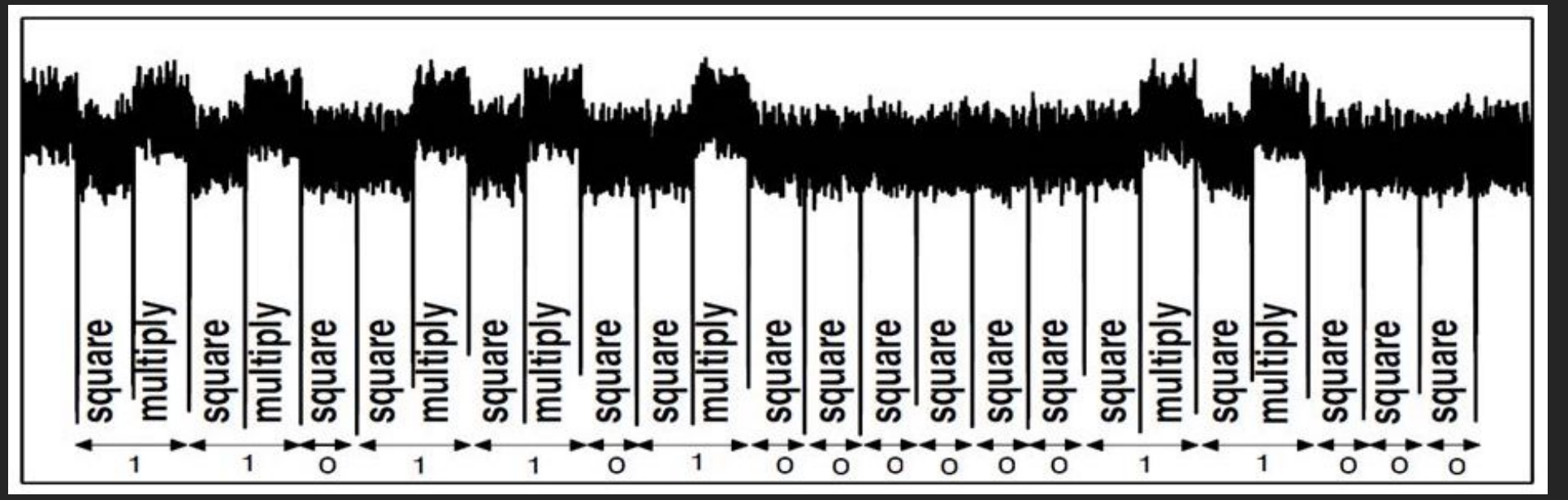






Device **power consumption**: Transparency of device operation to attackers

Power Analysis for Cheapskates
http://www.newae.com/files/OFLYNN_WhitePaper.pdf



Information Security – Theory vs. Reality: Power Analysis https://slideplayer.com/slide/9497570/

A system should be secure, **even if the crypto algorithm** and everything about the system - **except the key** - is made public.

KERCKHOFFS'S PRINCIPLE
BY AUGUSTE KERKHOFFS
DUTCH LINGUIST AND CRYPTOGRAPHER
1835-1903

A system should be secure, **even if the crypto algorithm** and everything about the system - **except the key** - is made public.

KERCKHOFFS'S PRINCIPLE
BY AUGUSTE KERKHOFFS
DUTCH LINGUIST AND CRYPTOGRAPHER
1835-1903

Feature Requests:

- Don't use class keys
- Allow key change on compromise
- Assume the attacker read your code

RANDOM NUMBER GENERATION

25

Random, or not?

APPLICATION DEVELOPERS AND SECURITY

Developers have trouble understanding the requirements for randomness - resulting in bad design choices.
Local attackers often control all external entropy sources or storage - and can reset the device at will.

