Feasibility of Security in Micro-Controllers

Aaron Ardiri

Chief Technology Officer - Evothings AB

jFokus IoT, Stockholm

3rd February, 2015

Overview

IoT Security

- why is it such a hot topic?
- why has it become an issue in the first place?
- what is the feasibility on the Arduino platform
- what is happening in the IoT developer ecosystem?
- food for thought: are we approaching it correctly?

IoT Security
why is it such a hot topic?

Hewlett Packard Report

HP's Fortify division recently tested a selection of IoT solutions currently available on the market by popular manufacturers including TVs, webcams, thermostats, power outlets, door locks and home control hubs.

- **250** vulnerabilities found
- **76%** of devices used unencrypted network resources
- **80%** failed to use strong passwords, many unchanged
- **60%** failed to protect firmware downloads/integrity

Sweden - National Security

3rd November 2014

http://www.dn.se/nyheter/sverige/it-expert-bristerna-ett-hot-mot-rikets-sakerhet/

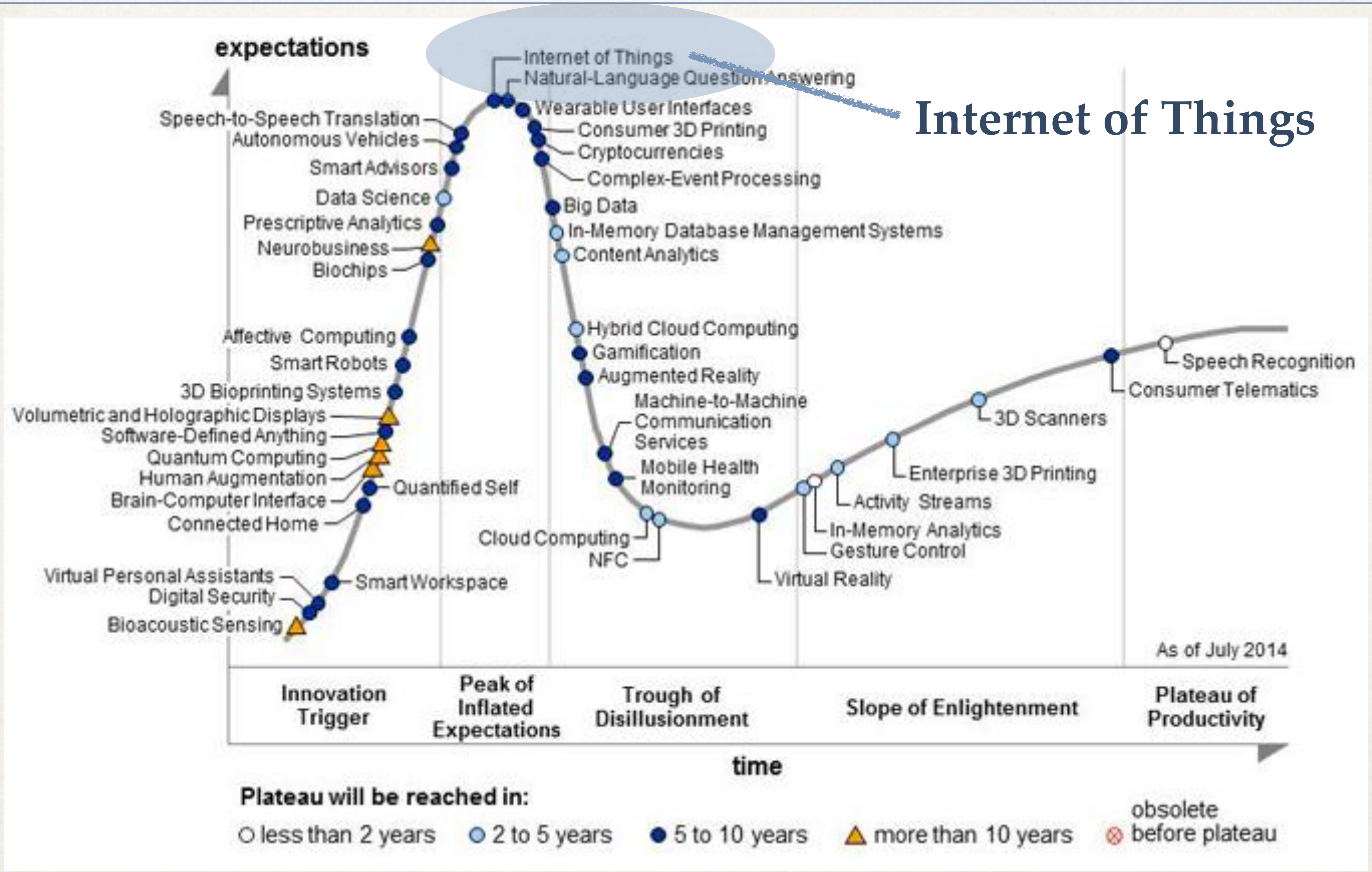
It was revealed a number of important public properties in Sweden including but not limited to Police Stations, Transit Stations, Data Centers and Space Center in Kiruna are completely open on the Internet and hackable - control of alarms, doors, heating, other sensitive systems

- sites are password protected but have weak security

IoT Security
why has it become such an issue?

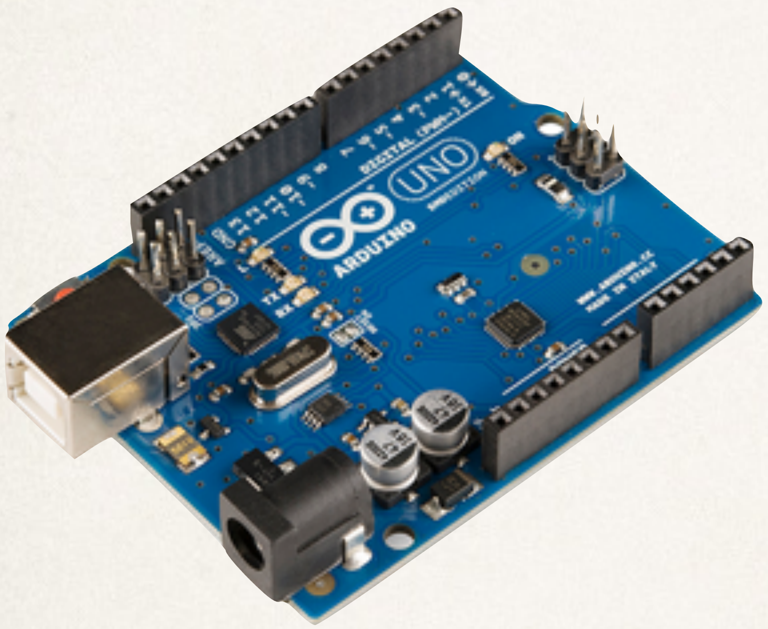
Gartner Hype Cycle Special Report

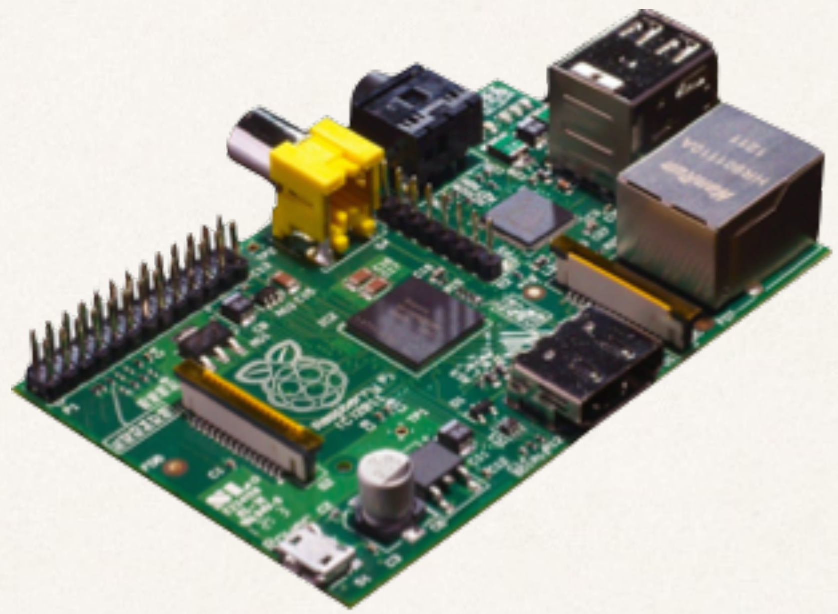
http://www.gartner.com/newsroom/id/2819918

