XKCP internals

Gilles Van Assche[1]

[1]STMicroelectronics

SCA workshop
Šibenik, Croatia, June 2019

Based on joint work with Ronny Van Keer

Outline

Outline

What is the XKCP?

Previously known as the KECCAK Code Package...

Repository of implementations of

- KECCAK-p[200 to 1600], SHA-3, (c)SHAKE, KMAC, KANGAROOTWELVE, KETJE, KEYAK, KRAVATTE, ...
- XOODOO, XOODYAK, XOOFFF, XOOFFF-SANE, ...

... now eXtended KCP

What is the XKCP?

Previously known as the KECCAK Code Package...

Repository of implementations of

- KECCAK-p[200 to 1600], SHA-3, (c)SHAKE, KMAC, KANGAROOTWELVE, KETJE, KEYAK, KRAVATTE, ...
- XOODOO, XOODYAK, XOOFFF, XOOFFF-SANE, ...

... now eXtended KCP

What is the XKCP?

Previously known as the KECCAK Code Package...

Repository of implementations of

- KECCAK-p[200 to 1600], SHA-3, (c)SHAKE, KMAC, KANGAROOTWELVE, KETJE, KEYAK, KRAVATTE, ...
- XOODOO, XOODYAK, XOOFFF, XOOFFF-SANE, ...

... now eXtended KCP

What is the XKCP?

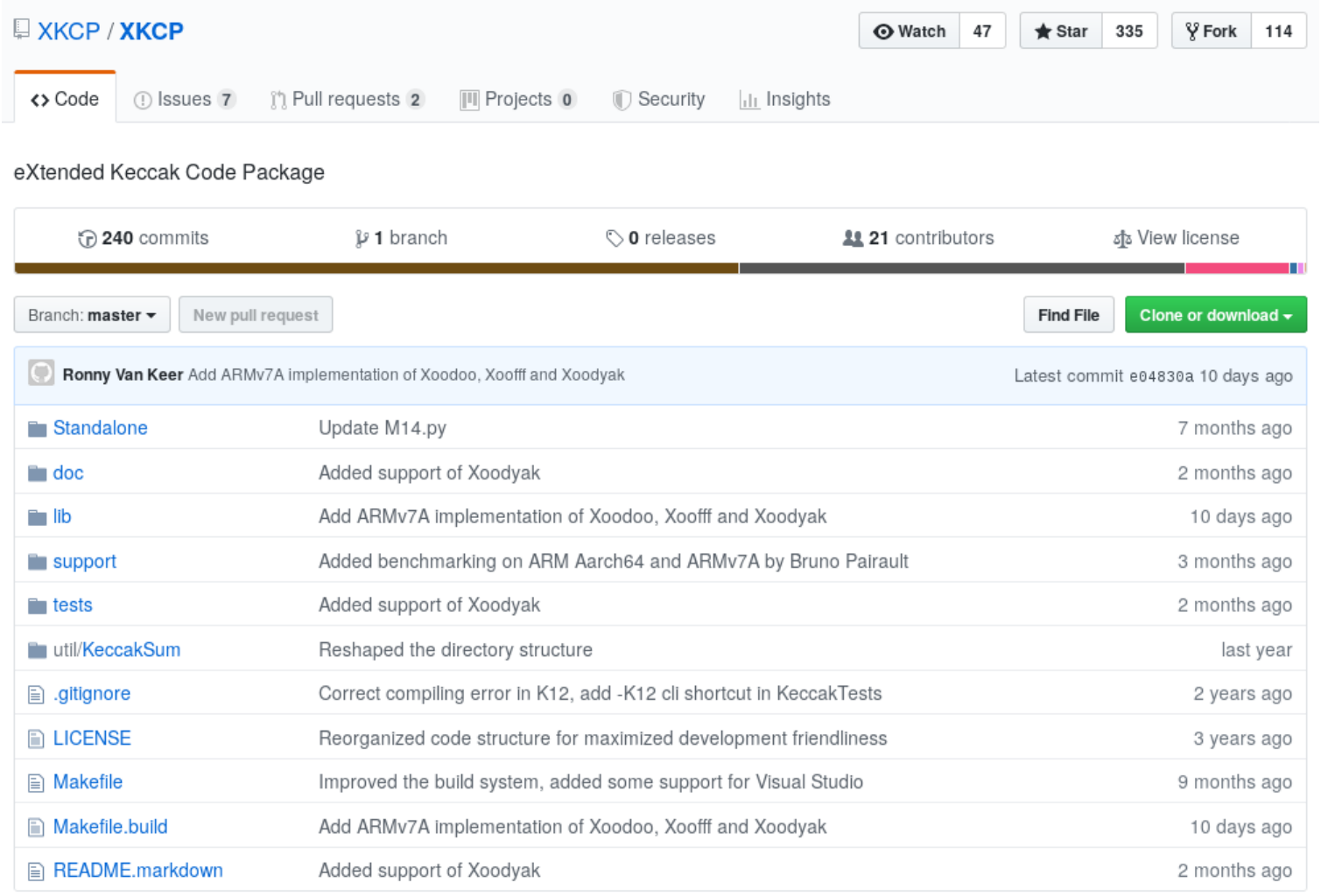
Previously known as the KECCAK Code Package...

