A Performance Study of Crypto-Hardware in the Low-end IoT

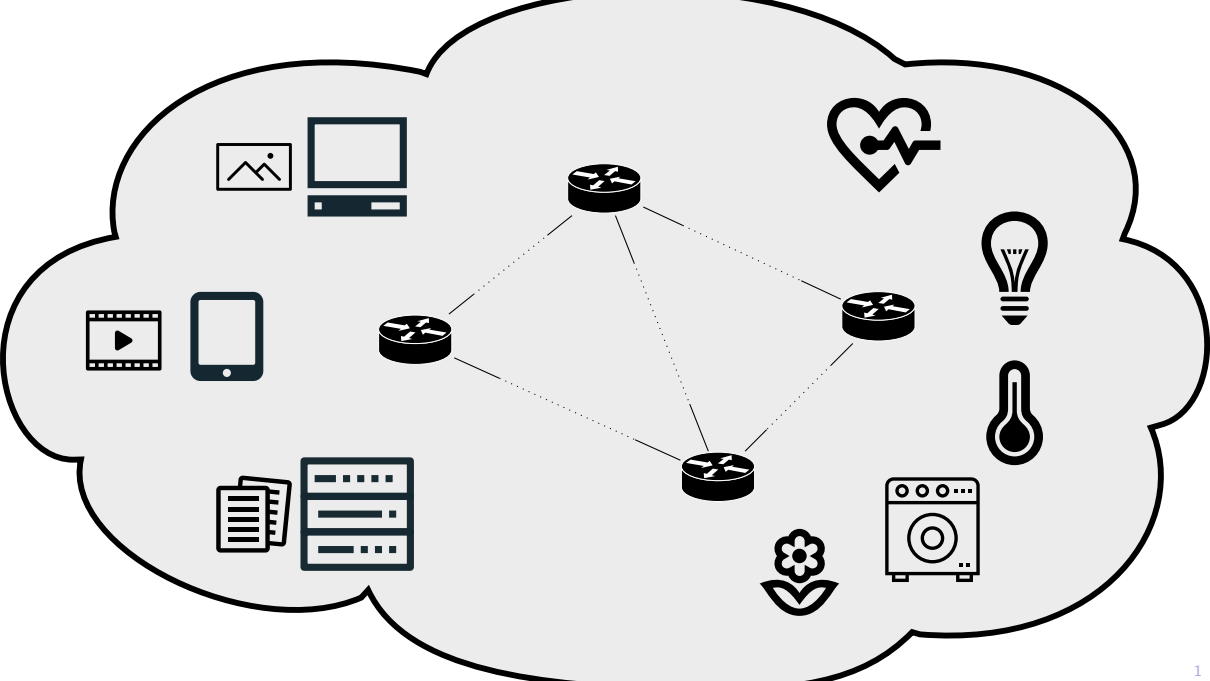
Peter Kietzmann[1], Lena Boeckmann[1], Leandro Lanzieri[1]
Thomas C. Schmidt[1] and Matthias Wählisch[2]

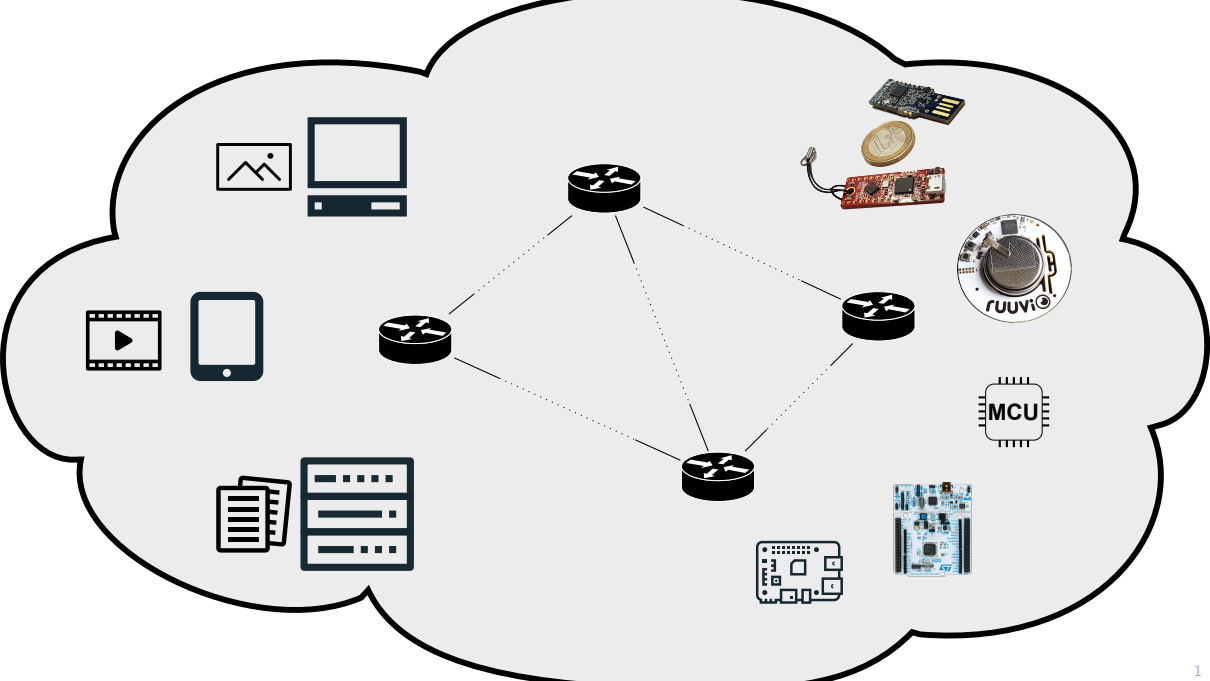
peter.kietzmann@haw-hamburg.de

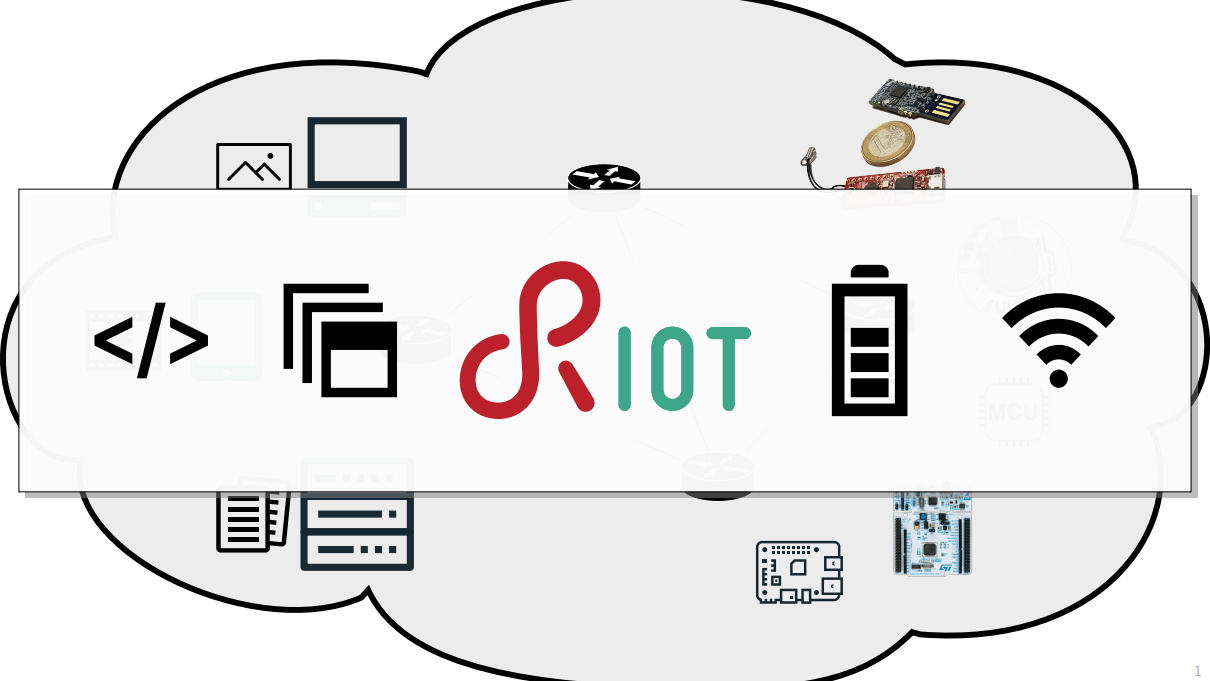
18th International Conference on Embedded Wireless Systems and Networks (EWSN 2021)