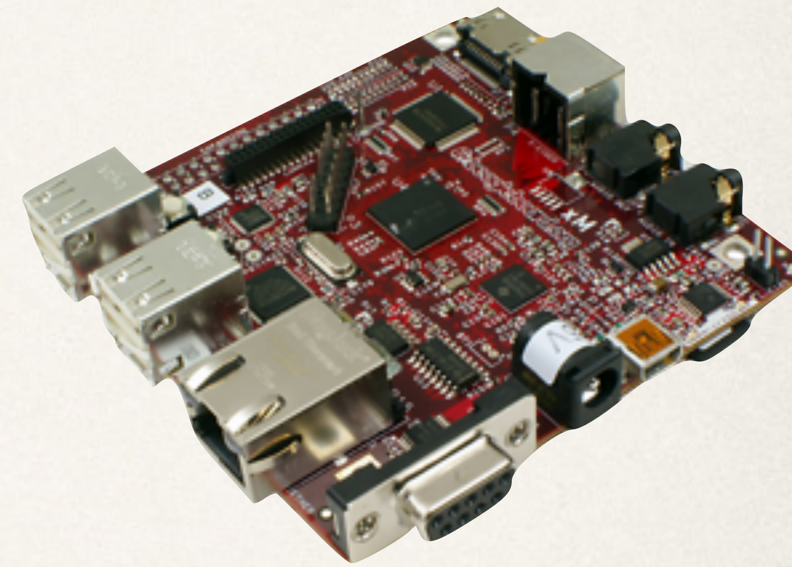


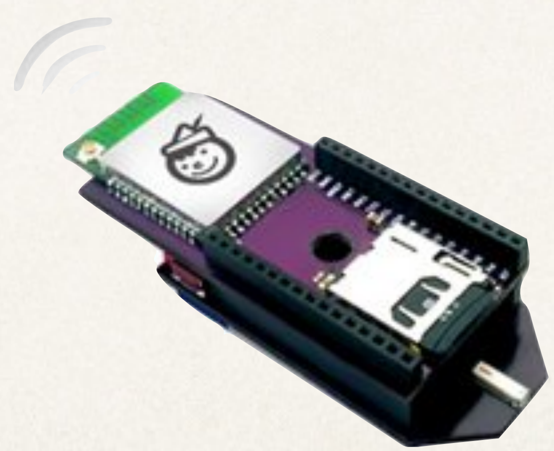
Internet of Things

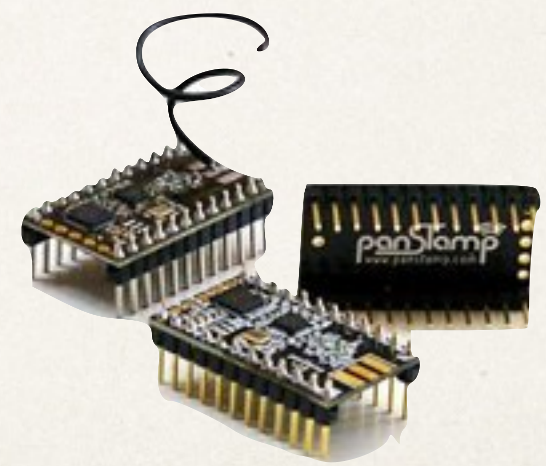
Explosion of micro-controllers











postscapes.com/internet-of-things-hardware

Products - what is happening

- companies making “land grab” in IoT space
  - focus is product-to-market, not deliver quality
  - a number of products are based on prototypes
  - failure to provide OTA and update mechanisms
- SSL/TLS - implementations
  - many micro-controllers have limited CPU / RAM
  - existing libraries are not optimised for embedded

Standards War

Hyper/Cat

intel
SAMSUNG
BROADCOM
WIND RIVER

OPEN INTERCONNECT CONSORTIUM

AllJoyn
QUALCOMM
Microsoft
CISCO
Panasonic

ALLSEEN ALLIANCE

IEEE Internet of Things

Google

0-day exploits (security) in 2014

Heartbleed

serious vulnerability in the popular OpenSSL cryptographic software library.

ShellShock

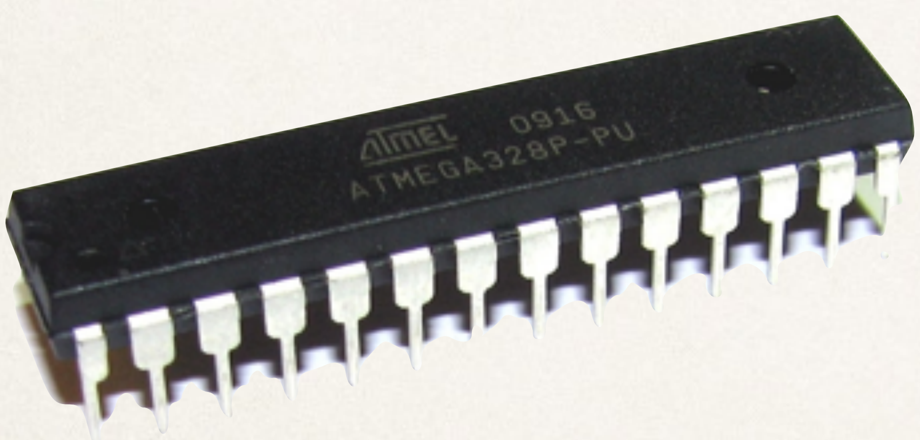
aka: Bashdoor group of bugs in the popular Bourne Again Shell (Bash).

POODLE

serious vulnerability in the popular OpenSSL cryptographic software library.

Operating Systems

- What are the options for IoT product manufacturers?