TIME IS NOT RANDOM

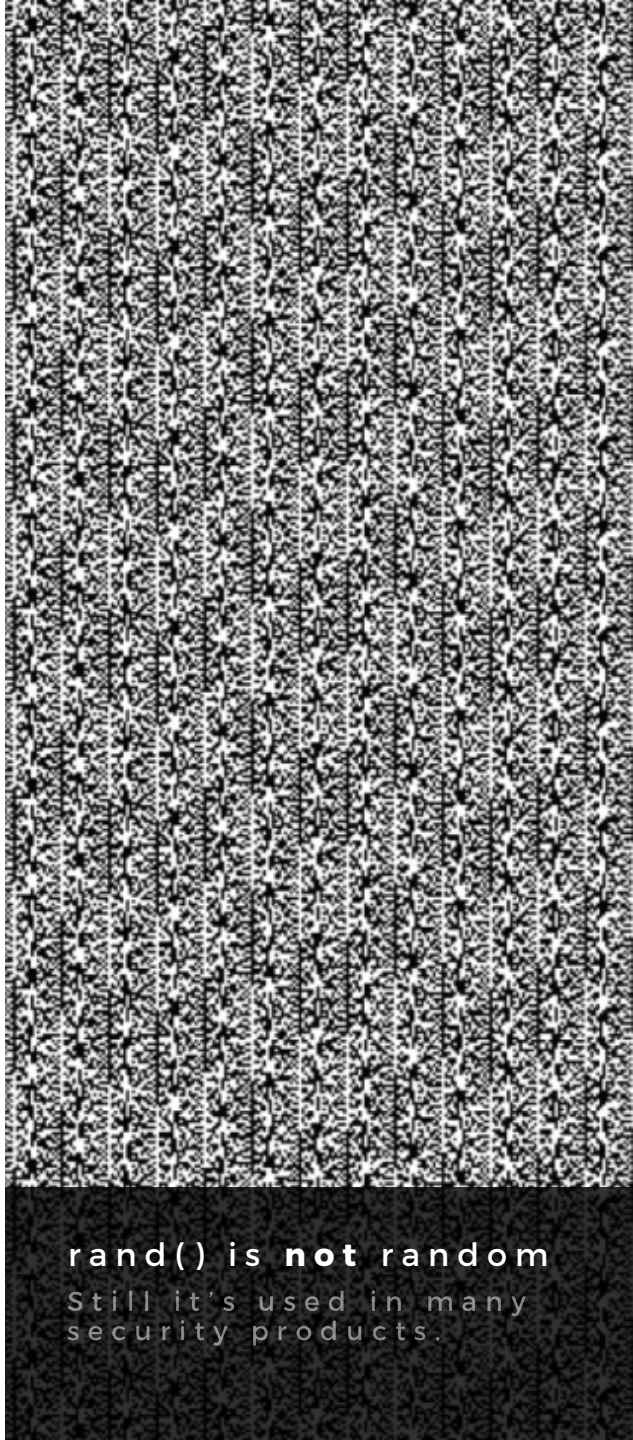
Assume that the attacker issues requests at precise times after booting your microcontroller and fully controls your clock sources & all hardware events.

PRNG vs. TRNG

Lack of a true random number generators microcontrollers is often used as an excuse for security flaws. Secure pseudorandom numbers are seldom implemented correctly.

PRNG SECURITY

PRNGs have often predictable seeds and are seldom device-unique.
Local attackers can often reset the PRNG generator by accessing storage or by reboot.



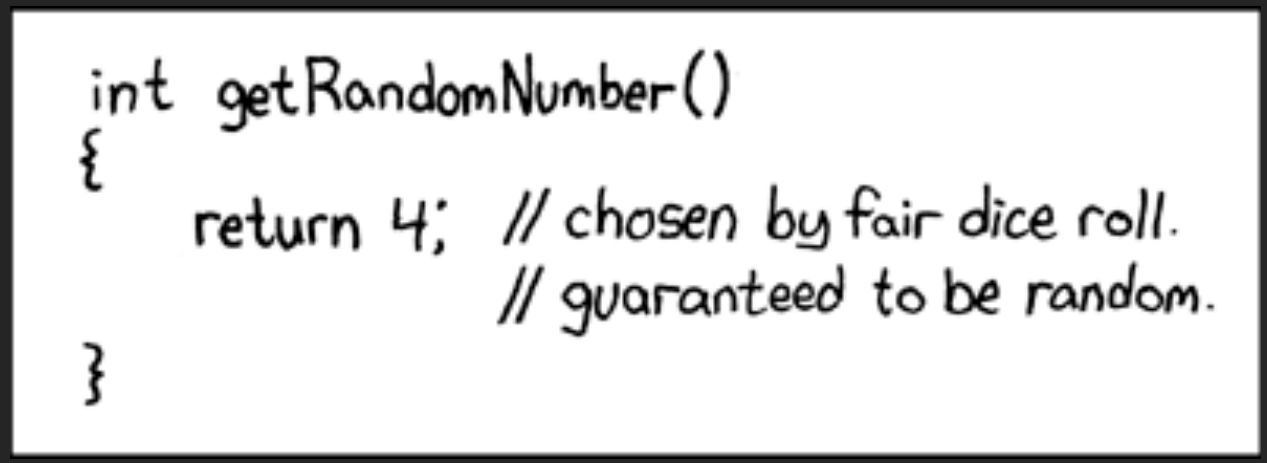
rand() is not random

Still it's used in many security products.

```
uint8_t key[104]; // pointless length
srand(time(NULL));
for (int i = 0; i < 104; i++) {
        key[i] = rand() & 0xff;
}
```

CODE FROM A DATABASE APPLICATION USED
BY THE GERMAN GOVERNMENT
FOR SECURITY AUDIT MANAGEMENT

Courtesy 30C3 Talk "Reverse engineering of CHIASMUS from GSTOOL"