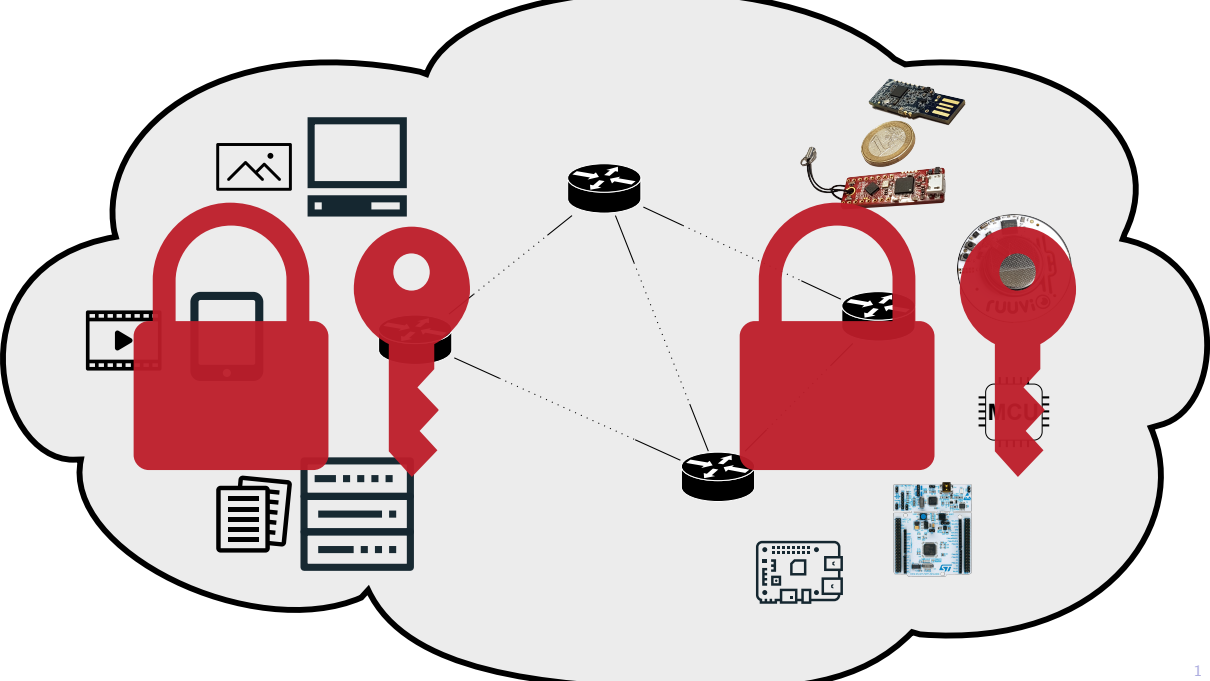
HAW HAMBURG

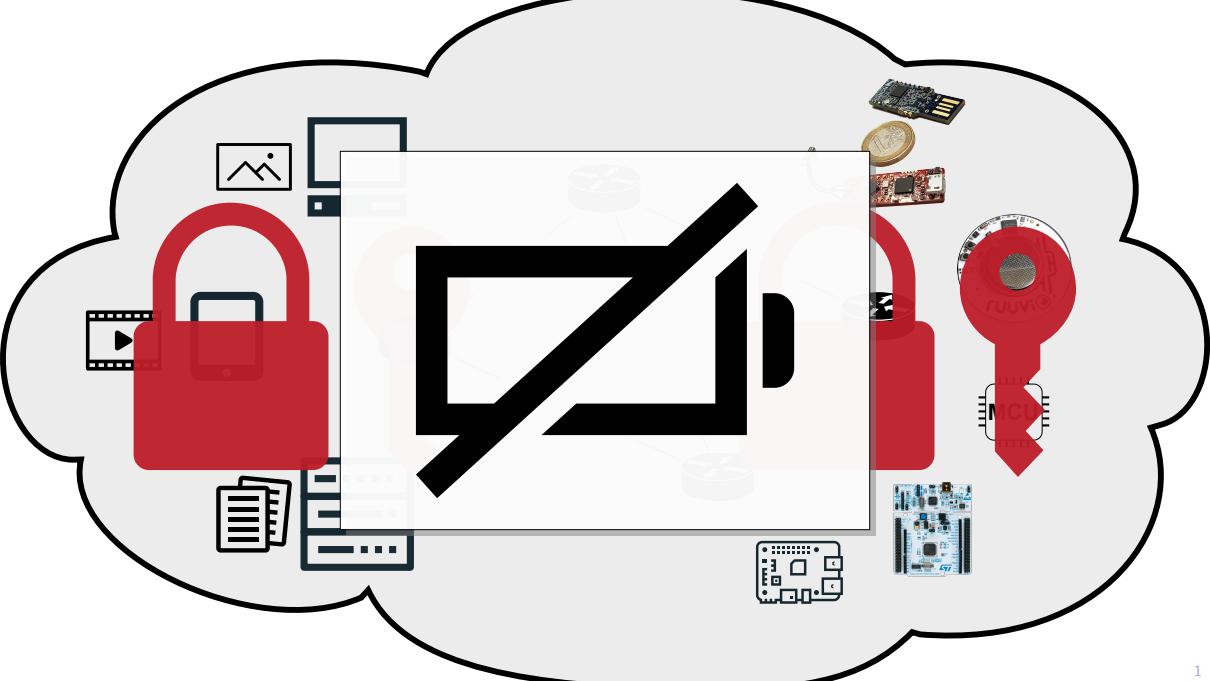
Freie Universität Berlin

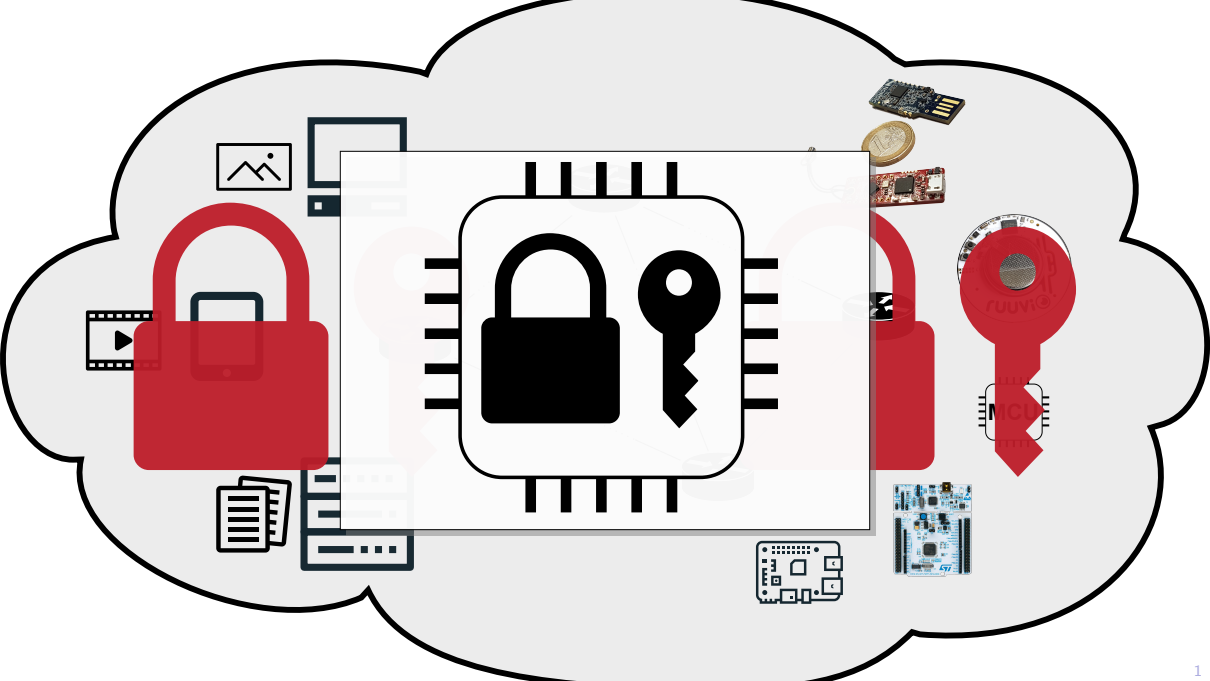



MCU


RIOT


MCU


MCU


MCU

Related Work

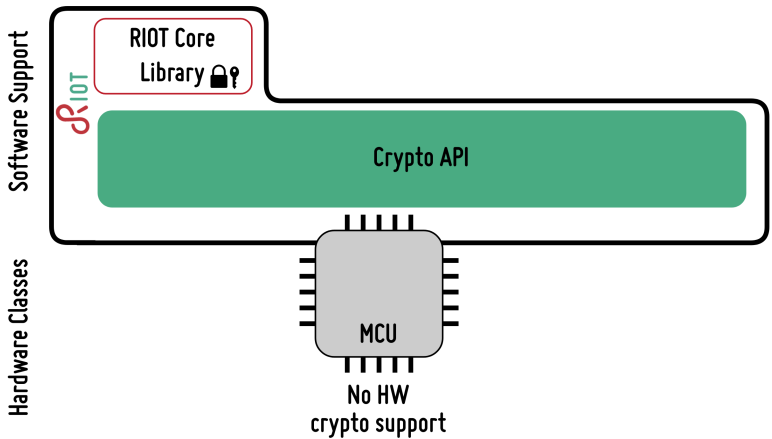
Categories	Paper References
Crypto support in operating systems	
Lack of seamless crypto-integration	[2, 3, 5, 6, 20, 46]
Missing hardware support	
Performance of crypto-software	
Specific to algorithm	[1, 9, 13, 14, 16, 19, 26, 29, 32, 36, 40, 44, 48]
Often bare-metal	
Performance of crypto-hardware	
Consistent integration missing	[11, 15, 17, 27, 30, 33, 34, 37, 39, 45, 47]
Only partial comparisons	