Repository of implementations of

- KECCAK-$p[200 \text{ to } 1600]$, SHA-3, (c)SHAKE, KMAC, KANGAROOTWELVE, KETJE, KEYAK, KRAVATTE, ...
- XOODOO, XOODYAK, XOOFFF, XOOFFF-SANE, ...

... now eXtended KCP

Where to find it



XKCP / XKCP

Watch 47 | Star 335 | Fork 114

Code | Issues 7 | Pull requests 2 | Projects 0 | Security | Insights

eXtended Keccak Code Package

240 commits | 1 branch | 0 releases | 21 contributors | View license

Branch: master | New pull request | Find File | Clone or download

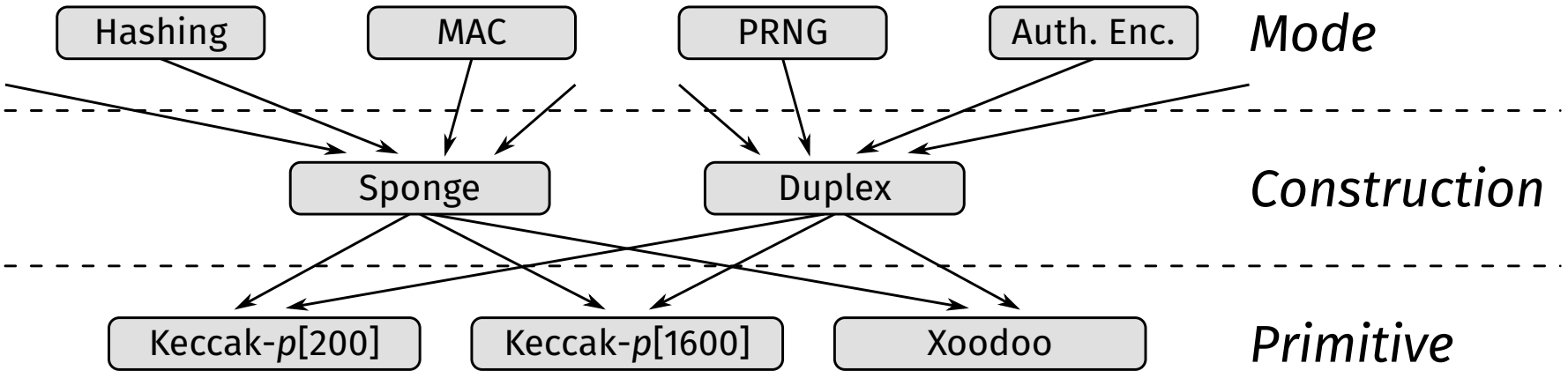
Ronny Van Keer Add ARMv7A implementation of Xoodoo, Xoofff and Xoodyak — Latest commit e04830a 10 days ago

Standalone	Update M14.py	7 months ago
doc	Added support of Xoodyak	2 months ago
lib	Add ARMv7A implementation of Xoodoo, Xoofff and Xoodyak	10 days ago
support	Added benchmarking on ARM Aarch64 and ARMv7A by Bruno Pairault	3 months ago
tests	Added support of Xoodyak	2 months ago
util/KeccakSum	Reshaped the directory structure	last year
.gitignore	Correct compiling error in K12, add -K12 cli shortcut in KeccakTests	2 years ago
LICENSE	Reorganized code structure for maximized development friendliness	3 years ago
Makefile	Improved the build system, added some support for Visual Studio	9 months ago
Makefile.build	Add ARMv7A implementation of Xoodoo, Xoofff and Xoodyak	10 days ago
README.markdown	Added support of Xoodyak	2 months ago

