Permutation-based cryptography for the Internet of Things

Gilles Van Assche[1]

Joint work with Guido Bertoni, Joan Daemen[1,2], Seth Hoffert, Michaël Peeters[1] and Ronny Van Keer[1]

[1]STMicroelectronics
[2]Radboud University

RIOT Summit 2017
Berlin, September 25-26, 2017

Outline

Outline

On the cost of cryptography for the IoT

- code size
- memory usage
- execution time
- efficiency on the high-end server?
- protections against side-channel attacks?

On the cost of cryptography for the IoT

- code size
- memory usage
- execution time
- efficiency on the high-end server?
- protections against side-channel attacks?

On the cost of cryptography for the IoT

- code size
- memory usage
- execution time
- efficiency on the high-end server?
- protections against side-channel attacks?

What are side-channel attacks?

- Leakage from the device
  - Time, electrical consumption, EM radiation
  - *simple power analysis* (**SPA**) vs *differential power analysis* (**DPA**)

Picture by oskay on Flickr

What are side-channel attacks?

- Inducing faults in the device
  - Glitch, laser pulse

Picture by ViaMoi on Flickr

Usage and ownership

Actors:

- Key owner
- Device owner
- Actual user

Usually, these are the same person, but...

Usage and ownership

When key owner $\neq$ device owner

- Banking card
- DRM

But hopefully the same person in open-source contexts!

Usage and ownership

When key/device owner $\neq$ actual user

- Not always controlling the device
  - E.g., devices spread over a large area
  - E.g., on-site personnel
  - E.g., lost device
- Distant eavesdropping

Protections against SCA can be needed.

Outline

Symmetric crypto: what textbooks and intro's say

Symmetric cryptographic primitives:

- Block ciphers
- Stream ciphers
- Hash functions

And their modes-of-use

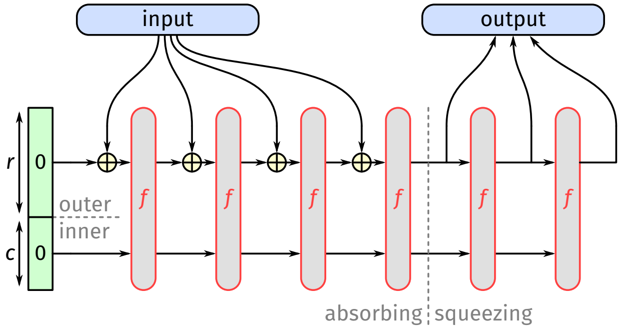
Picture by GlasgowAmateur

Examples of permutations

- In Salsa, Chacha, Grindhal...
- In SHA-3 candidates: CubeHash, Grøstl, JH, MD6, ...
- In CAESAR candidates: Ascon, Icepole, Norx, π-cipher, Primates, Stribob, ...

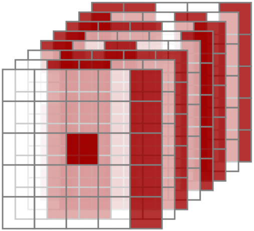
And of course in KECCAK

The sponge construction



- Calls a permutation f
- The capacity c determines the generic security:
  - Hashing: $2^{c/2}$
  - Authentication, encryption: $2^{c-\epsilon}$

KECCAK-f



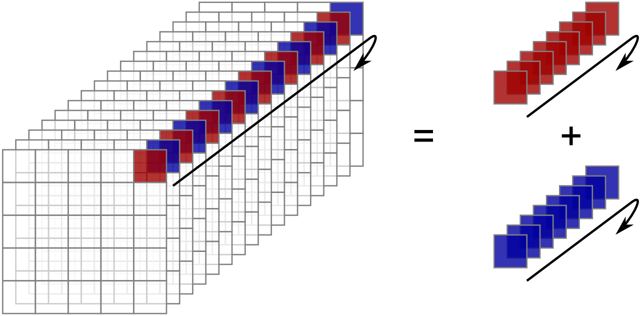
- The seven permutation army:
  - 25, 50, 100, 200, 400, 800, 1600 bits
  - toy, lightweight, fastest
  - standardized in [FIPS 202]
- Repetition of a simple round function
  - that operates on a 3D state
  - (5×5) lanes
  - up to 64-bit each