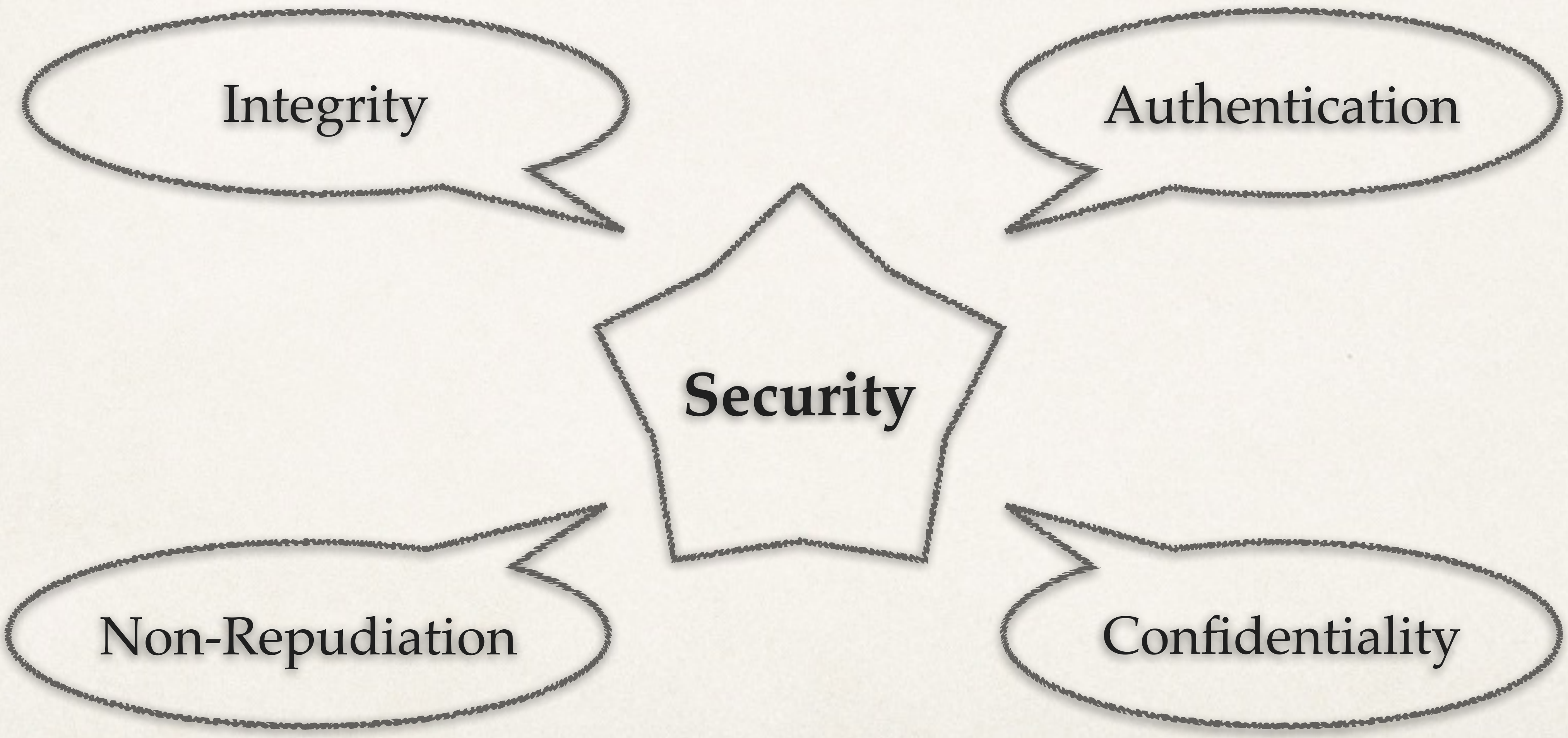


BareBones OS

or

Security is not only encryption

A common mis-conception; it is more than Encryption



http://www.securerf.com/security-is-not-encrypting-data/

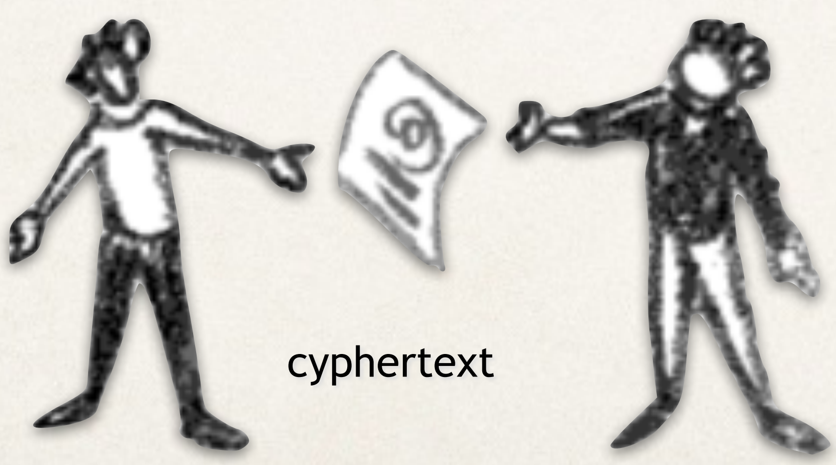
IoT Security
what is the feasibility on Arduino?

Public Key Cryptography

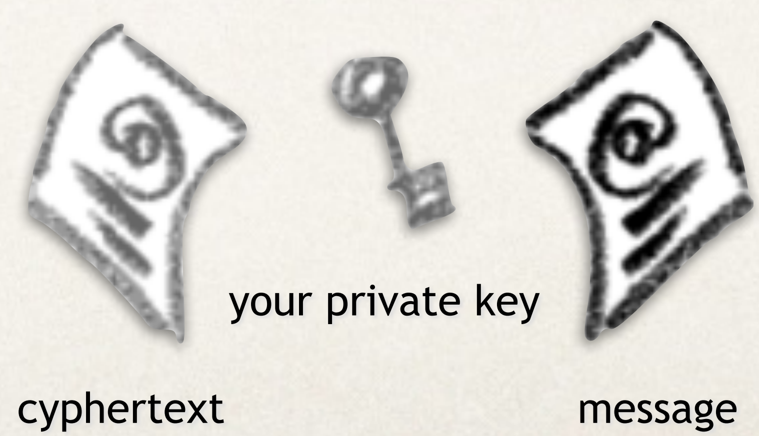
#1 provide your public key to sender

#2 sender uses your public key to encrypt message

#3 sender provides cyphertext to you



#4 use your private key to decrypt cyphertext



RSA: Basic Overview

- encryption

  $c = m^e \bmod n$
- decryption

  $m = c^d \bmod n$

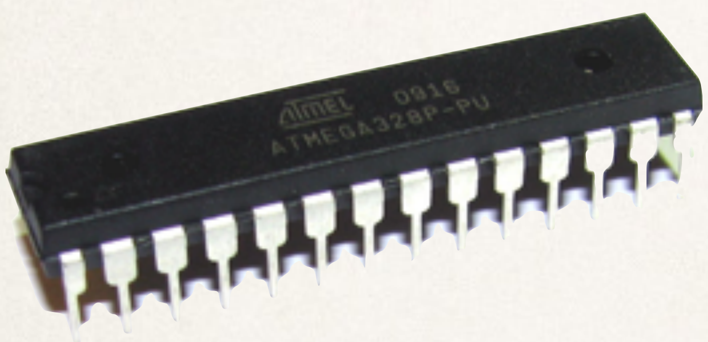
ALGORITHM KEY

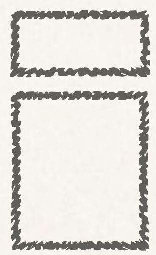
m = original message
c = cyphertext

e = public key exponent
d = private key exponent
n = modulus (primes multiplied)

the source text is to converted to an integer form that is then passed through the exponent modulus algorithm to create a second integer that can then be converted into a cyphertext string to be transmitted over the network.

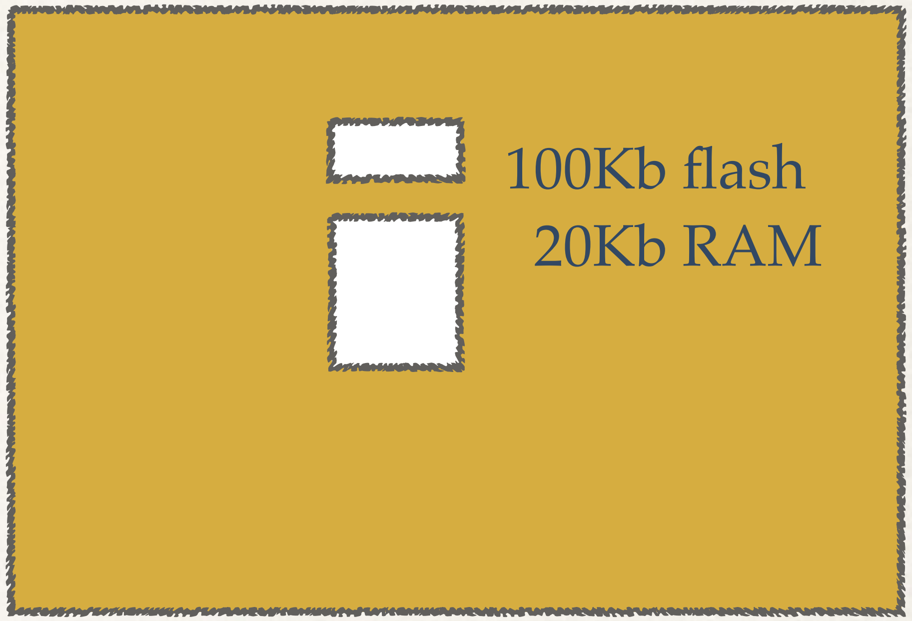
SSL: Market offering





32Kb flash
2Kb RAM

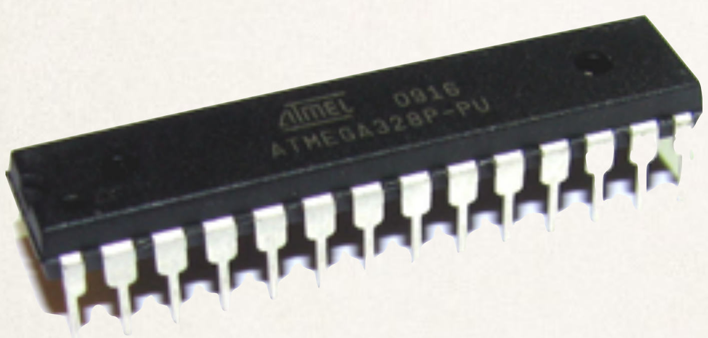
embedded version

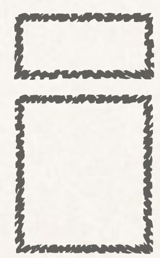


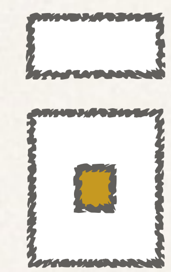
- SSL/TLS - implementations
  - many micro-controllers have limited CPU / RAM
  - existing libraries are simply too large for embedded

SSL: Evothings investigation

Secure







- feasibility study, specific to Arduino
- working prototype of end-to-end setup
  - encryption/decryption using RSA-1024

RSA implementation on Arduino

implementation of RSA encrypt/decrypt algorithms:

- custom written - mixture of C and assembly (avr only)
  - implemented specifically for RSA algorithms
  - keys are defined as (n,e) and (n,d) raw bit streams
  - designed to be portable with a small code footprint
- 128, 256, 512, 1024 and 2048 keys (if possible)
  - limited SRAM of micro controller restricts key sizes