https://github.com/XKCP/XKCP

Outline

A layered approach



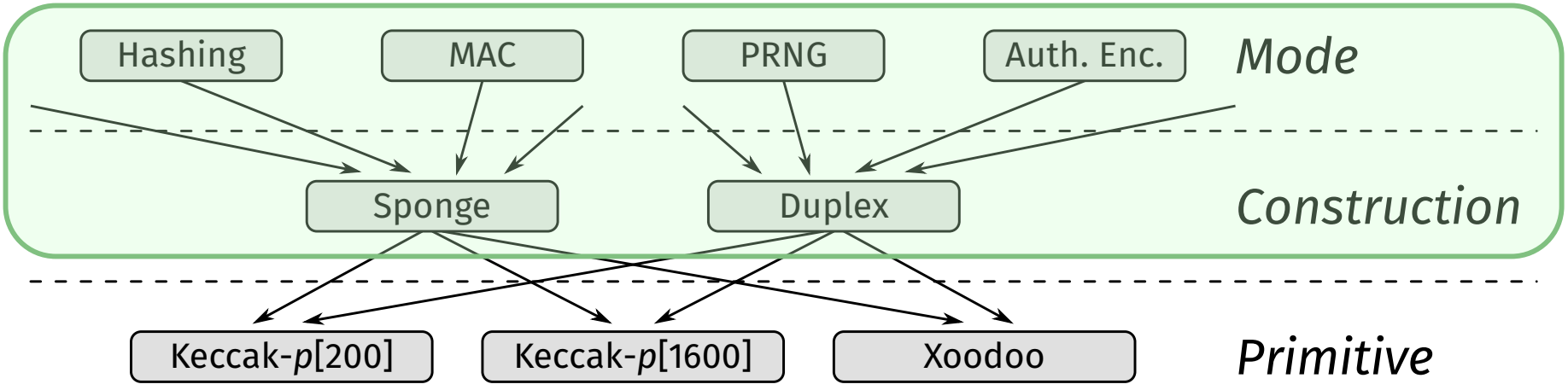
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

A layered approach



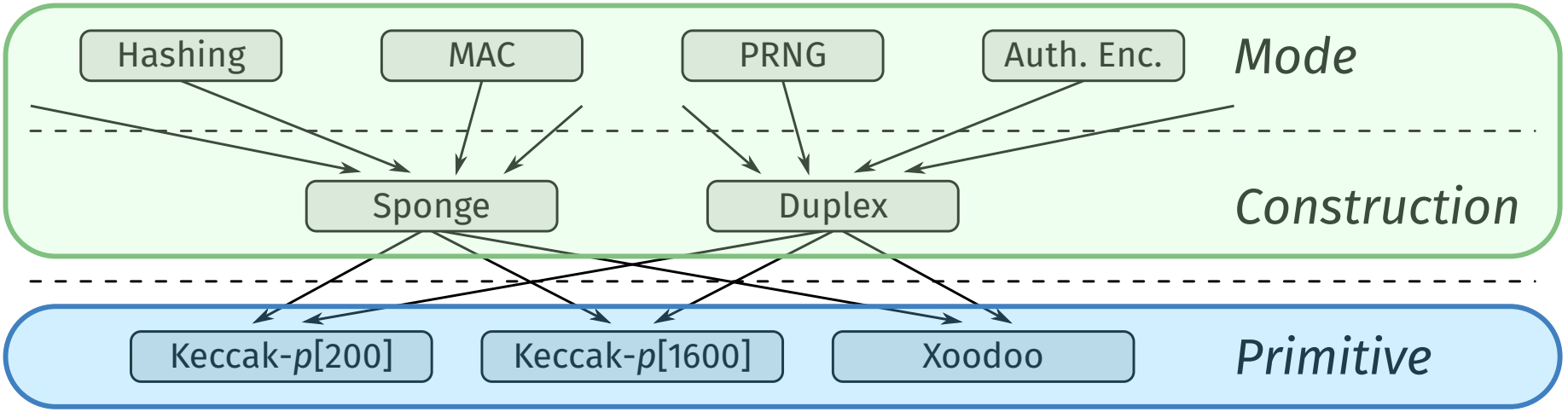
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

A layered approach



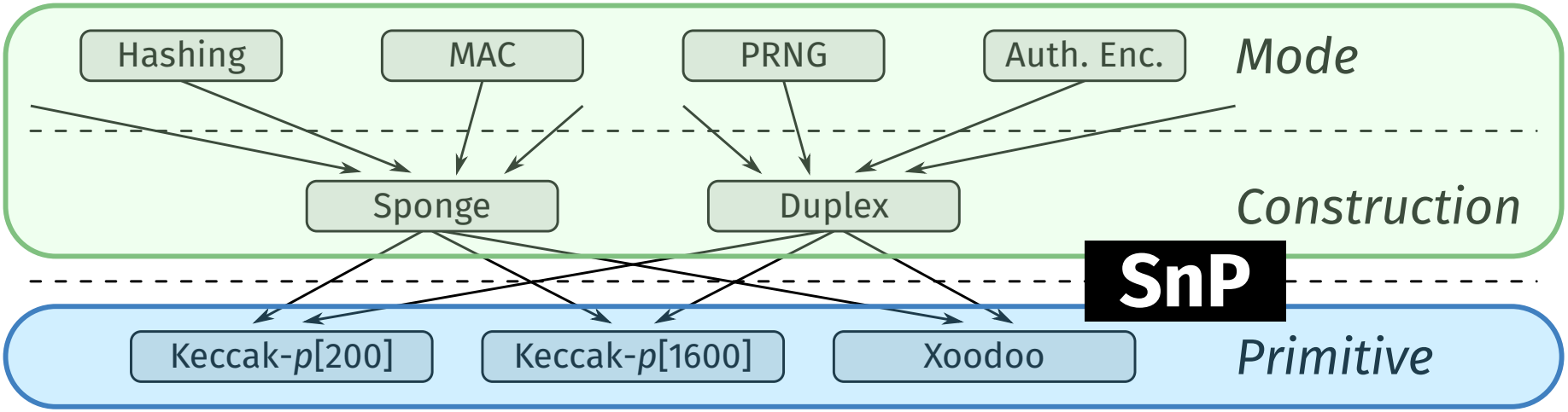
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