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

CODE FROM XKCD.COM,
A WEBCOMIC OF ROMANCE, SARCASM,
MATH AND LANGUAGE (XKDC.COM/221)

```
uint8_t key[104]; // pointless length
srand(time(NULL));
for (int i = 0; i < 104; i++) {
        key[i] = rand() & 0xff;
}
```

CODE FROM A DATABASE APPLICATION USED
BY THE GERMAN GOVERNMENT
FOR SECURITY AUDIT MANAGEMENT

Courtesy 30C3 Talk "Reverse engineering of CHIASMUS from GSTOOL"

randomness
/ˈrandəmnəs/
the quality or state of **lacking a pattern** or principle of organization; **unpredictability**.

DON’T BE THE ENGINEER THAT BREAKS IT…

randomness
/ˈrandəmnəs/
the quality or state of **lacking a pattern** or principle of organization; **unpredictability**.

DON'T BE THE ENGINEER THAT BREAKS IT…

Feature Requests:

- Choose devices with TRNGs when possible.
- For PRNGs - use non-volatile pools, seeded at manufacturing and updated during device-runtime.
- Do not mistake long PRNG periodicity with security – use secure PRNGs instead.

CODE & DATA STORAGE

30

Storage, seriously?

YOUR SECURITY ASSUMPTIONS ARE PROBABLY BROKEN

Most high-end application/mobile CPUs have follow standardized security models for securing code and data. Microcontroller storage security is still highly diverse and commonly started off with code-read protection in mind. Yet, keys and firmware must be secure against tampering and rollback. Malware can't be allowed to become persistent by installing itself into flash.

OUT OF MEMORY

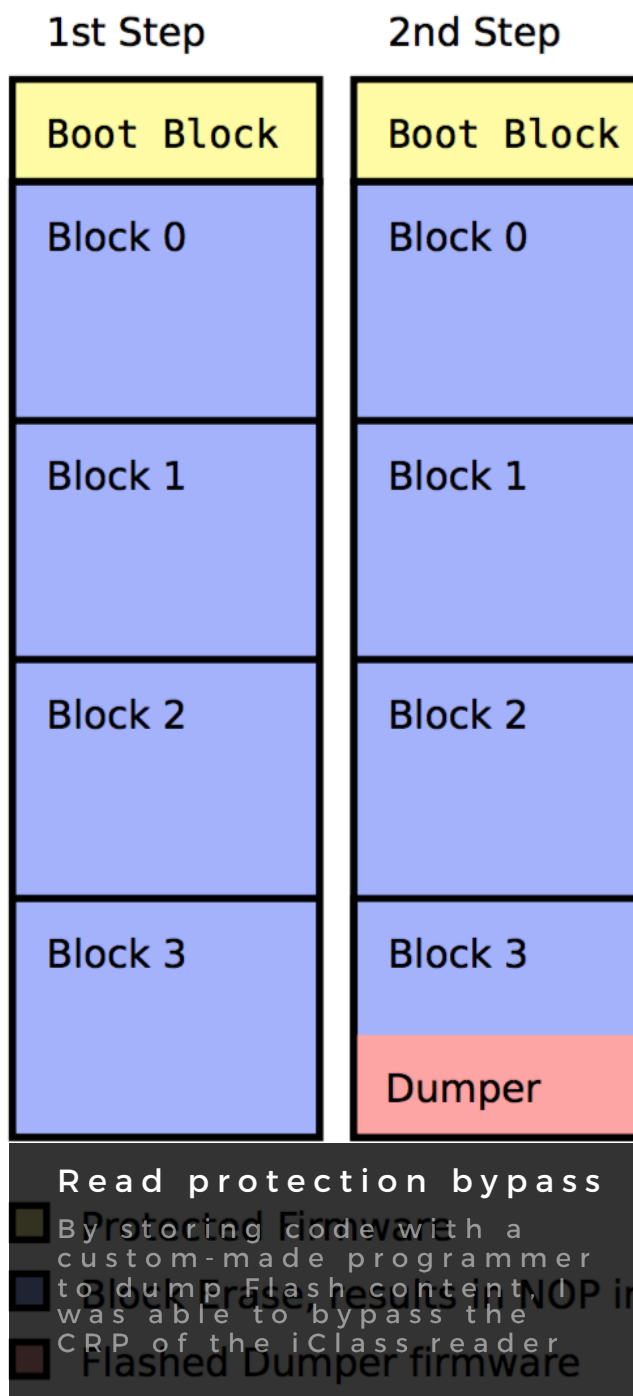
As wireless stacks and functionality grows, microcontrollers run out of code memory – firmware updates must often be buffered externally to avoid bricking,
These mechanisms are commonly flawed in the presence of local attackers controlling power or data.