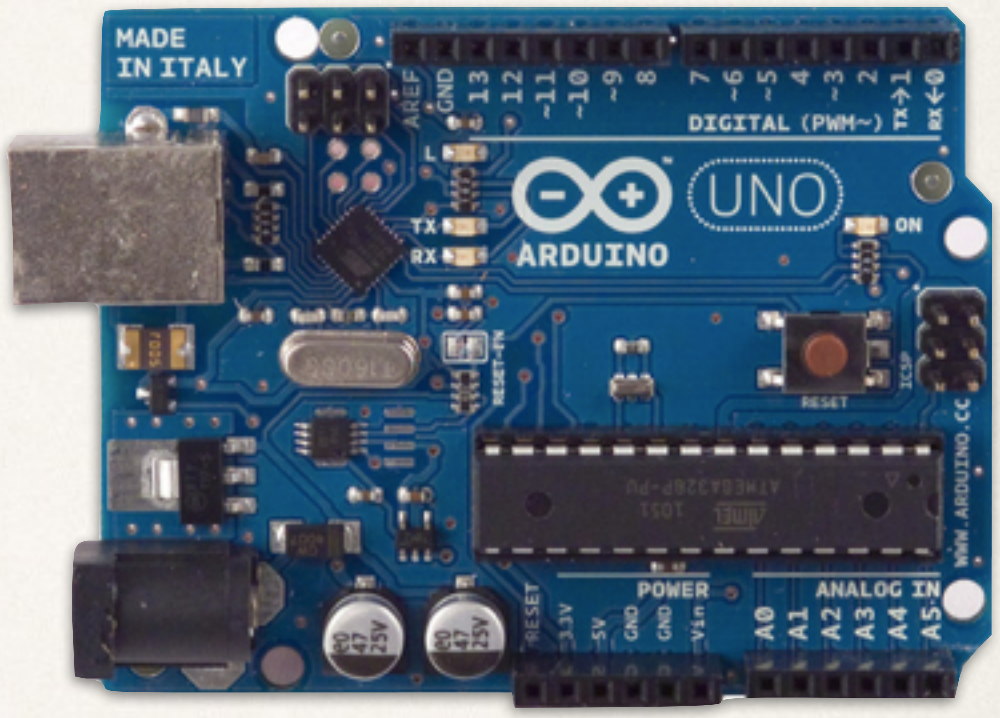
RSA implementation on Arduino

```
BigInt e, d, n, m, c;

// define our public(n,e), private(n,d) and message
BigInt_assignFromBuffer(&d, (unsigned char *)key_device_prv);
BigInt_assignFromBuffer(&e, (unsigned char *)key_device_pub);
BigInt_assignFromBuffer(&n, (unsigned char *)key_device_mod);
BigInt_assignFromBuffer(&m, (unsigned char *)rsa_message);

// encrypt message ‘m’ into cypher text ‘c’
BigInt_exponent_with_modulus(&c, &m, &e, &n);

// decrypt cypher text ‘c’ into message ‘m’
BigInt_exponent_with_modulus(&m, &c, &d, &n);
```

IoT Security
feasibility - results on the Arduino

RSA: Arduino UNO

- CPU
  - ATmega328
  - 16Mhz
- 32Kb program mem
- 2Kb SRAM



Performance Results (ms) - compiled with 8bit, pure C

algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	288	1070	4103	16160	N/A*
decrypt: private key	3155	22365	175452	1383240	N/A*

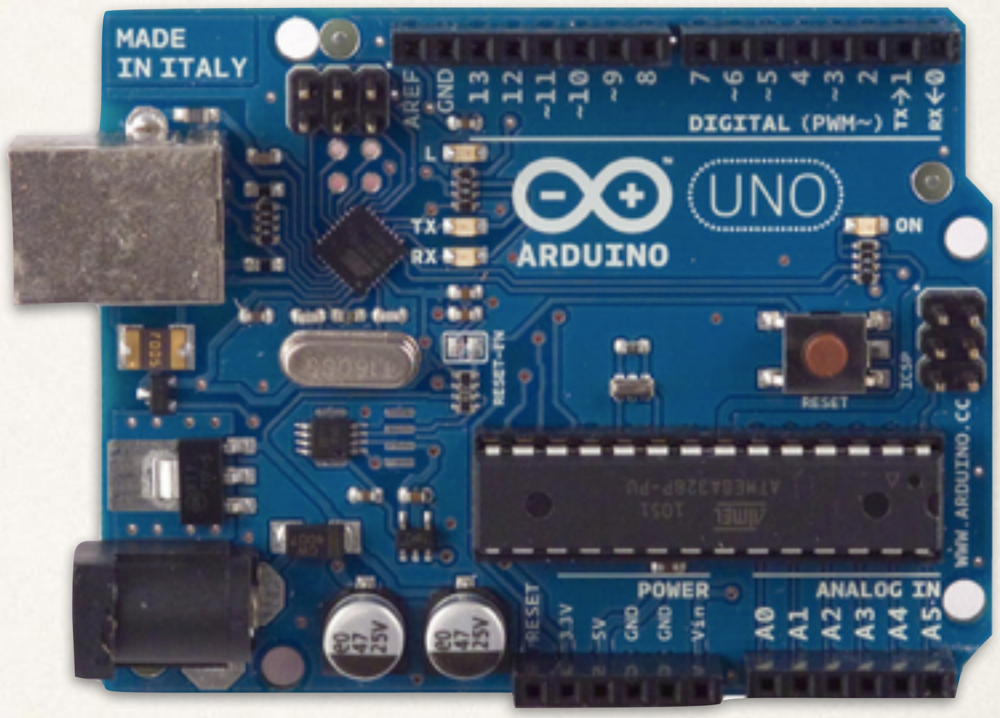
* insufficient SRAM to perform

RSA: Arduino UNO

48%

performance boost

- CPU
  - ATmega328
  - 16Mhz
- 32Kb program mem
- 2Kb SRAM



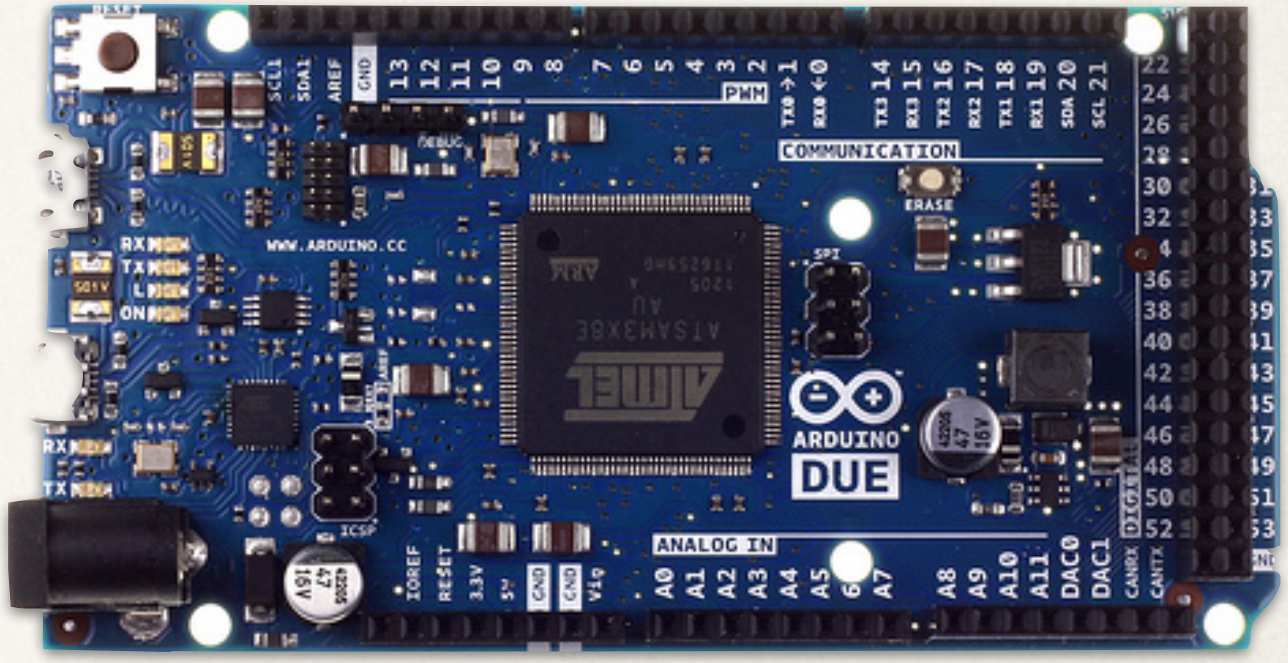
Performance Results (ms) - compiled with 8bit, avr asm

algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	178	609	2225	8504	N/A*
decrypt: private key	1951	12716	95079	727955	N/A*

* insufficient SRAM to perform

RSA: Arduino Due

- CPU
  - AT91SAMX8E
  - 84Mhz
- 512Kb program mem
- 96Kb SRAM



Performance Results (ms) - compiled with 32bit, 100% C

algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	25	77	264	1032	4122
decrypt: private key	261	1586	11206	88216	701668