Related Work

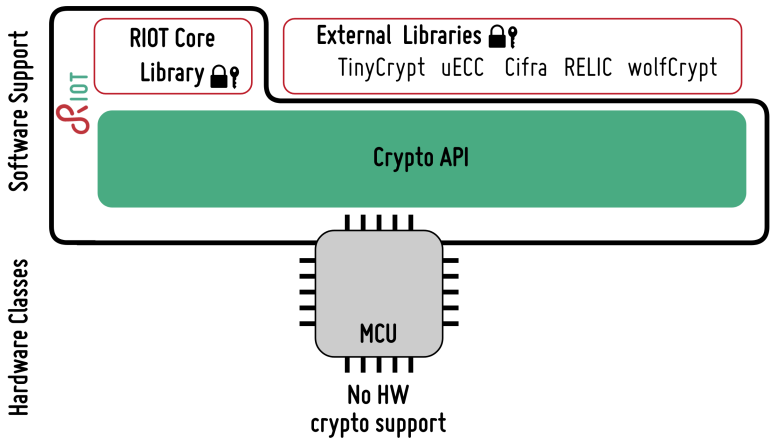
Categories	Paper References
Crypto support in operating systems	
Lack of seamless crypto-integration	[2, 3, 5, 6, 20, 46]
Missing hardware support	
Performance of crypto-software	
Specific to algorithm	[1, 9, 13, 14, 16, 19, 26,
Often bare-metal	29, 32, 36, 40, 44, 48]
Performance of crypto-hardware	
Consistent integration missing	[11, 15, 17, 27, 30, 33,
Only partial comparisons	34, 37, 39, 45, 47]

We present a software/hardware analysis across off-the-shelf devices using novel crypto-subsystem

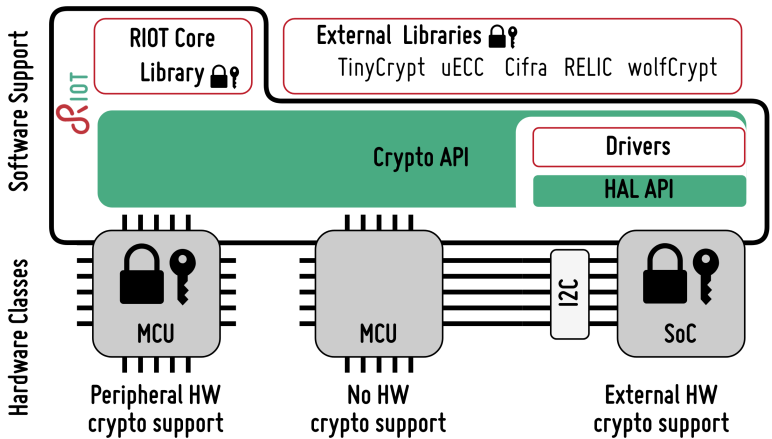
A Crypto-Support Layer for RIOT



A Crypto-Support Layer for RIOT



A Crypto-Support Layer for RIOT



Outline

Designing the RIOT Crypto-Subsystem

Basic Crypto-Hardware Acceleration

ECC Hardware Acceleration

Conclusion & Outlook

Designing the RIOT Crypto-Subsystem

Driver Integration

Vendor Code

- Advantage of specific vendor knowledge and testing
- Sustainable maintenance and updates in OS

Third-Party Package

- RIOT package clones, compiles, and links repositories
- Software wrappers connect hardware specific code to OS

Mutex Allocation

- Protect device against concurrent access in every function
- Dual-accelerators require management to select free device

Low-Power Management

- Turn on/off fast device for every operation
- Turn on/off slow device based on user counter

Driver Integration

Vendor Code

- Advantage of specific vendor knowledge and testing
- Sustainable maintenance and updates in OS

Third-Party Package

- RIOT `package` clones, compiles, and links repositories
- Software wrappers connect hardware specific code to OS

Mutex Allocation

- Protect device against concurrent access in every function
- Dual-accelerators require management to select free device

Low-Power Management

- Turn on/off fast device for every operation
- Turn on/off slow device based on user counter

Driver Integration

Vendor Code

- Advantage of specific vendor knowledge and testing
- Sustainable maintenance and updates in OS

Third-Party Package

- RIOT package clones, compiles, and links repositories
- Software wrappers connect hardware specific code to OS

Mutex Allocation

- Protect device against concurrent access in every function
- Dual-accelerators require management to select free device

Low-Power Management

- Turn on/off fast device for every operation
- Turn on/off slow device based on user counter

Driver Integration

Vendor Code

- Advantage of specific vendor knowledge and testing
- Sustainable maintenance and updates in OS

Third-Party Package

- RIOT package clones, compiles, and links repositories
- Software wrappers connect hardware specific code to OS

Mutex Allocation

- Protect device against concurrent access in every function
- Dual-accelerators require management to select free device

Low-Power Management

- Turn on/off fast device for every operation
- Turn on/off slow device based on user counter

Context Handling

Generic definition

- Crypto functions operate on internal state
- Hardware specific, defined by vendor driver
- Common `struct` to face OS

```
typedef struct {
  CRYS_HASHUserContext_t ctx;
} sha_ctx_t;
```

Individual allocation

- Applications require individual state
- External allocation and state maintenance

```
void sha_init(sha_ctx_t *c);

int main(void)
{
  /*Allocate generic struct*/
  sha_ctx_t my_ctx;
  sha_init(&my_ctx);

  return 0;
}
```

Caveat: Few devices require read-save-write

Context Handling

Generic definition

- Crypto functions operate on internal state
- Hardware specific, defined by vendor driver
- Common `struct` to face OS

```
typedef struct {
  CRYS_HASHUserContext_t ctx;
} sha_ctx_t;
```

Individual allocation

- Applications require individual state
- External allocation and state maintenance

Caveat: Few devices require read-save-write

```
void sha_init(sha_ctx_t *c);

int main(void)
{
  /*Allocate generic struct*/
  sha_ctx_t my_ctx;
  sha_init(&my_ctx);

  return 0;
}
```

System Modules

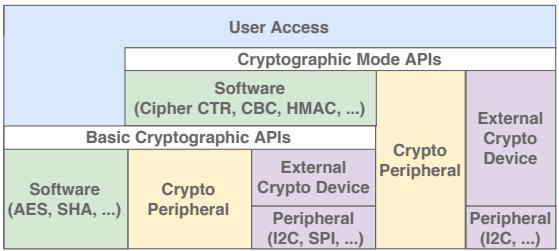
Stacked Design

Two hardware-agnostic user APIs

1. *Cryptographic Mode API* (*e.g.*, AES CBC)
2. *Basic Cryptographic API* (*e.g.*, AES block)

Three levels of acceleration

1. Full hardware acceleration
2. Partial hardware acceleration
3. No hardware acceleration



System Modules

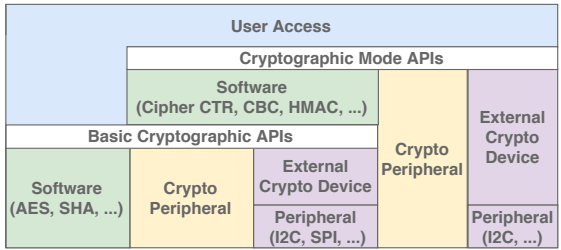
Stacked Design

Two hardware-agnostic user APIs

1. *Cryptographic Mode API* (*e.g.*, AES CBC)
2. *Basic Cryptographic API* (*e.g.*, AES block)

Three levels of acceleration

1. Full hardware acceleration
2. Partial hardware acceleration
3. No hardware acceleration



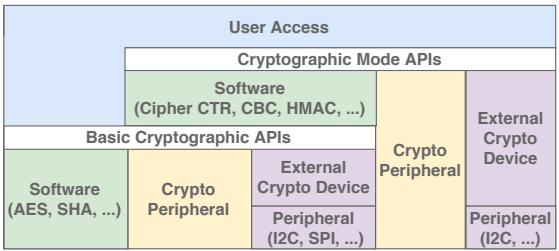
System Modules

Configuration with `Kconfig`

- Each block in crypto-stack is a `Kconfig` symbol
- Hardware capabilities indicated by symbols
- Crypto APIs implemented as `choice` symbols

