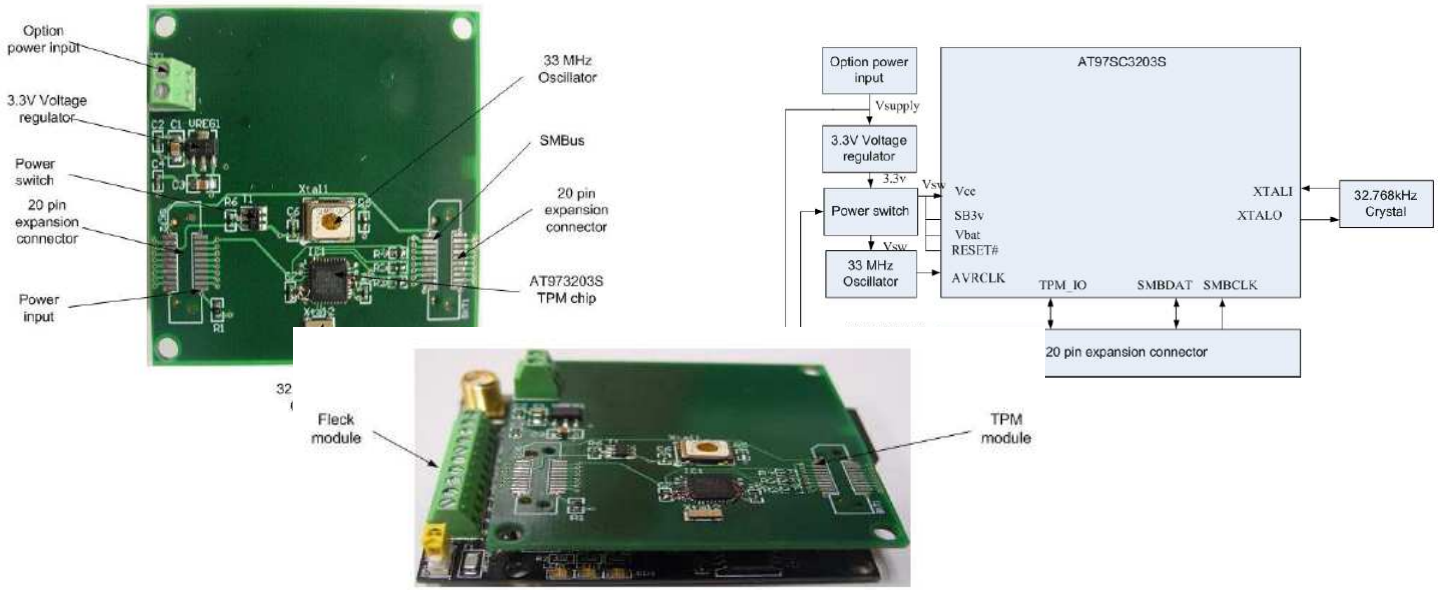


www.csiro.au

A Public Key Infrastructure for Wireless Sensor Networks

Dr. Wen Hu
Research Scientist
CSIRO ICT Centre, Australia

Motivations

- Is security an issue in WSNs?
  - Over-the-air-programming (bootloader)
  - 6LowPan, TinyWebService, Unix-like OS...

- Security applications, health monitoring applications, metering applications.

WSN security challenges

- Very limited resources
  - 8-bit/16-bit microcontrollers
  - Less than 10KB RAM
  - AA batteries
- Security algorithms are computational and memory intensives

Related work

- Symmetric key algorithms are computational efficient
  - AES, XTEA, TinySec
  - Provide message confidentiality
  - Poor support for message authenticity and integrity
  - Key distribution algorithm?
- Asymmetric key algorithms are resource intensive
  - TinyPK (RSA) and TinyECC