KECCAK-*f* in pseudo-code

```
KECCAK-F[b](A) {
  forall i in 0…nr-1
    A = Round[b](A, RC[i])
  return A
}

Round[b](A,RC) {
  θ step
  C[x] = A[x,0] xor A[x,1] xor A[x,2] xor A[x,3] xor A[x,4],  forall x in 0…4
  D[x] = C[x-1] xor rot(C[x+1],1),                            forall x in 0…4
  A[x,y] = A[x,y] xor D[x],                                   forall (x,y) in (0…4,0…4)

  ρ and π steps
  B[y,2*x+3*y] = rot(A[x,y], r[x,y]),                         forall (x,y) in (0…4,0…4)

  χ step
  A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y]),          forall (x,y) in (0…4,0…4)

  ι step
  A[0,0] = A[0,0] xor RC

  return A
}
```

https://keccak.team/keccak_specs_summary.html

Bit interleaving



$$\mathrm{ROT}_{64} \leftrightarrow 2 \times \mathrm{ROT}_{32}$$

The unbearable lightness of permutations

- Example: hashing with target security strength $2^{c/2}$
  - Davies-Meyer block cipher based hash
    - chaining value (block size): $n \geq c$
    - input block size ("key" length): typically $k \geq n$
    - feedforward (block size): n
    - $\Rightarrow$ total state $\geq 3c$
  - Sponge
    - permutation width: $c + r$
    - r can be made arbitrarily small, e.g., 1 byte
    - $\Rightarrow$ total state $\geq c + 8$

Cost of primitives and modes together

Conclusions

- Our multi-purpose Keccak outperforms our multi-purpose AES in terms of throughput over area by an average of 4.0.
- In Keyak mode our multi-purpose Keccak reaches 28.732 Gbps on Altera Stratix-IV, AES-GCM 5.586 Gbps.
- Typically a *plain* AES is much smaller than a *plain* Keccak.
- Addition of modes is more costly for AES than Keccak
  $\Rightarrow$ Keccak is more flexible than AES.

[Yalla, Homsirikamol, Kaps, DIAC 2014]

Symmetric crypto: a more correct picture

Symmetric cryptographic primitives:

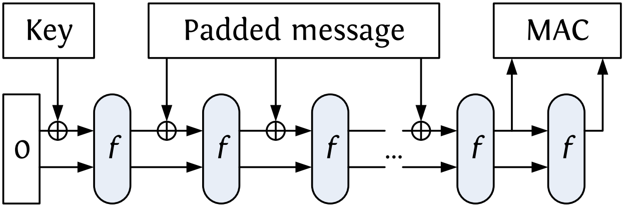
- Block ciphers
- Key stream generators
- **Permutations**

And their modes-of-use

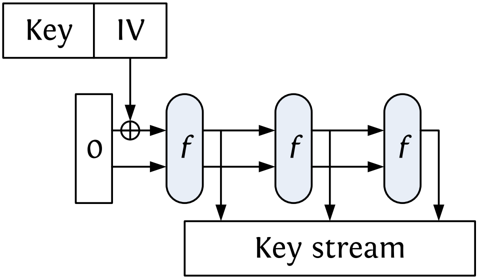
Picture by Sébastien Wiertz

Outline

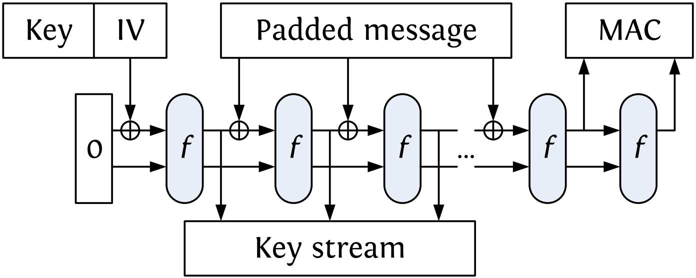
Use Sponge for MACing



Use Sponge for (stream) encryption

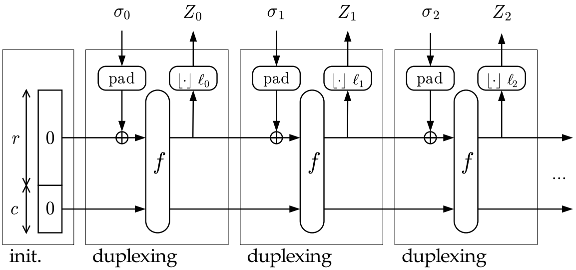


Single pass authenticated encryption



- But this is no longer the sponge ...

The duplex construction



- Generic security provably equivalent to that of sponge
- Applications: authenticated encryption, reseedable pseudorandom generator ...