RSA: Arduino Yún

* use Bridge Library to execute RSA algorithms on Linux CPU

- CPU
  - ATmega32U4 and AR9331
  - 16Mhz and 400Mhz
- 32Kb program mem
- 2.5Kb SRAM

Performance Results (ms) - compiled with 32bit, 100% C

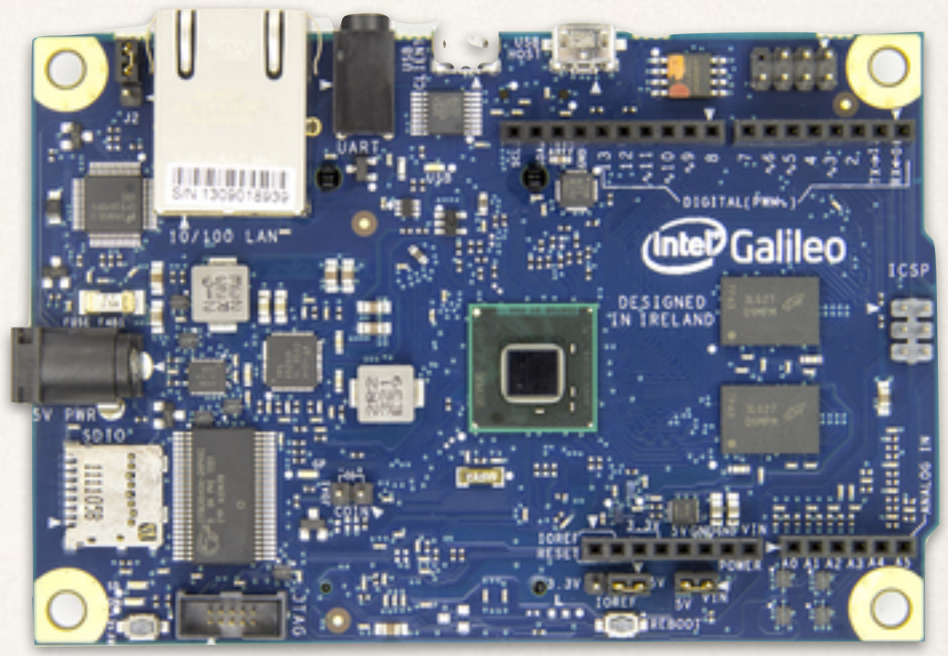
algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	329	355	512	707	N/A*
decrypt: private key	437	562	1681	10799	N/A*

* insufficient SRAM to perform

* the Bridge implementation has a 100-200ms fluctuation in results depending on key size

RSA: Intel Galileo

- CPU
  - Quark SoC X1000
  - 400Mhz
- 256Kb program mem
- 512Kb SRAM

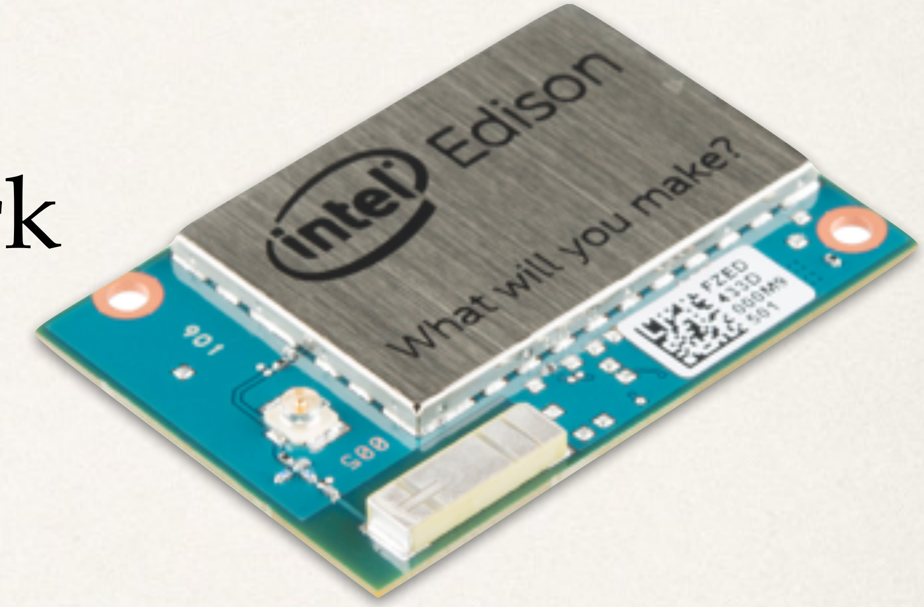


Performance Results (ms) - compiled with 32bit, 100% C

algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	4	20	57	192	706
decrypt: private key	95	397	2310	16055	119499

RSA: Intel Edison

- CPU
  - dual core Atom SoC and Quark
  - 500Mhz and 100Mhz
- 10Mb program mem
- 1Gb SRAM



Performance Results (ms) - compiled with 32bit, 100% C

algorithm	128 bit	256 bit	512 bit	1024 bit	2048 bit
encrypt: public key	3	7	23	76	273
decrypt: private key	30	150	976	6548	46579

RSA 1024: Resource Usage (avr)

empty sketch:

Sketch uses 450 bytes (1%) of program storage space.

Maximum is 32,256 bytes.

Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for local variables. Maximum is 2,048 bytes.

RSA 1024 with public key only

Sketch uses 4,116 bytes (12%) of program storage space.

Maximum is 32,256 bytes.

Global variables use 981 bytes (47%) of dynamic memory, leaving 1,067 bytes for local variables. Maximum is 2,048 bytes.

resulting code size:

3,666 bytes of program storage space

972 bytes of dynamic memory

~ 3.5Kb for code, < 1Kb for RAM

RSA 1024: Resource Usage (ARM)

empty sketch:

Sketch uses 10,492 bytes (2%) of program storage space.

Maximum is 524,288 bytes.

Global variables use 9 bytes (0%) of dynamic memory, leaving 98,295 bytes for local variables. Maximum is 98,304 bytes.

RSA 1024 with public key only

Sketch uses 12,836 bytes (2%) of program storage space.

Maximum is 524,288 bytes.

Global variables use 981 bytes (0%) of dynamic memory, leaving 97,323 bytes for local variables. Maximum is 98,304 bytes.

resulting code size:

1,454 bytes of program storage space

972 bytes of dynamic memory

~ 1.4Kb for code, < 1Kb for RAM

RSA 1024: Resource Usage (x86)

empty sketch:

Sketch uses 55,375 bytes (21%) of program storage space.

Maximum is 262,144 bytes.

Global variables use 9 bytes (0%) of dynamic memory, leaving 524,279 bytes for local variables. Maximum is 524,288 bytes.

RSA 1024 with public key only

Sketch uses 63,805 bytes (24%) of program storage space.

Maximum is 262,144 bytes.