⇒**Transparent replacement of backends**

- `Kconfig` resolves build dependencies
- Hardware-acceleration always prioritized

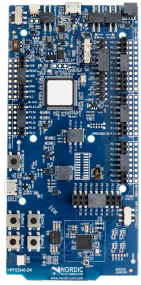


System Modules

Manual Reconfiguration with *menuconfig*

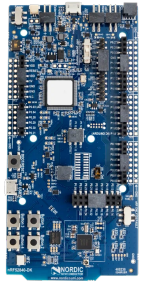
Basic Crypto-Hardware Acceleration

Platform Overview



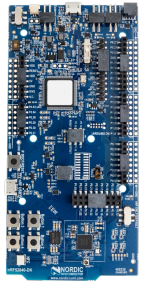
Device / Feature	**nRF52840** @64 MHz
TRNG	✓
SHA-256	✓
HMAC-SHA256	✓
AES-128	**ECB**, CTR, **CBC**, ...
ECC	**secp256**k/**r1**, ...
ECDSA / ECDH	✓

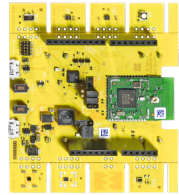
Platform Overview



Feature \ Device	**nRF52840** @64 MHz	**EFM32**(PG12) @40 MHz
TRNG	✓	✓
SHA-256	✓	✓
HMAC-SHA256	✓	✘
AES-128	**ECB**, CTR, **CBC**, ...	**ECB**, CTR, **CBC**, ...
ECC	**secp256**k/**r1**, ...	**secp256r1**, ...
ECDSA / ECDH	✓	✘

Platform Overview

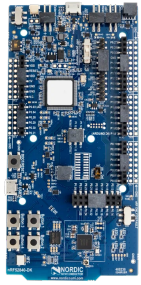


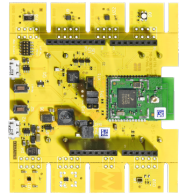


Feature \ Device	**nRF52840** @64 MHz	**EFM32**(PG12) @40 MHz	**MKW22D** @48 MHz
TRNG	✓	✓	✓
SHA-256	✓	✓	✓
HMAC-SHA256	✓	✘	✘
AES-128	**ECB**, CTR, **CBC**, ...	**ECB**, CTR, **CBC**, ...	✓
ECC	**secp256**k/**r1**, ...	**secp256r1**, ...	✘
ECDSA / ECDH	✓	✘	✘

Platform Overview

I2C



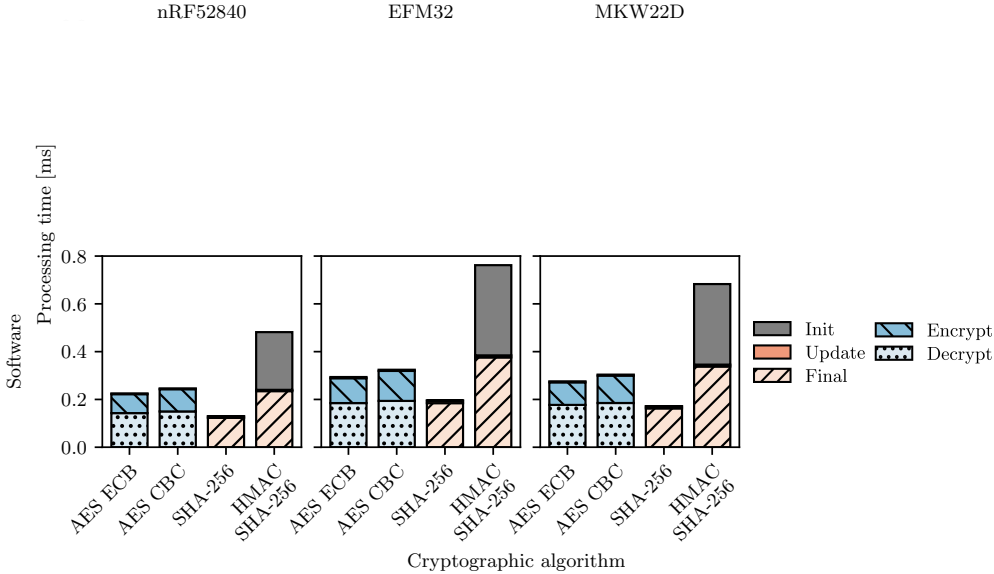


Feature \ Device	**ATECC608A** I2C@400kbps	**nRF52840** @64 MHz	**EFM32**(PG12) @40 MHz	**MKW22D** @48 MHz
TRNG	✓	✓	✓	✓
SHA-256	✓	✓	✓	✓
HMAC-SHA256	✓	✓	✘	✘
AES-128	**ECB**, GCM	**ECB**, CTR, **CBC**, ...	**ECB**, CTR, **CBC**, ...	✓
ECC	**secp256r1** (**P-256**)	**secp256k**/**r1**, ...	**secp256r1**, ...	✘
ECDSA / ECDH	✓	✓	✘	✘

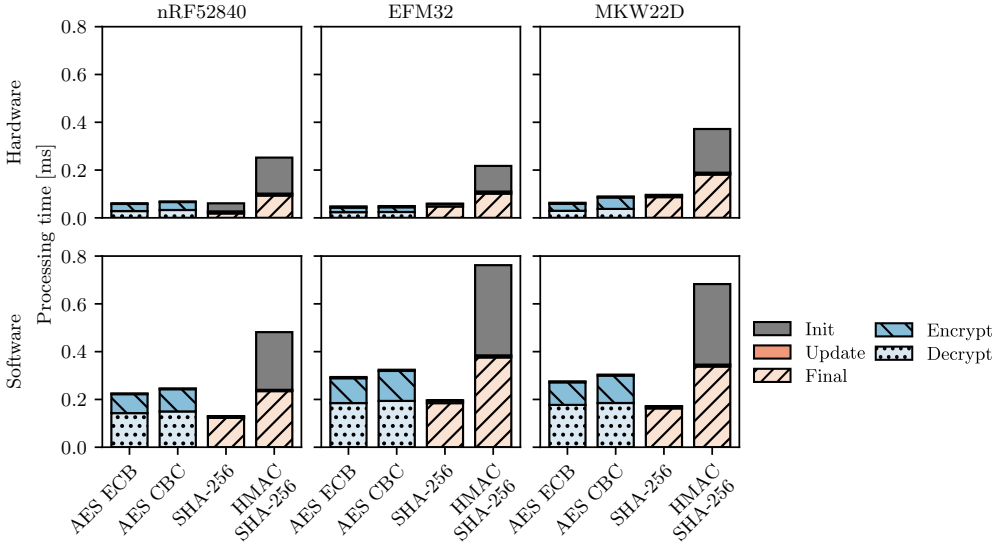
Platform Overview

Featur	ATECC60 I2C@400kB	nRF52840 64 MHz	EF / PG MHz	MK 2D @4 Hz
TRNG	✓	✓		
SHA-2 6	✓	✓		
HMAC SHA256	✓	✓		
AES-1 8		ECB, C BC,	ECB, C	
ECC	secp256r1 (P-256)	secp256k/r1, ...	secp256r1, ...	✘
ECDS / ECDH	✓	✓	✘	✘

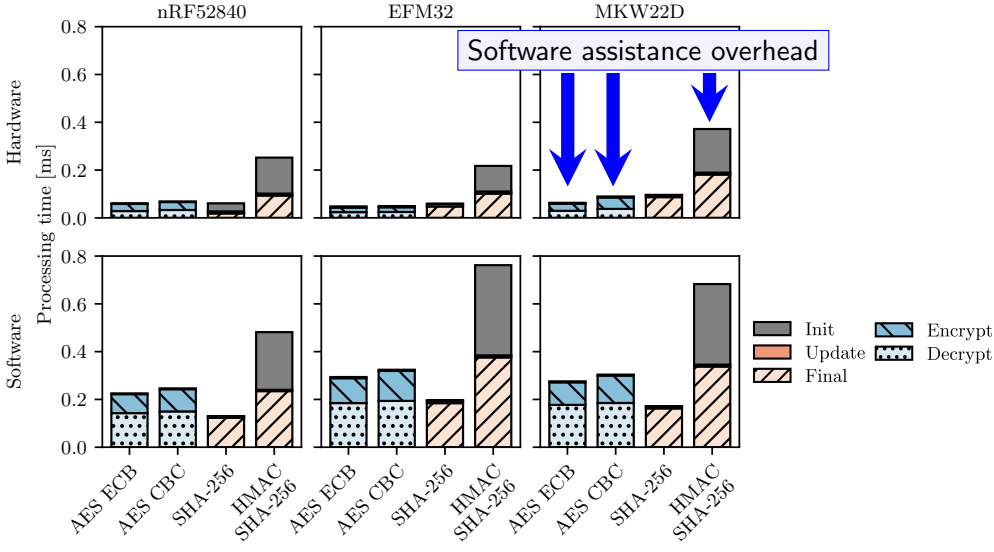
Basic Crypto - Processing Time for **Short** Inputs (32 Byte)