Outline

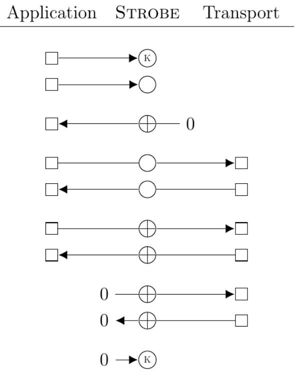
What is STROBE?

- Layer above the duplex construction
- Safe and easy syntax, to achieve, e.g.,
  - secure channels
  - signatures over a complete session
- Very compact implementation
- Mechanism to prevent side-channel attacks

[Mike Hamburg — https://strobe.sourceforge.io/]

Operations and data flow in STROBE

Abbr.	Operation	Flags
KEY	Secret key	*AC*
AD	Associated data	*A*
PRF	Hash / PRF	*IAC*
CLR	Send cleartext data	*A T*
recv-CLR	Receive cleartext data	*IA T*
ENC	Encrypt	*ACT*
recv-ENC	Decrypt	*IACT*
MAC	Compute MAC	*CT*
recv-ENC	Verify MAC	*I CT*
RATCHET	Rekey to prevent rollback	*C*



Legend: □ Send/recv ○ Absorb into sponge ⊕ Xor with cipher Ⓚ Roll key

figure courtesy of Mike Hamburg

Example: key derivation

- **KEY**(master shared key K)
- **RATCHET**
- derived key 1 $\leftarrow$ **PRF**(16 bytes)
- **RATCHET**
- derived key 2 $\leftarrow$ **PRF**(16 bytes)

Example: protocol

- **KEY**(shared key *K*)
- **AD**[nonce](sequence number *i*)
- **AD**[auth-data](client IP address | server IP address)
- **send_ENC**("GET file")
- **send_MAC**(128 bits)
- **recv_ENC**(buffer)
- **recv_MAC**(128 bits)

Outline

KETJE goals

- Nonce-based AE function
- 96-bit or 128-bit security (incl. multi-target)
- Sessions of header-body pairs
  - keeping the state during the session
- Small footprint
- Target niche: secure channel protocol on secure chips
  - banking card, ID, (U)SIM, secure element, FIDO, etc.
  - secure chip has strictly incrementing counter
- Using reduced-round KECCAK-$f[400]$ or KECCAK-$f[200]$, to allow
  - implementation re-use
  - cryptanalysis re-use
  - reasonable side-channel protections

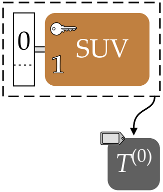
Ketje instances and lightweight features

feature		Ketje Jr	Ketje Sr
state size		25 bytes	50 bytes
block size		2 bytes	4 bytes
processing		**computational cost**	
initialization	per session	12 rounds	12 rounds
wrapping	per block	1 round	1 round
8-byte tag comp.	per message	9 rounds	7 rounds

KEYAK goals

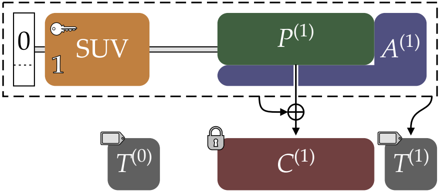
- Nonce-based AE function
- 128-bit security (incl. multi-target)
- Session of header-body pairs
  - keeping the state during the session
- Optionally parallelizable
- Conservative safety margin
- Using reduced-round KECCAK-$f[1600]$ or KECCAK-$f[800]$, to allow
  - implementation re-use
  - cryptanalysis re-use
  - reasonable side-channel protections