Message authenticity and integrity

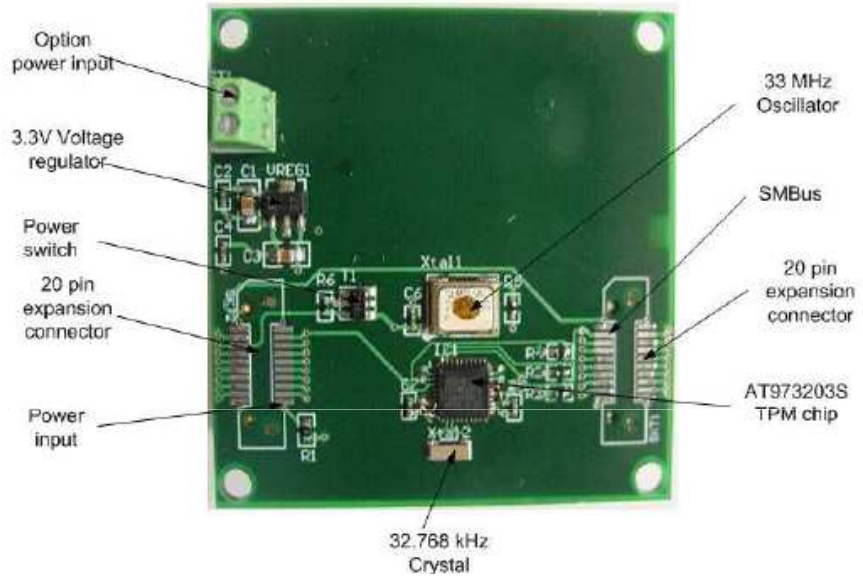
- PKCs such as RSA are widely used in the Internet to ensure message authenticity and integrity
  - Secure hashing algorithms such as SHA-1 to ensure message integrity
  - Digital signatures to ensure message authenticity

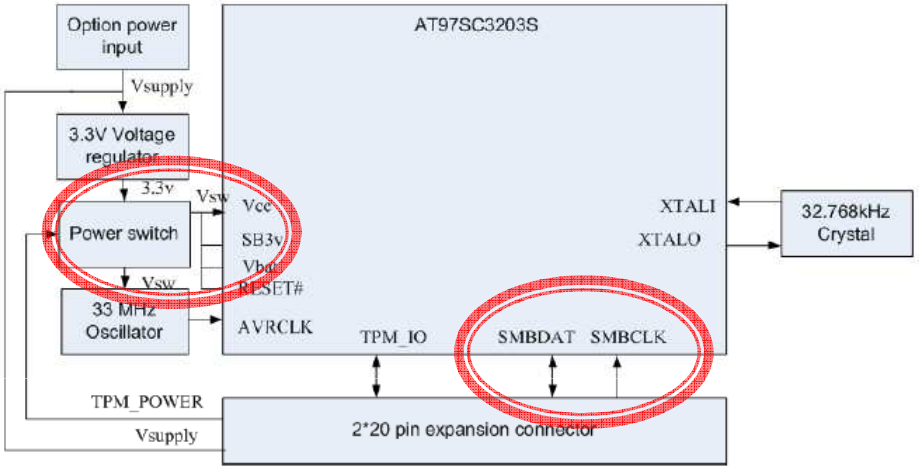
- RSA parameters
  - Public component (e)
  - Key size (s)

Trusted Platform Module (TPM)

- Cryptography operation engine
  - RNG, SHA-1, HMAC, RSA
- Platform Configuration Registers (PCRs)
  - Stores the signatures (by SHA-1 or HMAC)
  - Remote/local attestation
- Secure I/O
- Seal Storage

trustedFleck





TrustedFleck Primitives

```
/* trustedFleck public key collector. */
uint8_t fos_tpm_getPubKey(uint8_t *pubKey);

/* Asymmetric key encryption/decryption. */
uint8_t fos_tpm_encryption(uint8_t *msg, uint16_t len,
                           uint8_t *pubKey, uint8_t *cipher);
uint8_t fos_tpm_decryption(uint8_t *cipher, uint8_t *msg,
                           uint16_t *len);

/* Digital signature and verification. */
uint8_t fos_tpm_sign(uint8_t *digest, uint8_t *signature);
uint8_t fos_tpm_verifySign(uint8_t *signature, uint8_t *pubKey,
                           uint16_t *digest);

/* Symmetric session key encryption/decryption. */
uint8_t fos_xtea_encipher(uint8_t *msg, uint8_t *key,
                          uint8_t *cipher, uint8_t nRounds);
uint8_t fos_xtea_decipher(uint8_t *cipher, uint8_t *key,
                          uint8_t *msg, uint8_t nRounds);

/* Symmetric session key retrieve and store*/
fos_xtea_getkey(uint8_t *key, uint8_t location);
fos_xtea_storekey(uint8_t *key, uint8_t location);
```

Evaluation (I)

Table I. Comparison of RSA encryption times.

Public Exponent (e)	Software 1024 bit	Software 2048 bit	Hardware 2048 bit
3	0.45s	65s	N/A
65,537	4.185s	450s	0.055s

Table II. RSA computation time in trustedFleck for $e = 65,537$ and 2048 bit key.