DATA SECURE, TOO?

How confident are you, that your SRAM is erased when resetting the copy protection fuses on your microcontrollers?
Also data is often extracted indirectly using CPU core registers and stepping through existing code.

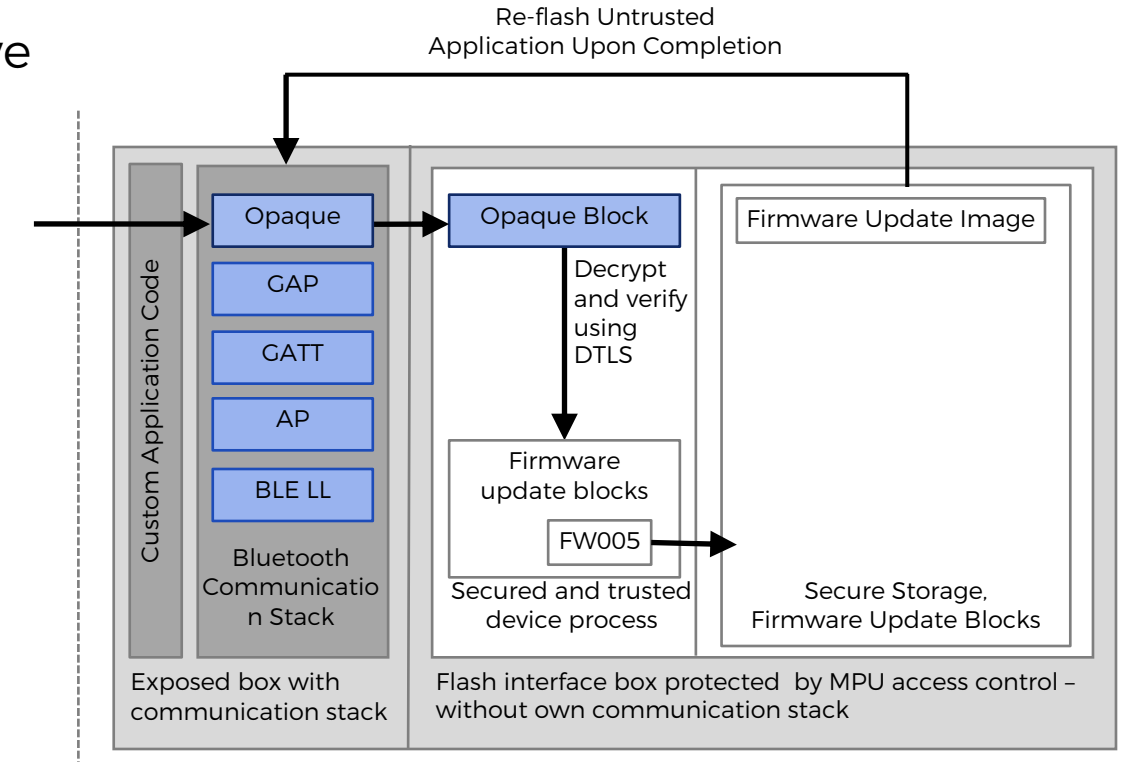
SIDE CHANNELS

Microcontrollers often vulnerable to extraction via side channels like power.



Case Study: Secure Firmware Update

- Flash-peripheral access is granted exclusive to the firmware update core-service.
- Uses the MPU for blocking access to the flash controller to everybody except the firmware update service code.
- Malware is forced to use APIs to attempt writing to flash memories:
  - Public-Key firmware signatures by the device owner or manufacturer are required by API to install updates.
  - Firmware is downloaded piece-by-piece into secure storage.
  - The system reboots after initial verification into a small boot-loader for copying the new firmware into its actual position of the internal flash.
  - The internal firmware is activated after final signature verification (temporal isolation!).



, Messages delivered independently of communication stacks

- Crypto-watchdog enforces remote control side-channel - even in presence of on-device malware controlling communication.
- Only the server can re-trigger device watchdog using the cryptographic secret.

"It ain't **what you don't know** that gets you into trouble. It's what you know for sure **that just ain't**."

MARK TWAIN

WRITER, ENTERPRENEUR & PUBLISHER

"It ain't **what you don't know** that gets you into trouble. It's what you know for sure **that just ain't**."

MARK TWAIN

WRITER, ENTERPRENEUR & PUBLISHER

Feature Requests:

- Understand the threat model of your hardware.
- Review hardware & software security in combination.
- Ask for 2nd pair of eyes security reviews for own designs and code.

Questions?

… and recommended reading:

- Arm Security Manifesto, Pages 4&5: goo.gl/3pWUPs
- Practical real-time operating system security for the masses, Pages 10ff: goo.gl/t6EP8U

Find me at the conference
or email at milosch@meriac.com

THANKS!