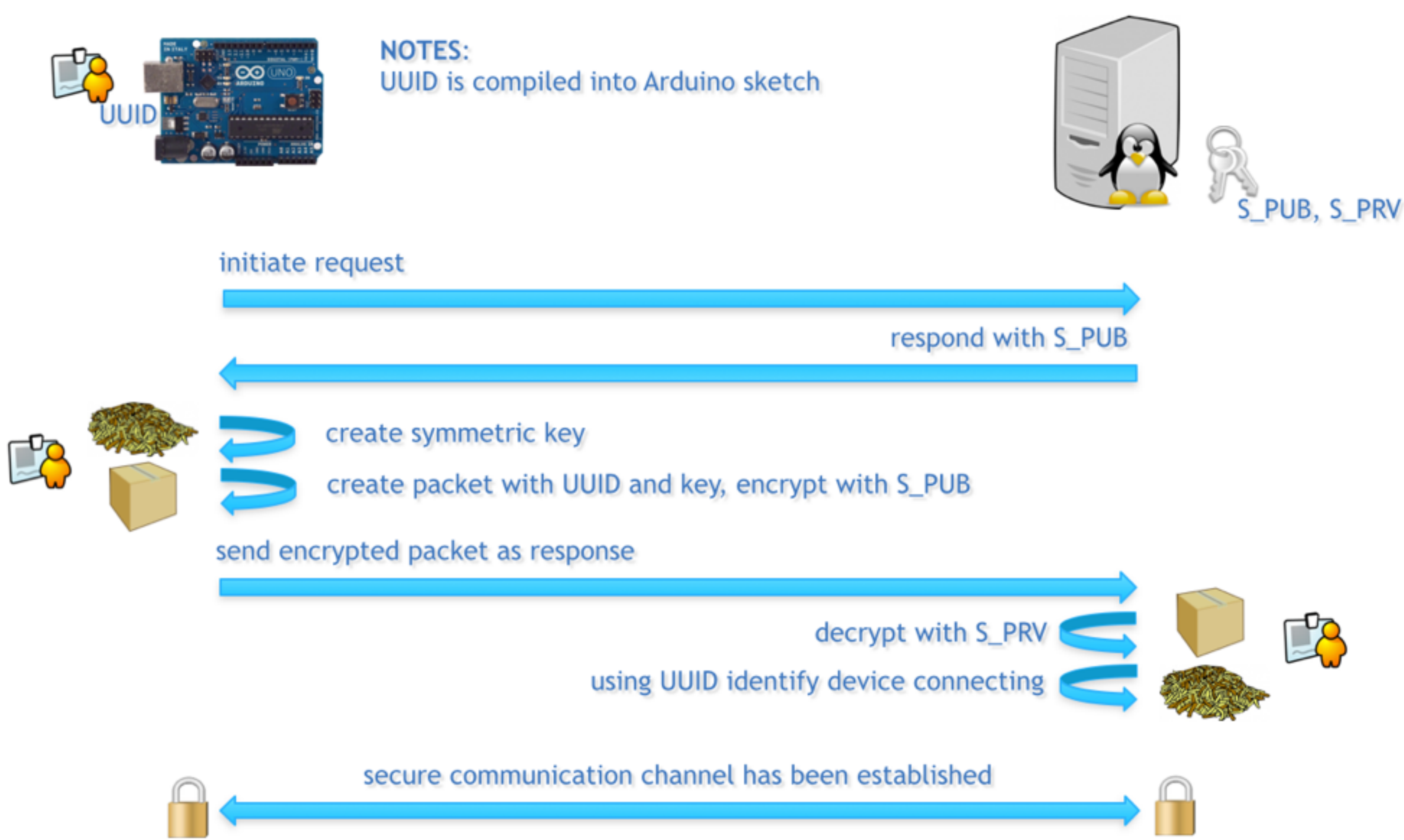
Global variables use 981 bytes (0%) of dynamic memory, leaving 523,307 bytes for local variables. Maximum is 524,288 bytes.

resulting code size:

8,430 bytes of program storage space

972 bytes of dynamic memory

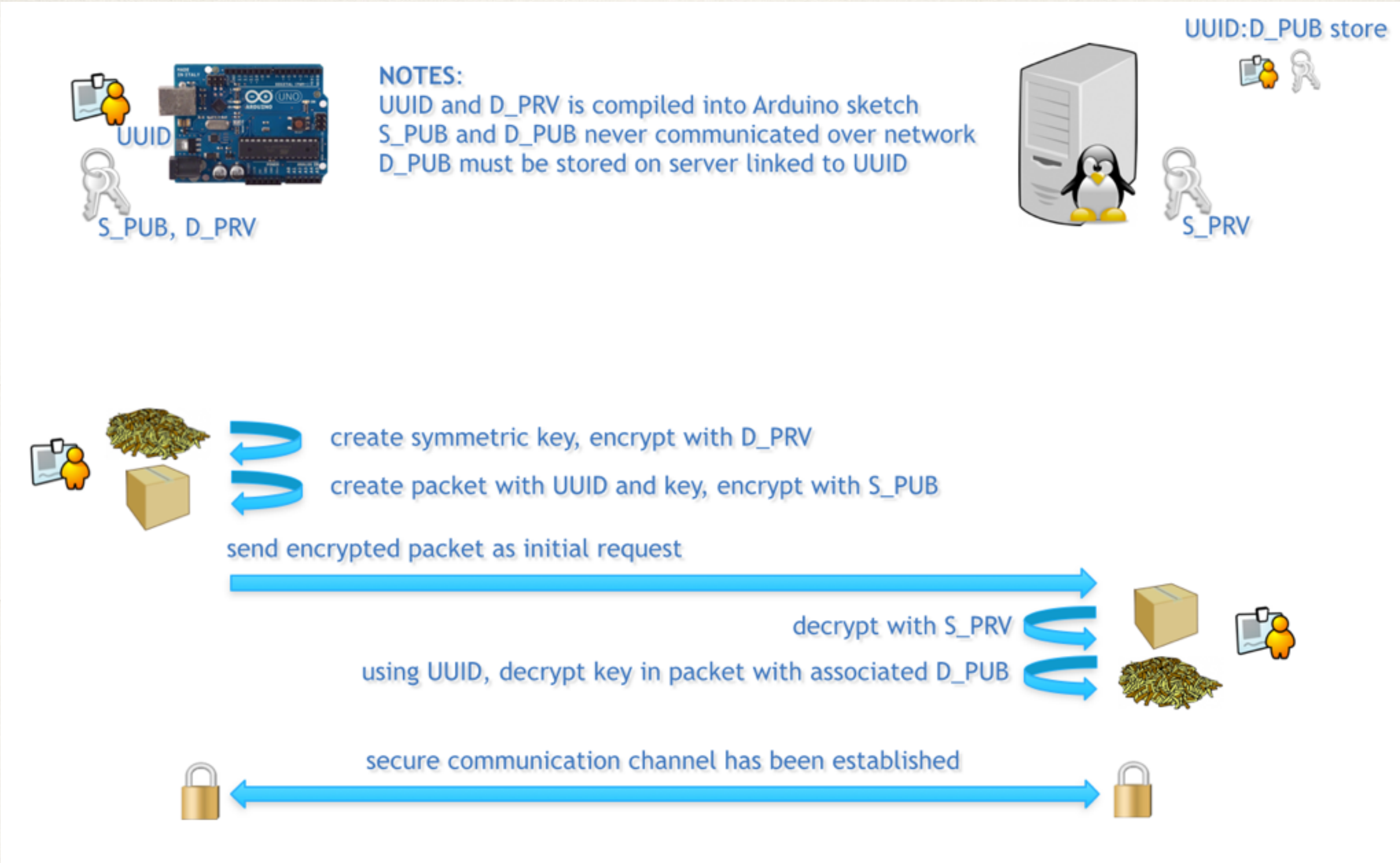
~ 8.2Kb for code, < 1Kb for RAM


NOTES:
UUID is compiled into Arduino sketch
UUID
S_PUB, S_PRV
initiate request
respond with S_PUB
create symmetric key
create packet with UUID and key, encrypt with S_PUB
send encrypted packet as response
decrypt with S_PRV
using UUID identify device connecting
secure communication channel has been established

Configuration Analysis

- Advantages
  - S_PUB can be dynamic between sessions
  - only S_PUB used for encryption, low CPU demands

- Disadvantages
  - S_PUB is communicated over network
  - no good method to validate that the server the device is talking to is authentic (no CA validation)


UUID:D_PUB store
NOTES:
UUID and D_PRV is compiled into Arduino sketch
S_PUB and D_PUB never communicated over network
D_PUB must be stored on server linked to UUID
UUID
S_PUB, D_PRV
S_PRV
create symmetric key, encrypt with D_PRV
create packet with UUID and key, encrypt with S_PUB
send encrypted packet as initial request
decrypt with S_PRV
using UUID, decrypt key in packet with associated D_PUB
secure communication channel has been established

Configuration Analysis

- Advantages
  - S_PUB, D_PUB never communicated over network
  - D_PUB is stored on server, associated to UUID
    - only Arduino's registered can communicate with server
    - can remove any "compromised" devices from server