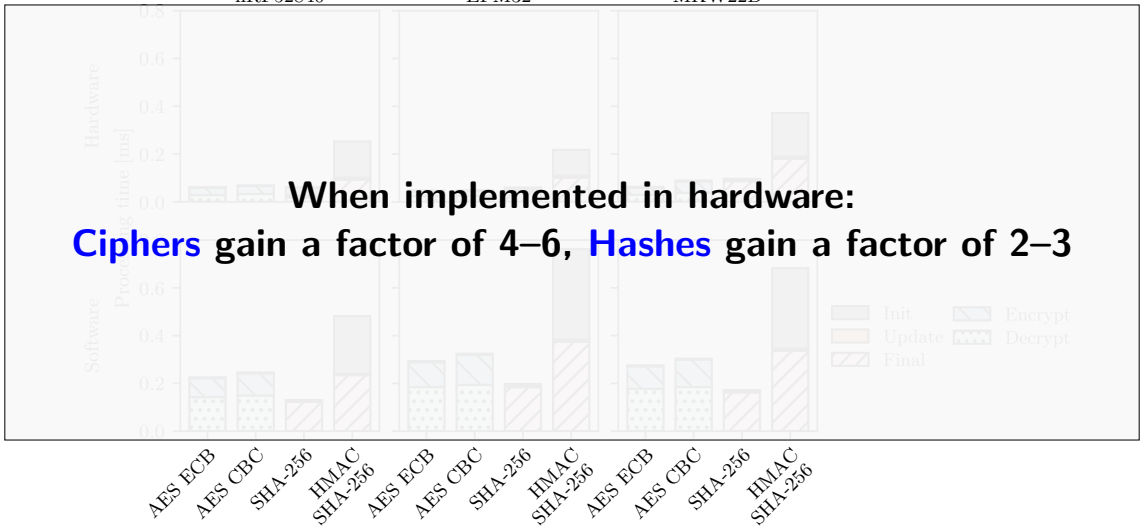
Basic Crypto - Processing Time for **Short** Inputs (32 Byte)



Basic Crypto - Processing Time for **Short** Inputs (32 Byte)

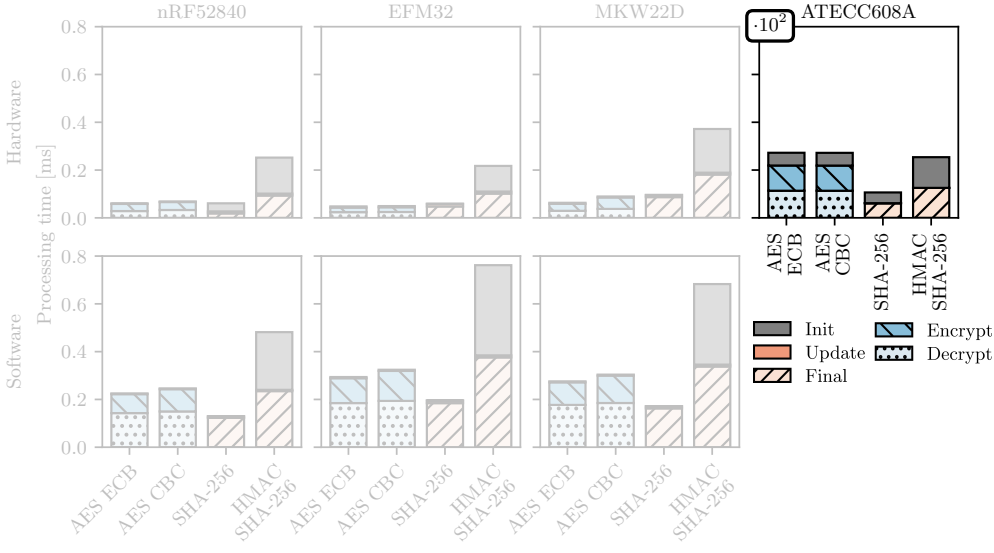


Basic Crypto - Processing Time for **Short** Inputs (32 Byte)

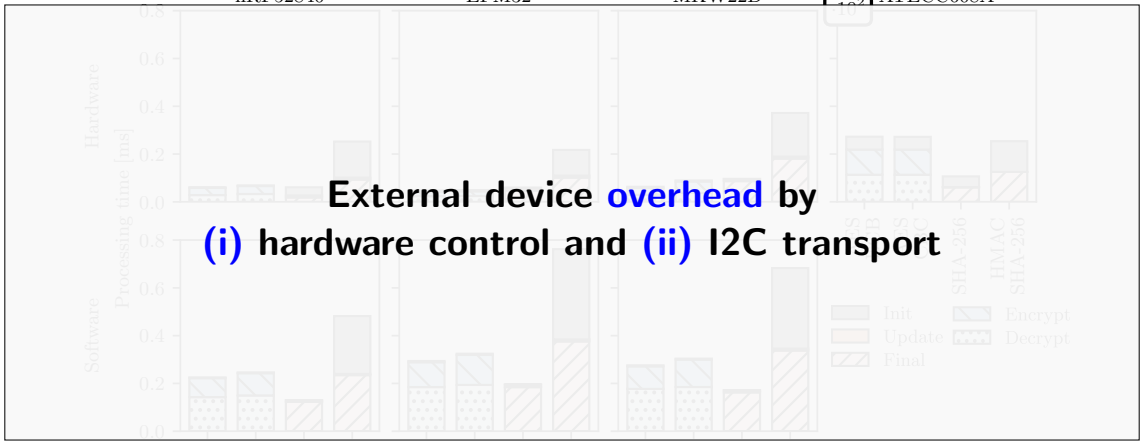


When implemented in hardware:
Ciphers gain a factor of 4–6, Hashes gain a factor of 2–3

Basic Crypto - Processing Time for **Short** Inputs (32 Byte)

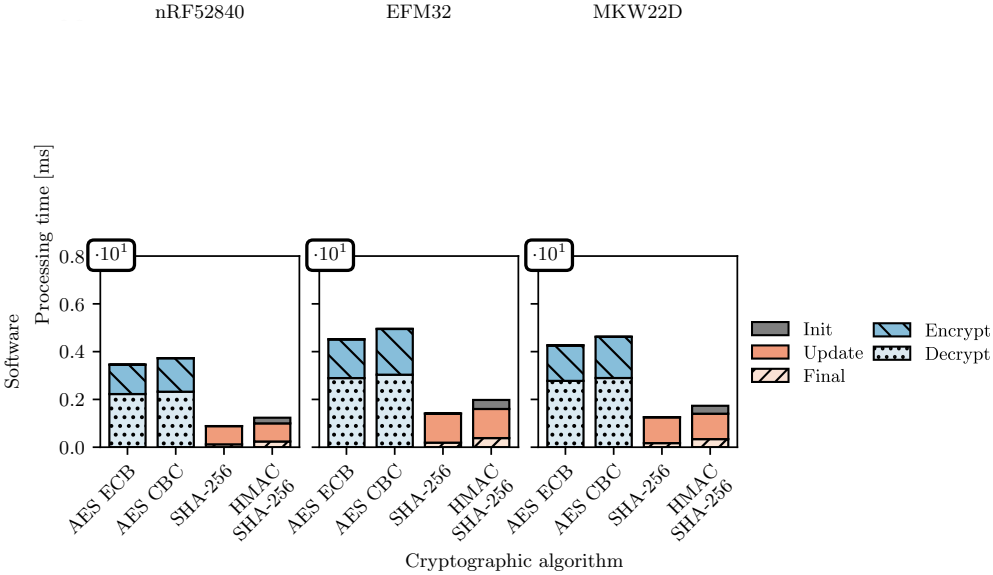


Basic Crypto - Processing Time for **Short** Inputs (32 Byte)