Encryption	Decryption	Sign	Verification
55ms	750ms	787ms	59ms

Evaluation (II)

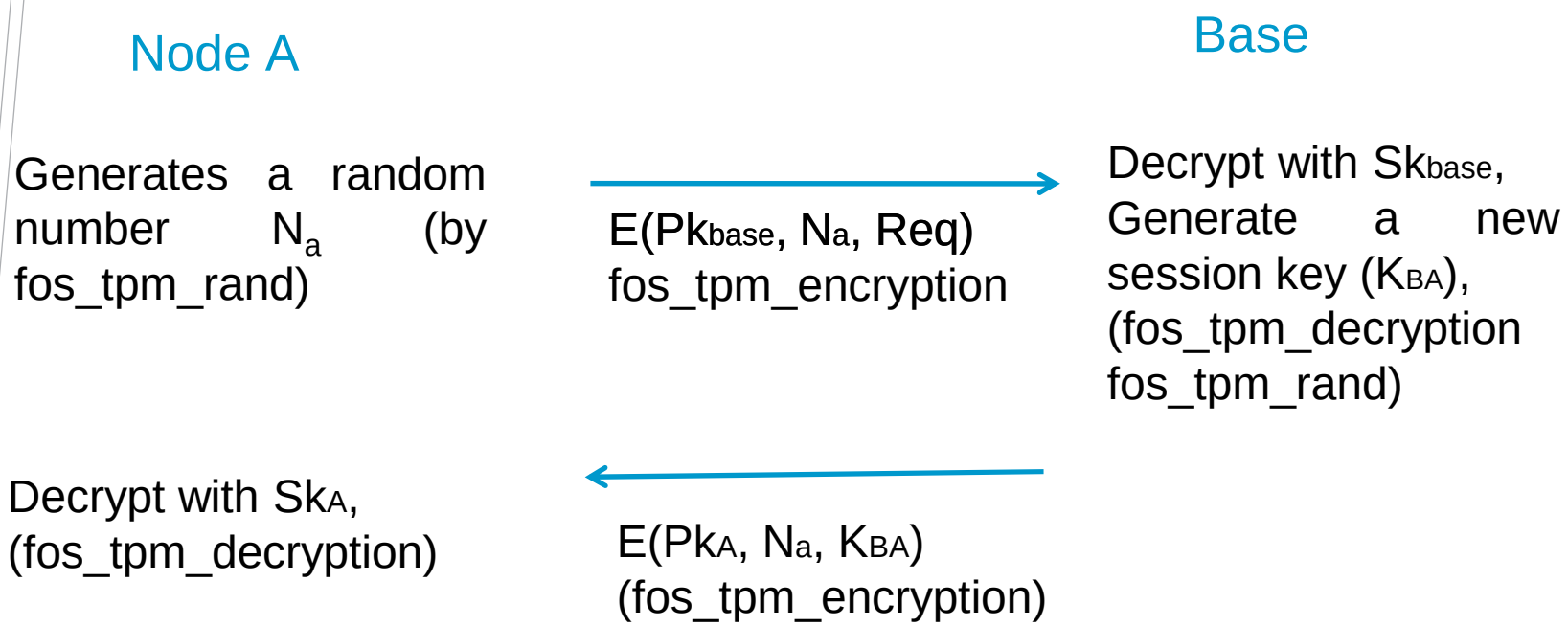
Table III. trustedFleck current consumption

Module	Current (mA)
Fleck3 (without radio, node idle)	8.0
Fleck3 + Receive	18.4
Fleck3 + Transmit	36.8
Fleck3 + TPM encryption	50.4
Fleck3 + TPM decryption	60.8
Fleck3 + TPM signature	60.8
Fleck3 + TPM signature verification	50.4