- Disadvantages
  - D_PRV is used to encryption on device = slower

Secure Random Number Generator

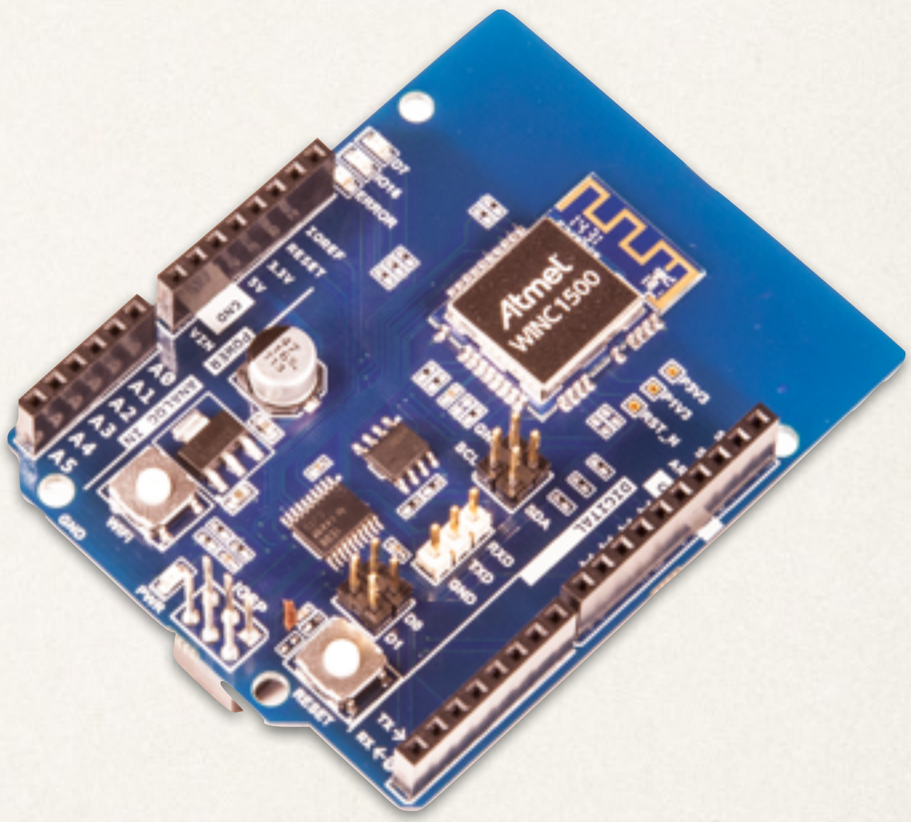
- Arduino devices provide at least one analog pin that can be used to create secure random numbers critical for symmetric keys (AES).
- 2-pass von Neumann algorithm to remove “bias” from analog feed
- re-use the existing PRNG random(), seeding at random intervals

```
int secureRandomByte()
{
  static int count = 0;
  static int next  = (randomByte() >> 2) + 1; // max 64 iterations
  if ((count++ % next) == 0)
    { randomSeed(randomWord()); next = (randomByte() >> 2) + 1; }
  return random(256);
}
```

IoT Security
what is the happening the IoT ecosystem?

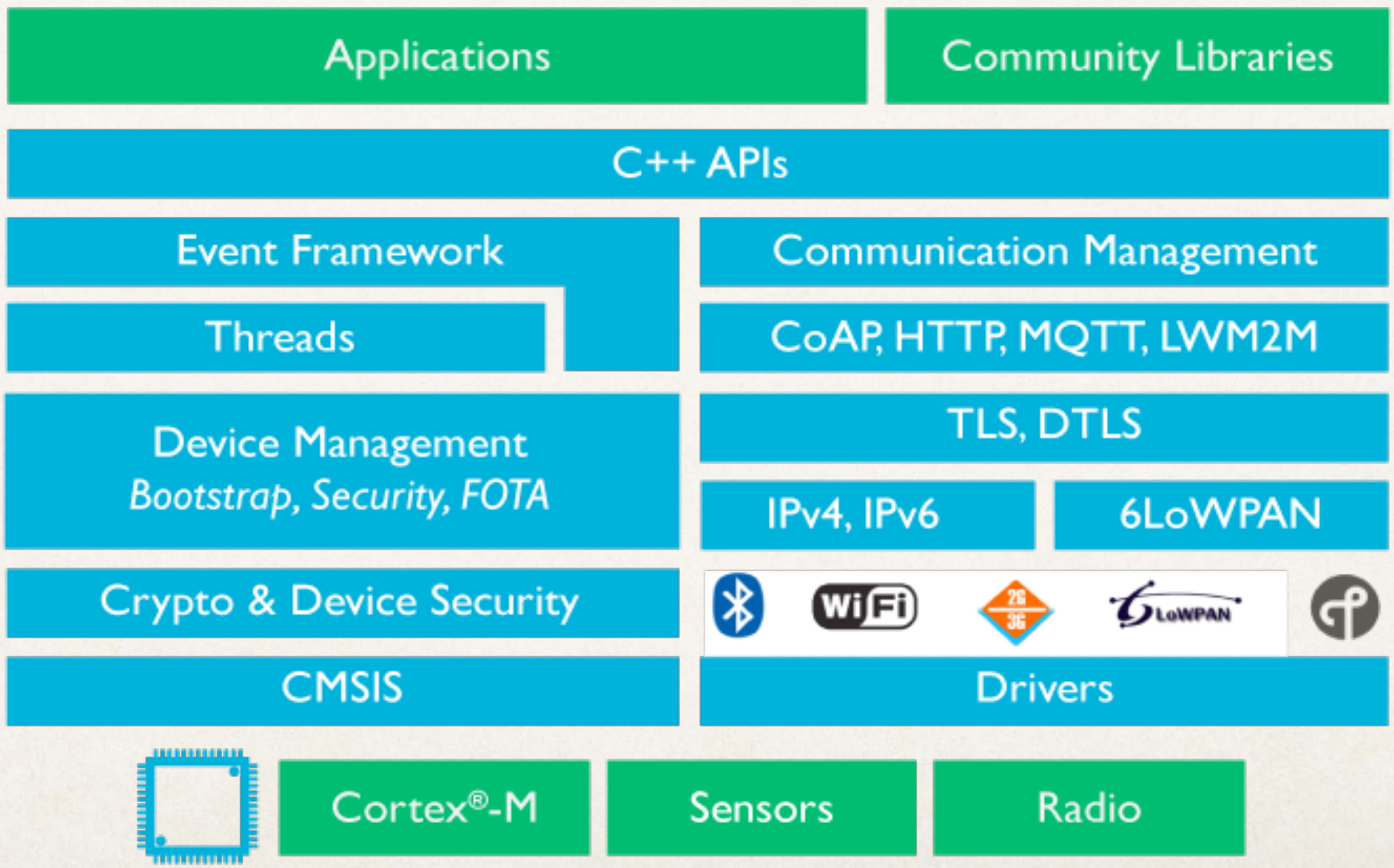
Arduino + Secure Wifi Shield

- WiFi shield with integrated WINC1500 processor
- TLS provided using:
  - ECC-256 (eq to RSA-3072)
  - AES-128
  - SHA-256



mbed OS - ARM

- open source: code/framework designed for ARM CPU



libCommas - avr

https://saifeinc.com/news/?p=223

- open source: code/framework designed for avr
- ECC (ECDSA) and SHA-2 algorithms
- proprietary server for communication end-point

AVR crypto-lib - avr

http://www.das-labor.org/wiki/AVR-Crypto-Lib/en

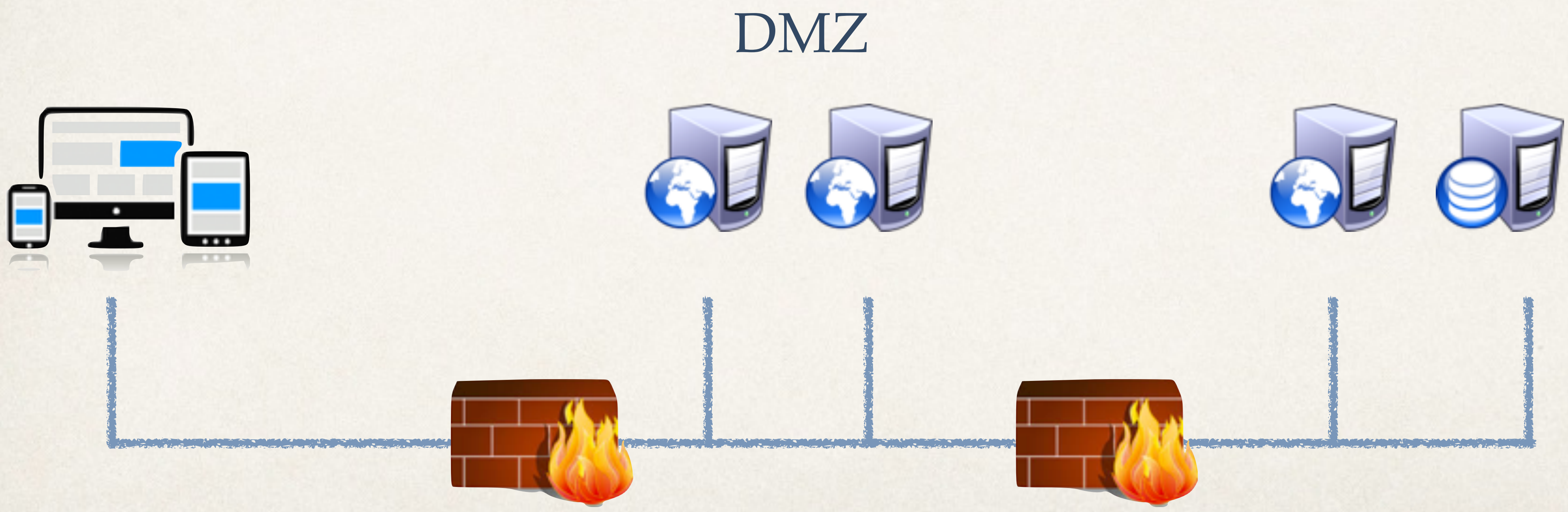
- open source: code/framework designed for avr
- various block, stream cyphers and hash functions

AVR-Crypto-Lib

IoT Security
are we approaching it the right way?