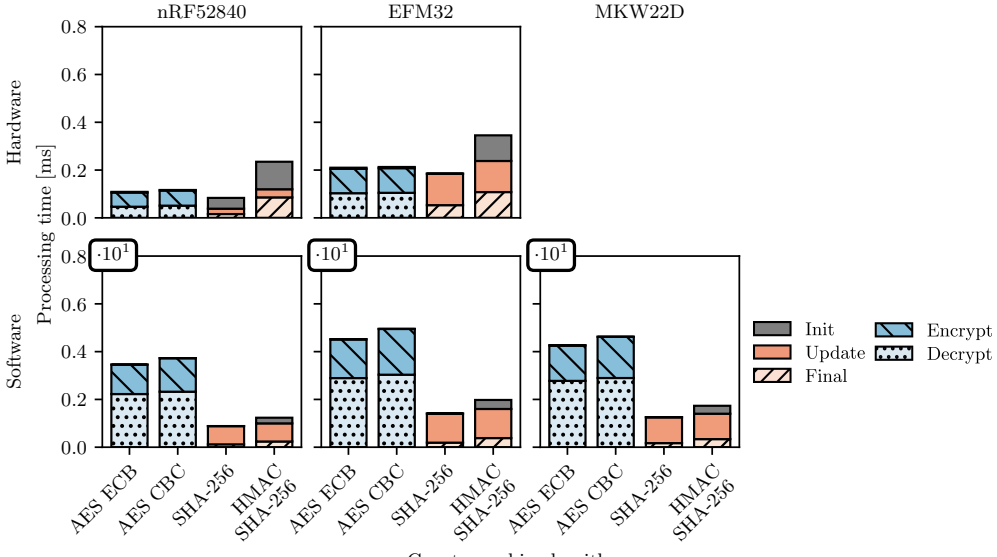


External device overhead by
(i) hardware control and (ii) I2C transport

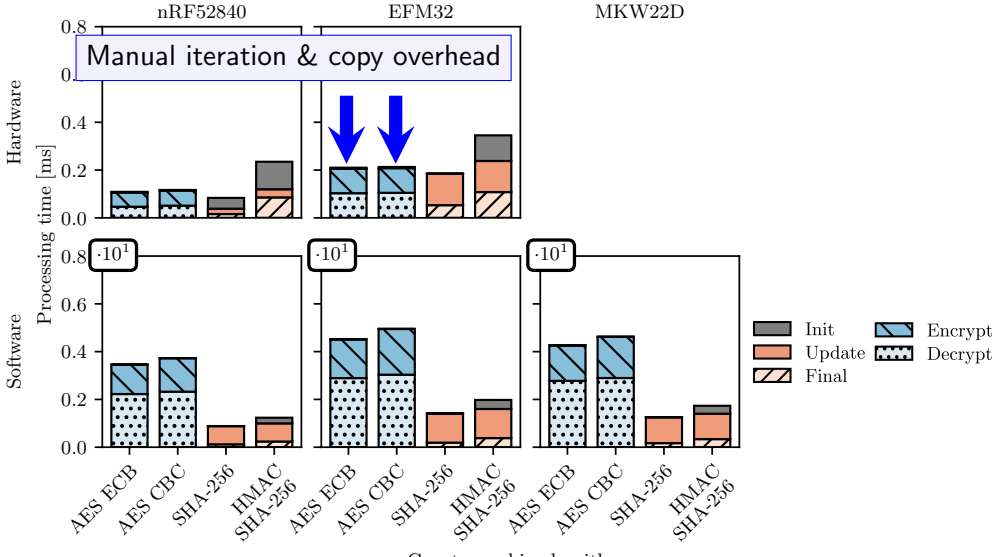
Basic Crypto - Processing Time for **Long** Inputs (512 Byte)



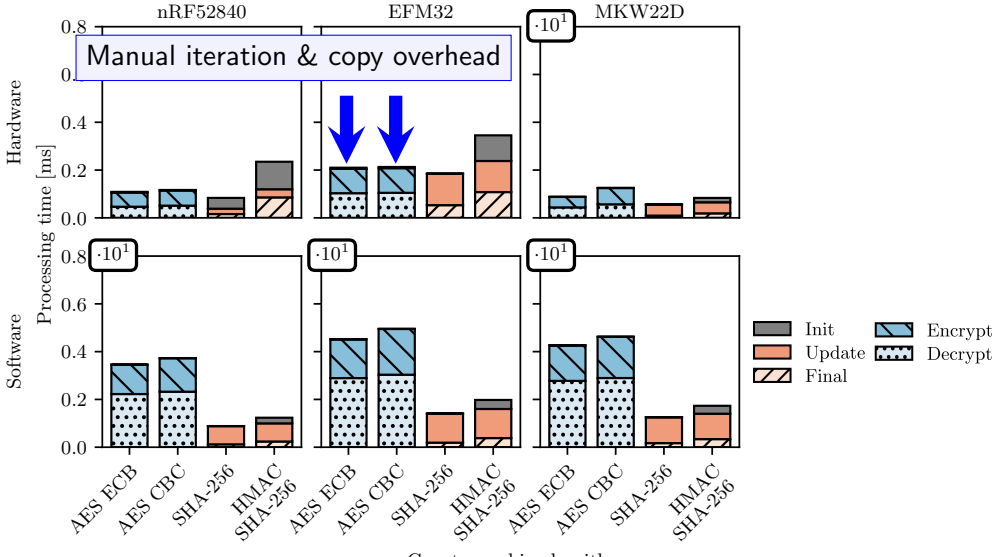
Basic Crypto - Processing Time for **Long** Inputs (512 Byte)



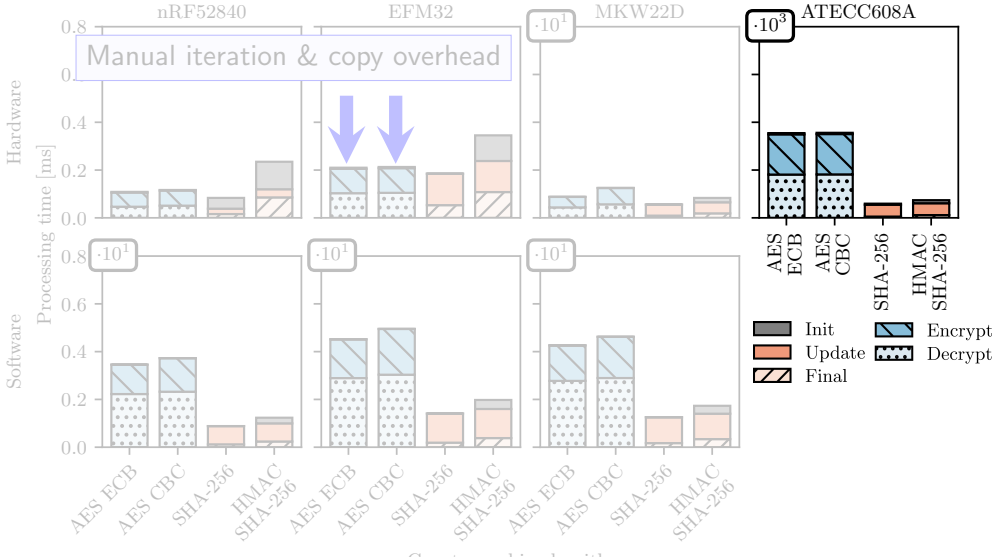
Basic Crypto - Processing Time for **Long** Inputs (512 Byte)



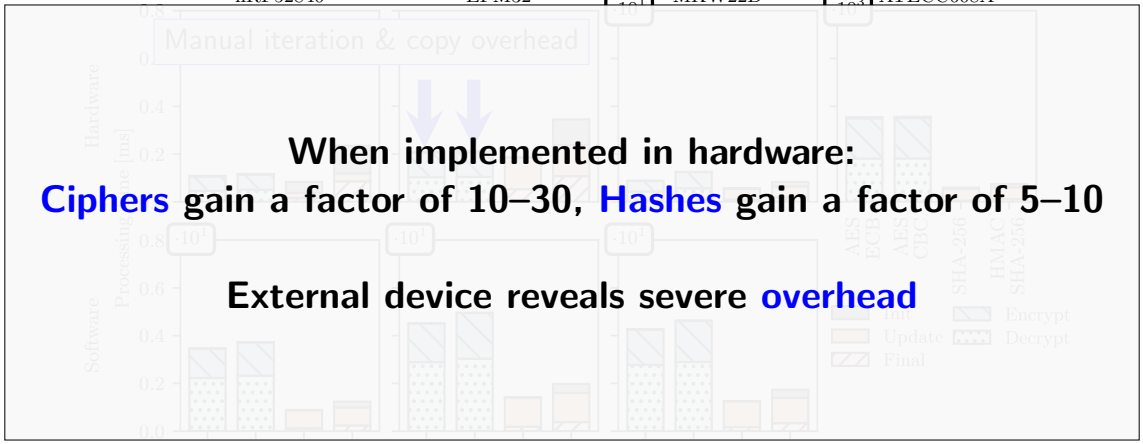
Basic Crypto - Processing Time for **Long** Inputs (512 Byte)



Basic Crypto - Processing Time for **Long** Inputs (512 Byte)



Basic Crypto - Processing Time for **Long** Inputs (512 Byte)