Keyak in a nutshell



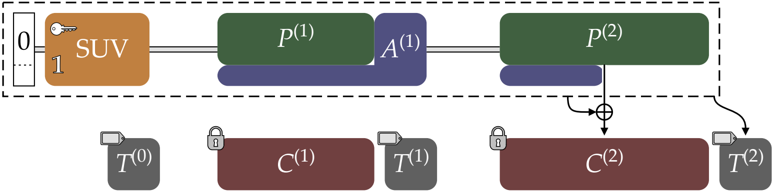
- SUV = Secret and Unique Value

Keyak in a nutshell



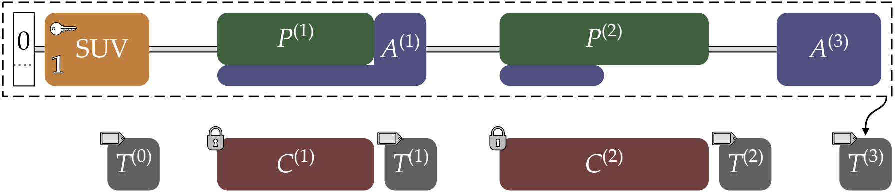
- SUV = Secret and Unique Value

KEYAK in a nutshell



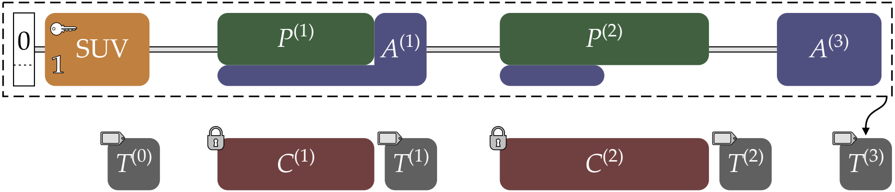
- SUV = Secret and Unique Value

KEYAK in a nutshell



- SUV = Secret and Unique Value

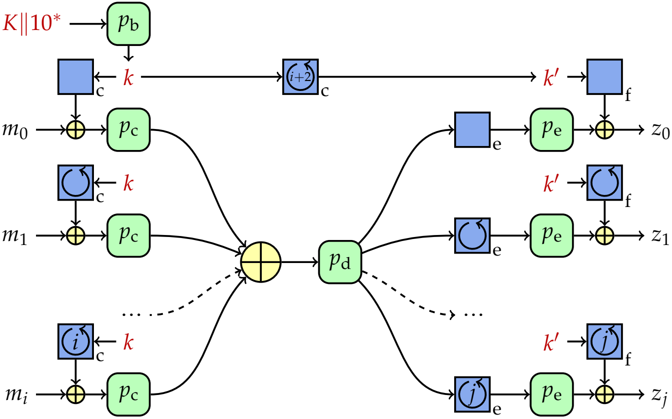
Leakage robustness



- SUV = Secret and Unique Value
- Provided that **uniqueness** is enforced
- then ...
  - the secret state is a *moving target* [Taha, Schaumont, HOST 2014]

Outline

The new Farfalle construction



[IACR ePrint 2016/1188]

Kravatte for many purposes

Kravatte = Farfalle + Keccak-$p[1600]$

Kravatte-PRF	Authentication
Kravatte-SAE	Session authenticated encryption
Kravatte-SIV	Synthetic-IV authenticated encryption
Kravatte-WBC	Wide block cipher, authenticated encryption with minimal expansion

Conclusions

- Permutations are well suited for IoT devices, especially for
  - code size
  - memory usage
- Farfalle brings efficiency also on the high-end server
- Bear in mind protections against side-channel attacks

Thanks for your attention!

Any questions?

Q?

https://keccak.team/

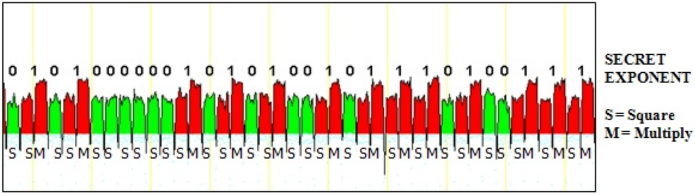
@KeccakTeam

A very classical example

RSA:

$$c^d \bmod n = m$$

Implemented using the *square & multiply* algorithm:



http://www.embedded.com/print/4199399

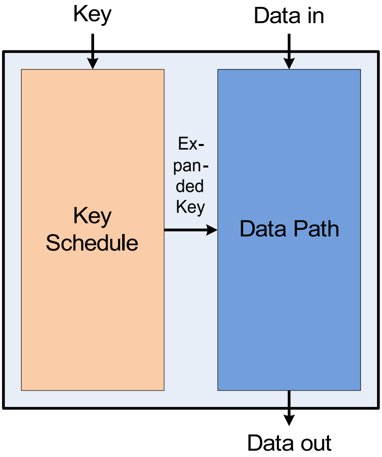
How to protect against side-channel attacks?

- Electrical-level countermeasures
  - E.g., balacing the processing of 0 and 1
- System-level countermeasures
  - E.g., limit the use of a key
- Algorithmic countermeasures
  - Randomization
  - E.g., instead of processing x, process y and z s.t. $x = y \oplus z$

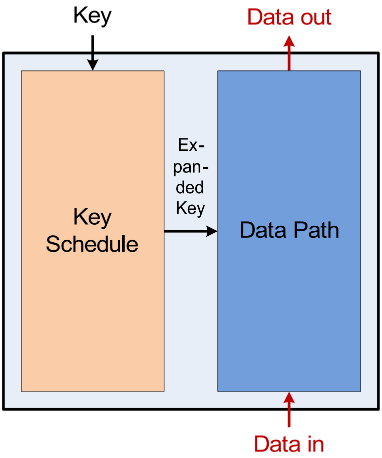
What block cipher are used for?