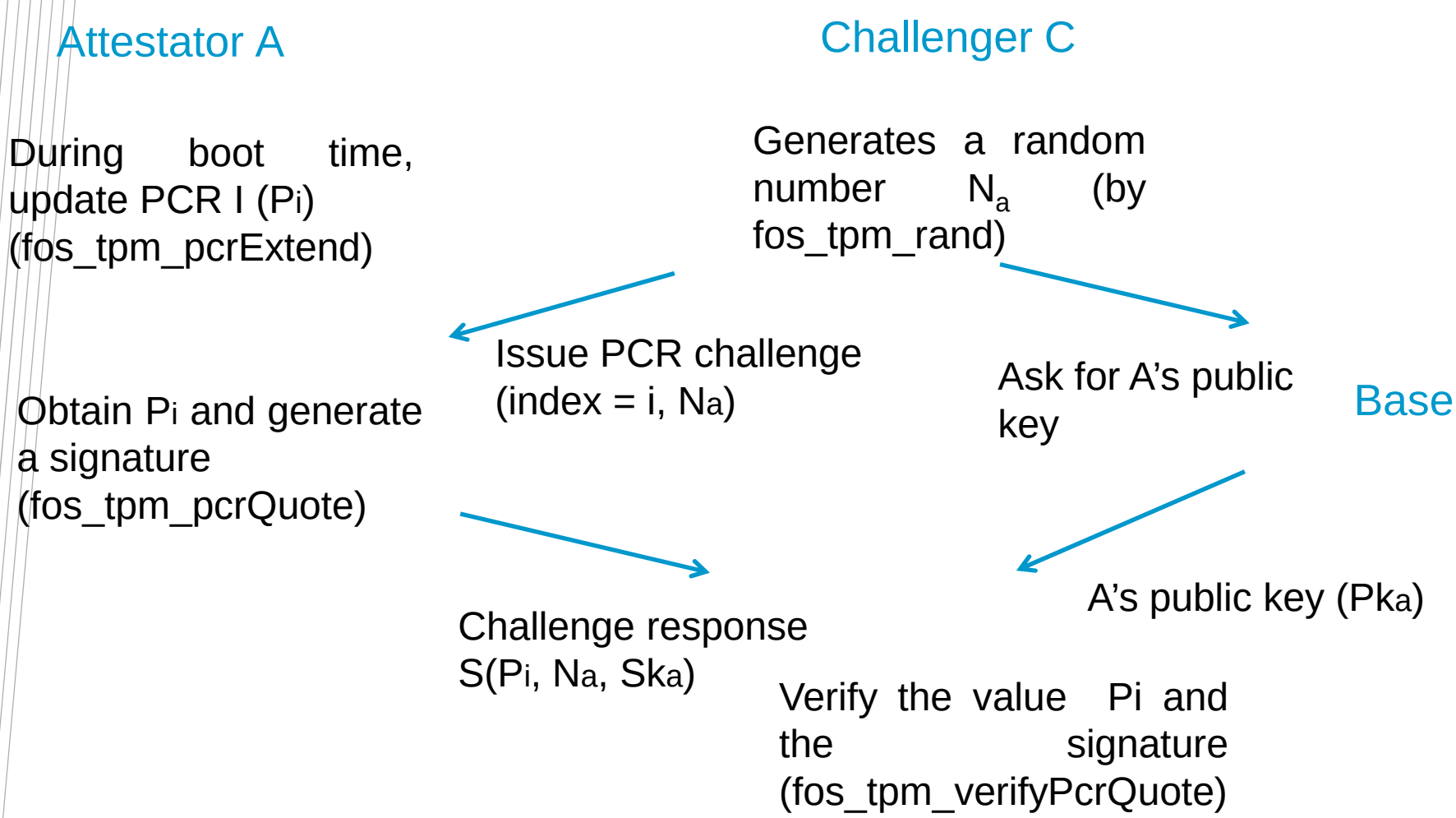
Table IV. trustedFleck (RSA and XTEA) encryption energy consumption for *one bit* of data.

Platform	Current (mA)	Time (μs)	Energy (μJ)
RSA (software, $e = 65,537$, 2048 bit key)	8.0	219,730	7,030.0
RSA (hardware, $e = 65,537$, 2048 bit key)	50.4	27	5.4
XTEA (software, 128 bit key)	8.0	18	0.6

Examples --- Symmetric Key Request



Examples --- Remote Attestation



Discussions

- Secure software update
- Backward and forward secrecy
- Secure Remote Procedure Call

Conclusions

- Strong (2048-bit) asymmetric key security for message authenticity and integrity
- Affordable (financially, form factor, and energy consumption)
- Remote attestations (trusted sensor networks)
- Easy to use

Q & A

Thank you !

Trusted Primitives

```
1  /*TPM PCR functions*/
2  uint8_t fos_tpm_pcrExtend(uint8_t pcrIndex, uint8_t *inputSha1,
3                            uint8_t *extendSha1);
4
5  uint8_t fos_tpm_pcrRead(uint8_t pcrIndex, uint8_t *outputSha1);
6
7  uint8_t fos_tpm_pcrQuote(uint8_t key_index, uint8_t *signature,
8                           uint8_t *digest);
9
10 uint8_t fos_tpm_verifyPcrQuote(uint8_t* signature, uint8_t *pubKey,
11                           uint8_t *digest);
```

Evaluations (II)

Table V. trustedFleck PCR operation computational time, current and energy consumption

Module	Computational Time (ms)	Current (mA)	Energy (mJ)
Fleck3 + TPM PCR read	5.8	52.8	0.918
Fleck3 + TPM PCR quote	1,400	64	268.8
Fleck3 + TPM PCR verify quote	900	52	140.4
Fleck3 + TPM PCR extend	see Fig. 8	51.2	see Fig. 8