Security Foundations: Classic

Security in computing has been typically bound to the security of the real-world - by defining elements such as keys, trusted-zones (DMZ), firewalls et al.



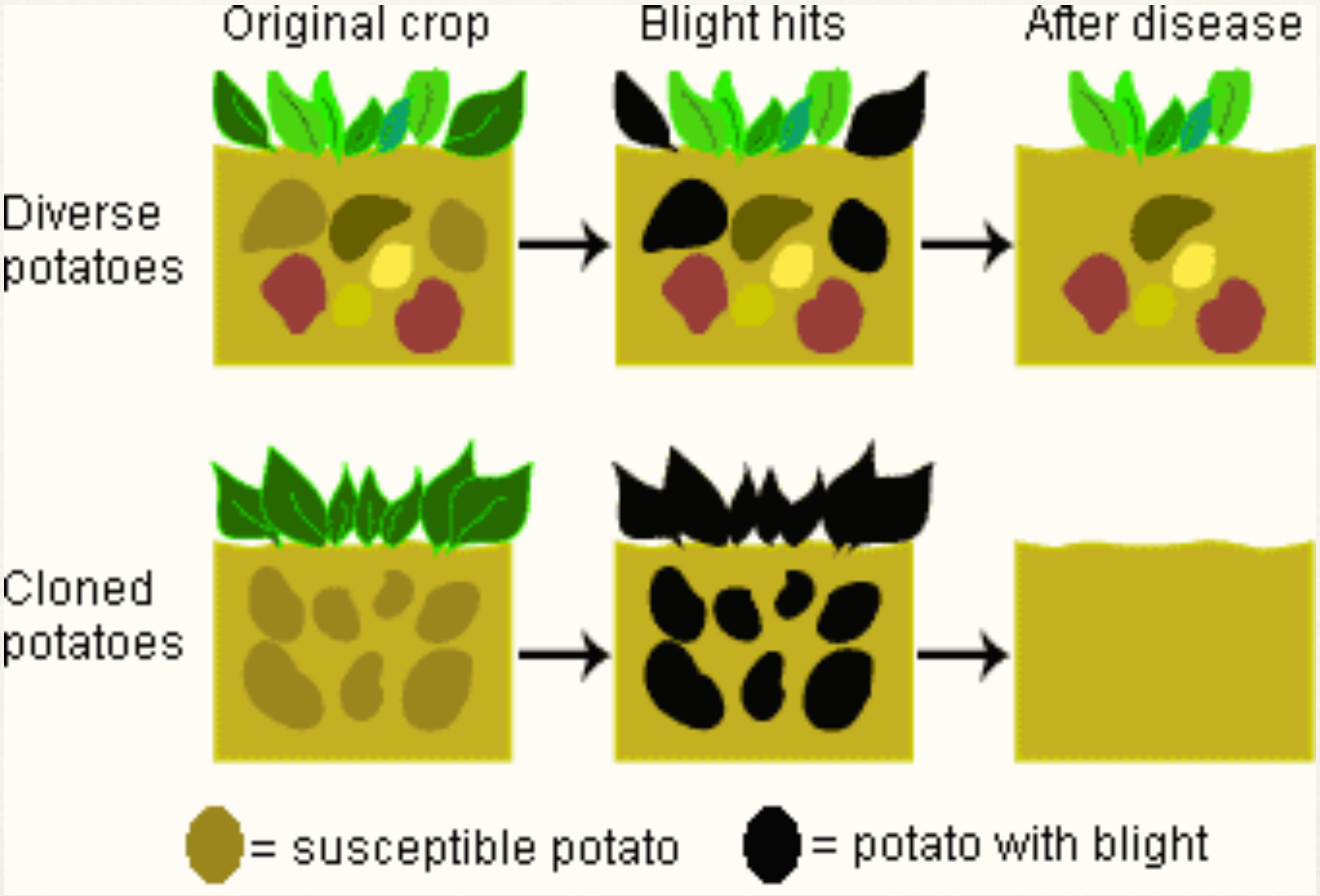
Security Foundations: Biology

Researchers have considered following nature's design and look at security from with a biological mindset - where devices would be open to infection and evolve.

Immunological defence based on identification and isolation of a threat with backup nodes to spawn off to fulfil the function of compromised nodes.

http://www.eetindia.co.in/ART_8800705403_1800001_NT_a11862e6.HTM

Importance of Diversity in Nature



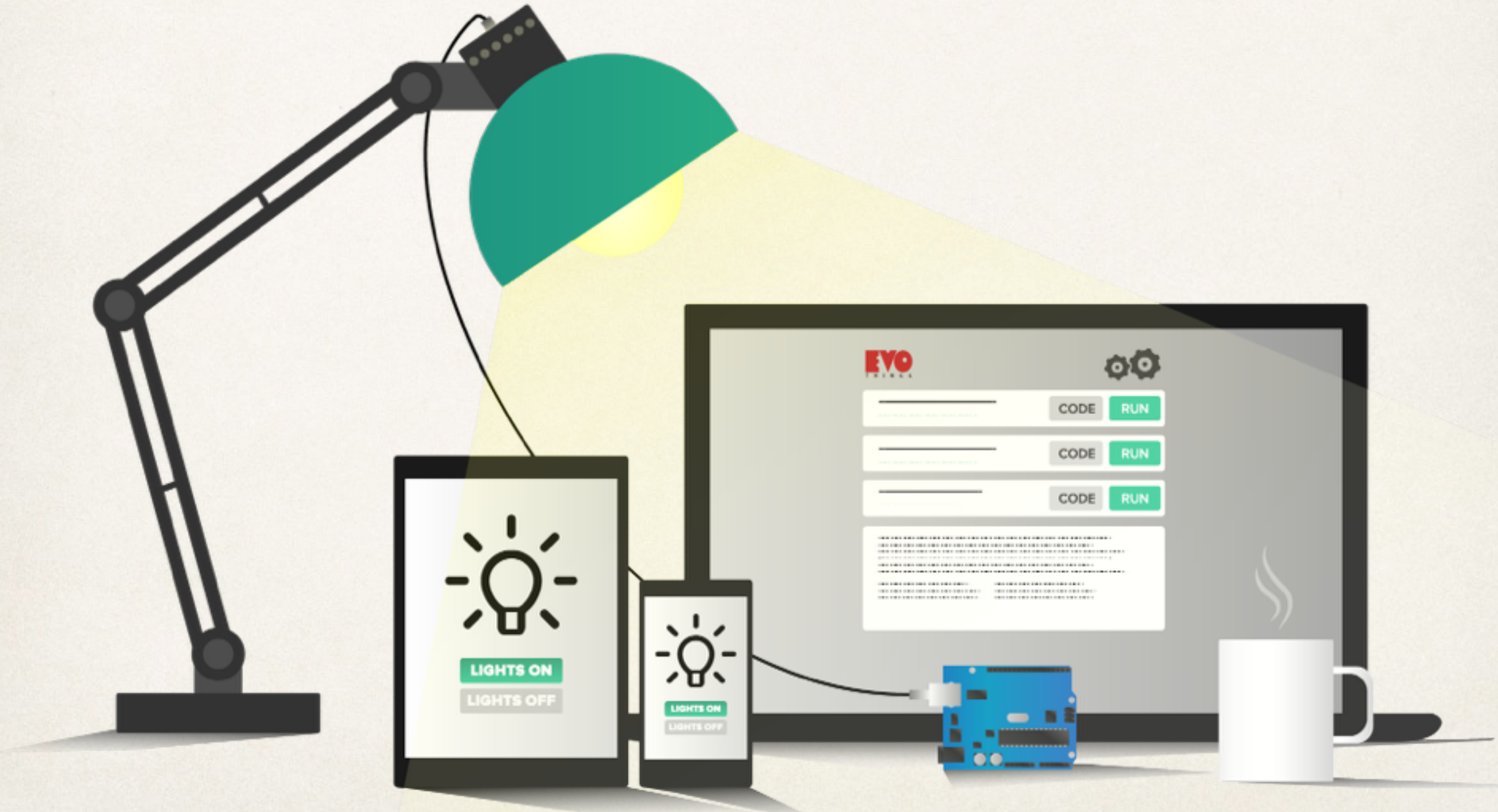
http://evolution.berkeley.edu/evolibrary/article/agriculture_02

How many IoT devices by 2020?

Gartner:	26 Billion	**Cisco:**	50 Billion
Intel:	200 Billion	**IDC:**	220 Billion

It's time to **act** now and ensure Security exists within IoT

Evothings Studio



Questions

Contact Information

aaron.ardiri@evothings.com

twitter: @ardiri

www.evothings.com

www.ardiri.com/blog