- Hashing: Davies-Meyer, ...
- Block encryption: ECB, CBC, ...
- Stream encryption:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- MAC computation: CBC-MAC, C-MAC, ...
- Authenticated encryption: OCB, GCM, CCM ...

Block cipher operation



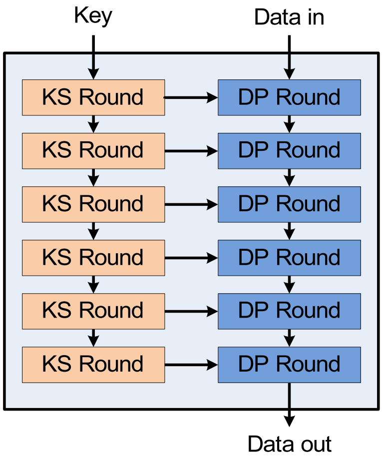
Block cipher operation: the inverse



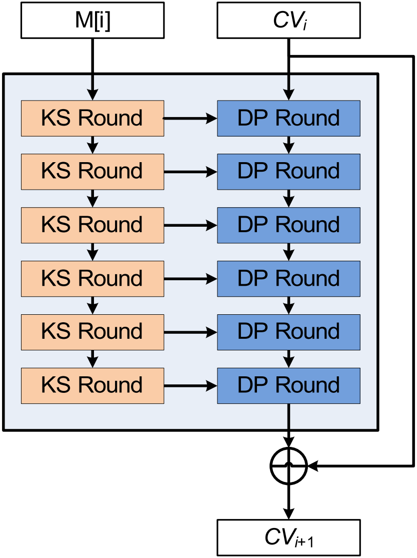
When do you need the inverse?

- Hashing and its modes HMAC, MGF1, ...
- Block encryption: ECB, CBC, ...
- Stream encryption:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- MAC computation: CBC-MAC, C-MAC, ...
- Authenticated encryption: OCB, GCM, CCM ...

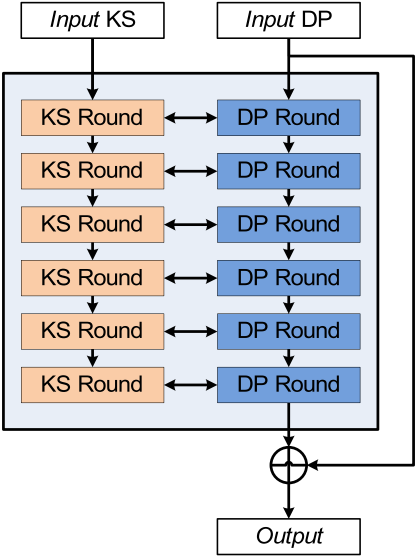
Block cipher internals



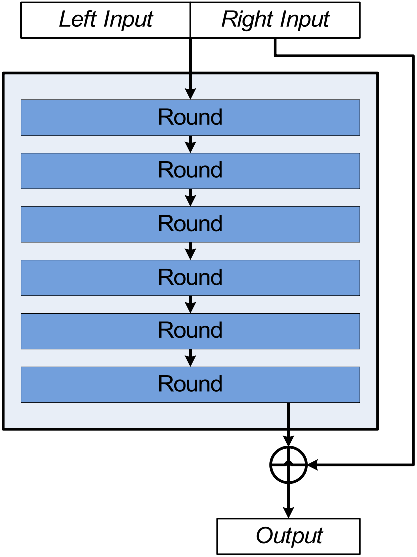
Hashing using Davies-Meyer



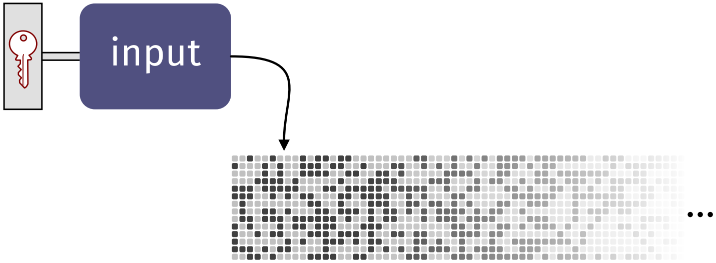
Removing diffusion restrictions



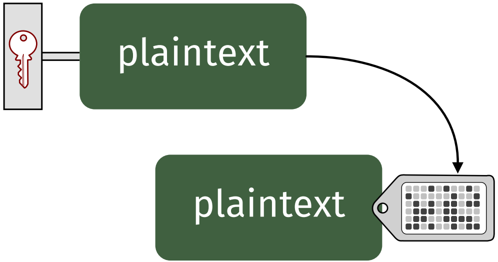
Simplifying the view: iterated permutation