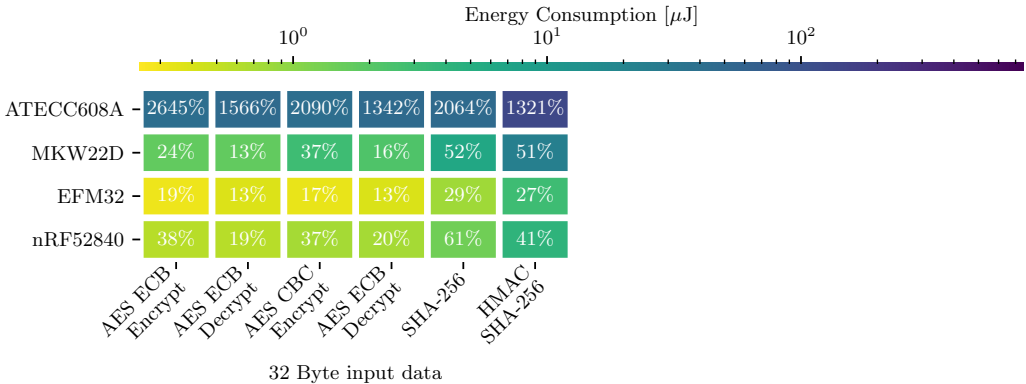


When implemented in hardware:
Ciphers gain a factor of 10–30, Hashes gain a factor of 5–10

External device reveals severe overhead

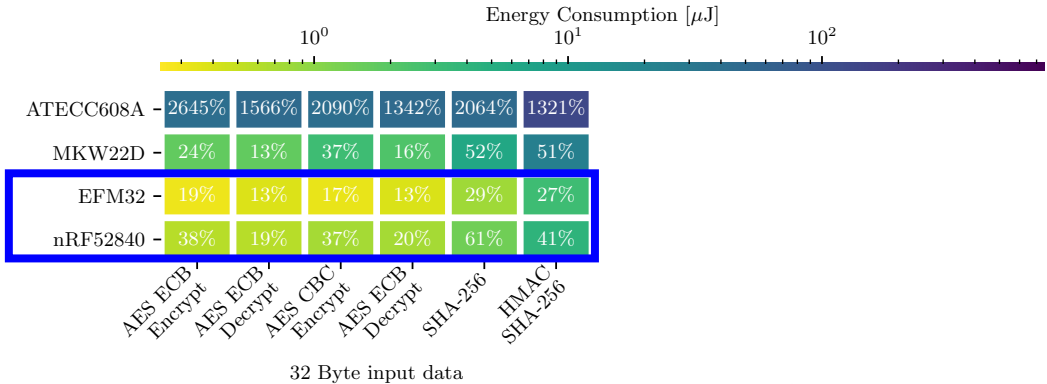
Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)



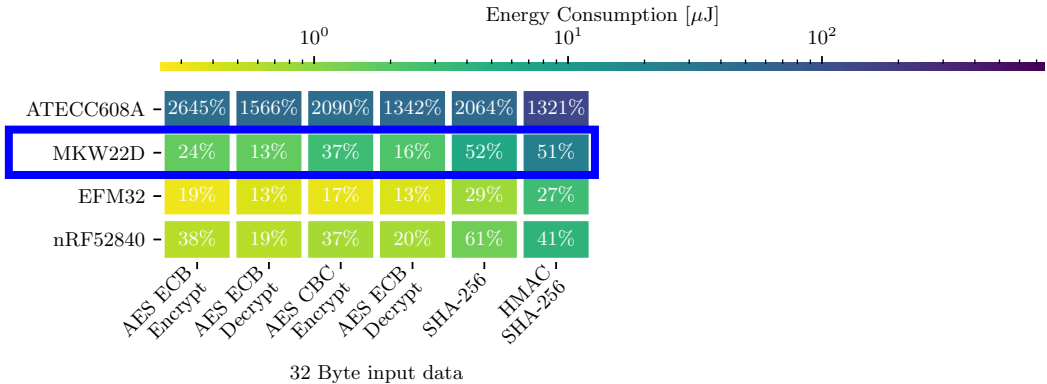
Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)



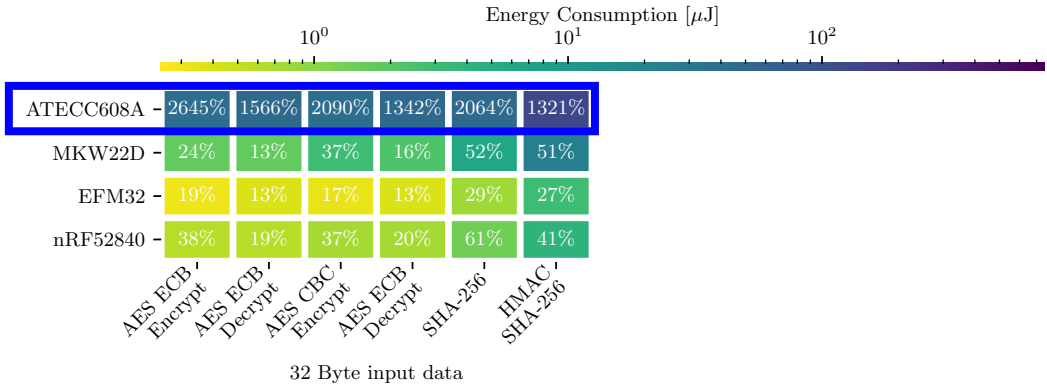
Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)



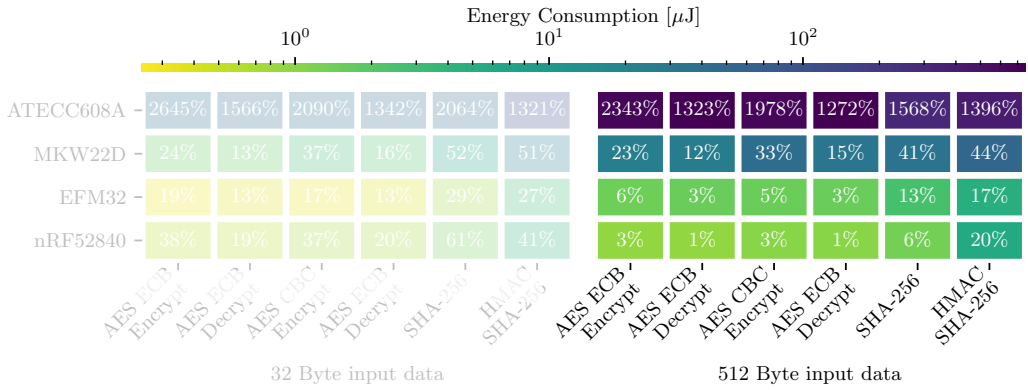
Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)



Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)



Basic Crypto - Energy Consumption

Color: Absolute energy consumption (hardware) **%:** Relative consumption (compared to software)

Performance gain increases with long inputs

Software-assistance equally profits for short/long inputs

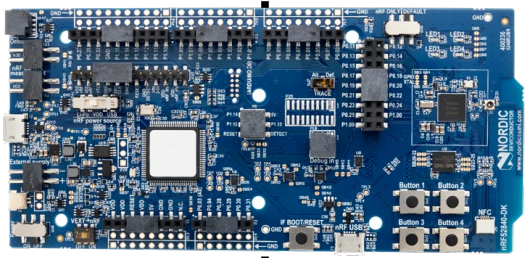
Energy demands for ext. device increase by 13–25

32 Byte input data

512 Byte input data

ECC Hardware Acceleration

ECC - Platform Overview



Software

Relic toolkit

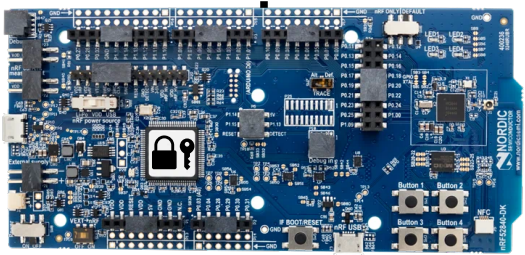
uECC Library

ECC - Platform Overview

Hardware

Software

I2C



uECC Library

ECC - Measurement Setup

ECDSA / ECDH on NIST P-256

Init

- Software/hardware bootstrap

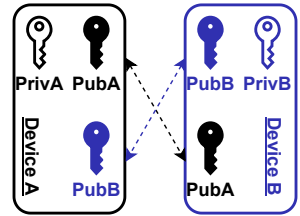
Keygen

- Generation of 256 Bit public/private key pair $\rightarrow$ *Pub*, *Priv*

ECC - Measurement Setup

ECDSA / ECDH on NIST P-256

Prerequisite:



Init

- Software/hardware bootstrap

Keygen

- Generation of 256 Bit public/private key pair $\rightarrow Pub, Priv$

Sign (ECDSA)

- Signature on 256 Bit message digest $\rightarrow S_A(PrivA)$

Verify (ECDSA)

- Verification of 256 Bit signature $\rightarrow V_B(S_A, PubA)$

Secret (ECDH)

- Generation of 256 Bit shared secret $\rightarrow \begin{matrix} Sec_A(PrivA, PubB) \\ Sec_B(PrivB, PubA) = Sec_A \end{matrix}$

ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

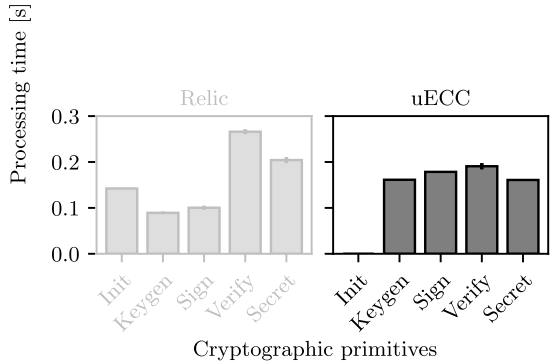
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

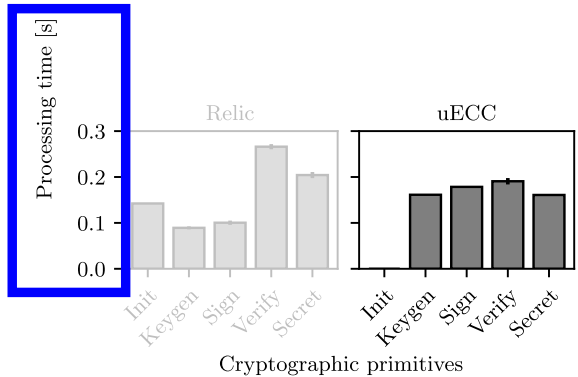
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

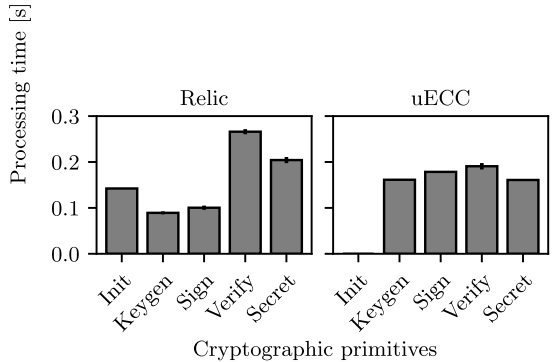
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

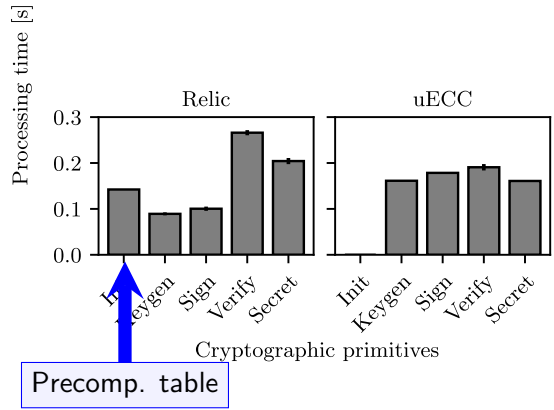
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

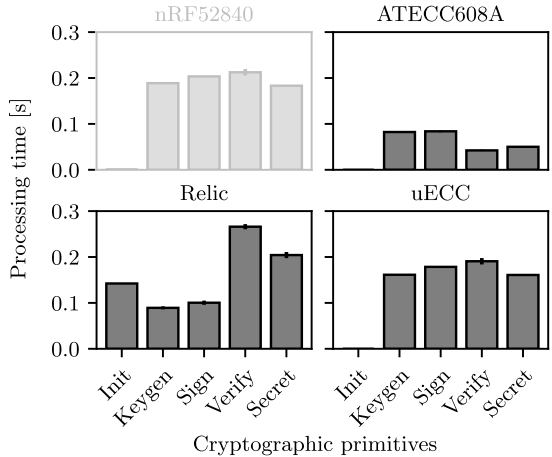
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

- Minimal, optimized library
- Static lookup tables

Relic

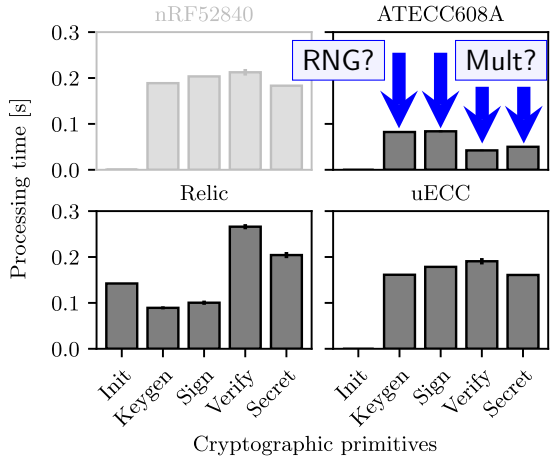
- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

nRF52840

- Hardware support for > 15 elliptic curves



ECC - Processing Time

ECDSA / ECDH on NIST P-256

uECC

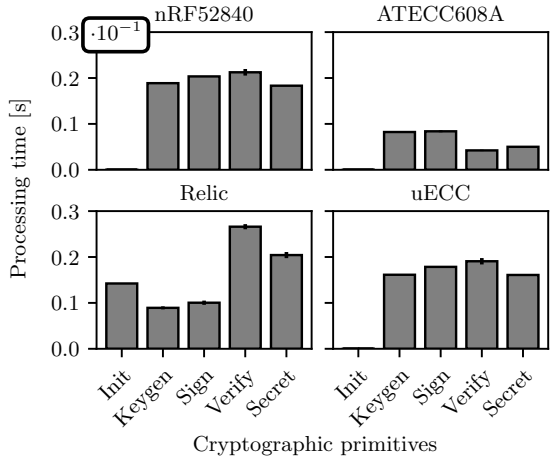
- Minimal, optimized library
- Static lookup tables

Relic

- Feature-rich crypto-toolkit
- Flexible and highly configurable

ATECC608A

- Constrained to single curve
- Secure memory for private keys

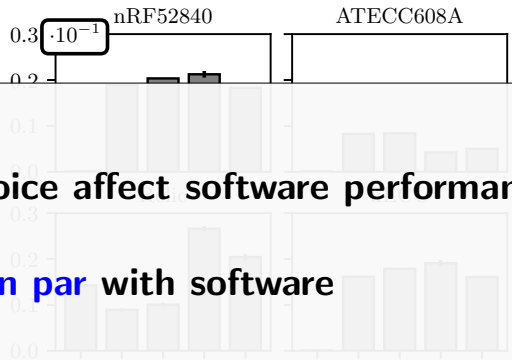


nRF52840

- Hardware support for > 15 elliptic curves

ECC - Processing Time

ECDSA / ECDH on NIST P-256



Configurability & algorithmic choice affect software performance

External device is on par with software

Peripheral accelerator gains by one order of magnitude

Conclusion & Outlook

Conclusion & Outlook

In this work, we contributed ...

- ... seamless hardware-crypto support for RIOT
- ... comparisons of crypto algorithms in hardware & software
- ... comprehensive system benchmarks on common IoT devices

Our results show that ...

- ... crypto-peripherals outperform software in runtime and energy
- ... ctx & stack usage are similar for hardware & software; drivers add memory overhead
- ... external devices are slow on symmetric crypto, performance advances on asym. crypto
- ... several vendor implementations reveal large optimization potential

In future work we will ...

- ... extend our analysis to trusted execution environments (*e.g.*, TrustZone)
- ... evaluate the impact on security protocols (*e.g.*, ACE-OAuth)

Conclusion & Outlook

In this work, we contributed ...

- ... seamless hardware-crypto support for RIOT
- ... comparisons of crypto algorithms in hardware & software
- ... comprehensive system benchmarks on common IoT devices

Our results show that ...

- ... crypto-peripherals outperform software in runtime and energy
- ... ctx & stack usage are similar for hardware & software; drivers add memory overhead
- ... external devices are slow on symmetric crypto, performance advances on asym. crypto
- ... several vendor implementations reveal large optimization potential

In future work we will ...

- ... extend our analysis to trusted execution environments (*e.g.*, TrustZone)
- ... evaluate the impact on security protocols (*e.g.*, ACE-OAuth)

Conclusion & Outlook

In this work, we contributed ...

- ... seamless hardware-crypto support for RIOT
- ... comparisons of crypto algorithms in hardware & software
- ... comprehensive system benchmarks on common IoT devices

Our results show that ...

- ... crypto-peripherals outperform software in runtime and energy
- ... ctx & stack usage are similar for hardware & software; drivers add memory overhead
- ... external devices are slow on symmetric crypto, performance advances on asym. crypto
- ... several vendor implementations reveal large optimization potential

In future work we will ...

- ... extend our analysis to trusted execution environments (*e.g.*, TrustZone)
- ... evaluate the impact on security protocols (*e.g.*, ACE-OAuth)

Thank You!

We support reproducible research.

`https://github.com/inetrg/EWSN-2021`