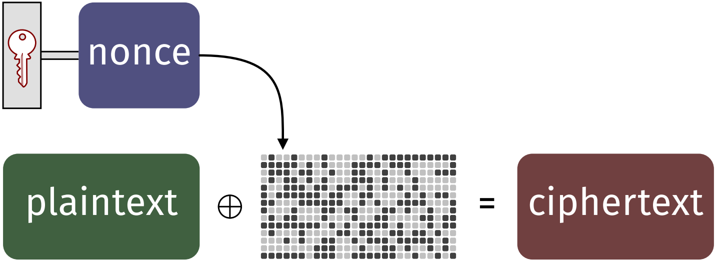
Pseudo-random function (PRF)



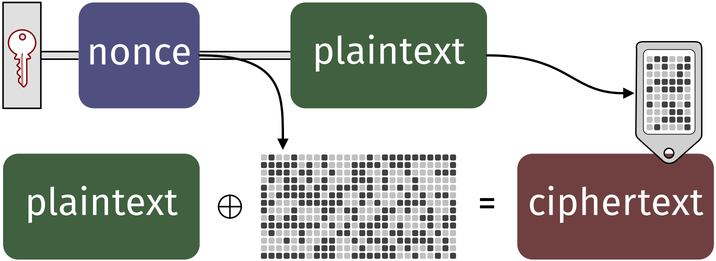
Message authentication code (MAC)



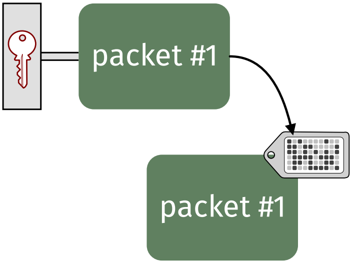
Stream cipher



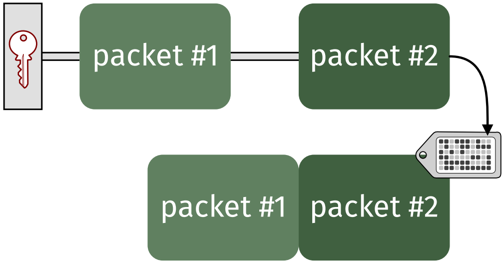
Authenticated encryption



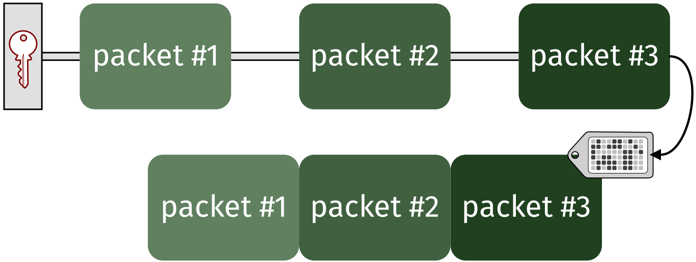
Incrementality



Incrementality



Incrementality



In-place processing

Store $A[x, y]$ at round i in (x', y') with

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}^i \begin{pmatrix} x \\ y \end{pmatrix}.$$

- Interacts with π: the output of χ can overwrite its input
- Matrix of order 4
  - $\Rightarrow$ no performance loss if 4 rounds unrolled

[Bertoni et al., Keccak implementation overview]

In-place processing

Store $A[x, y]$ at round i in (x', y') with

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}^i \begin{pmatrix} x \\ y \end{pmatrix}.$$

- Interacts with π: the output of χ can overwrite its input
- Matrix of order 4
  - $\Rightarrow$ no performance loss if 4 rounds unrolled

[Bertoni et al., KECCAK implementation overview]

In-place processing

Store $A[x, y]$ at round i in (x', y') with

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}^i \begin{pmatrix} x \\ y \end{pmatrix}.$$

- Interacts with π: the output of χ can overwrite its input
- Matrix of order 4
  - $\Rightarrow$ no performance loss if 4 rounds unrolled

[Bertoni et al., Keccak implementation overview]