A layered approach



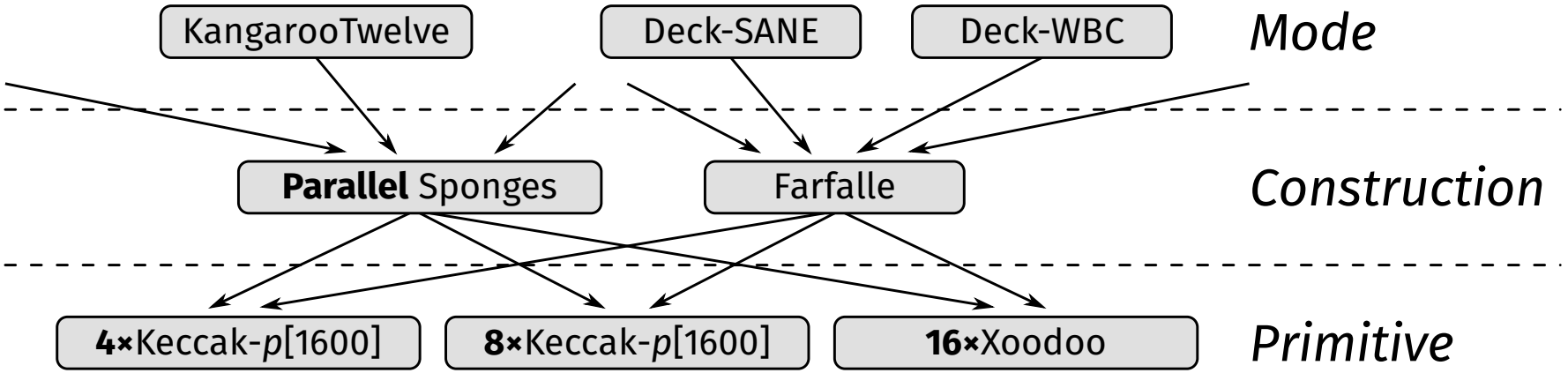
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

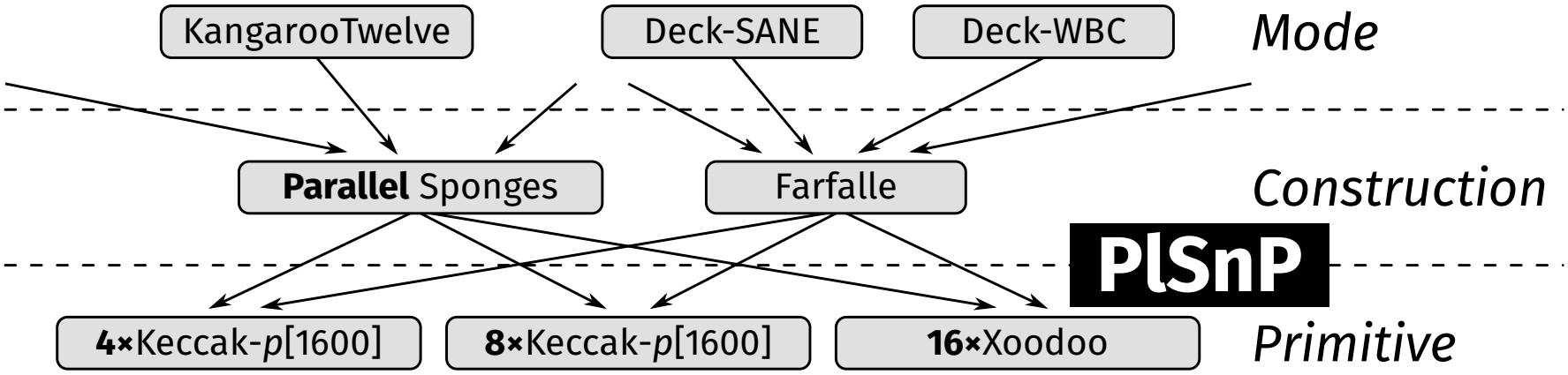
- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

Parallel processing



- SnP = State and Permutation
- PlSnP = Parallel States and Permutations

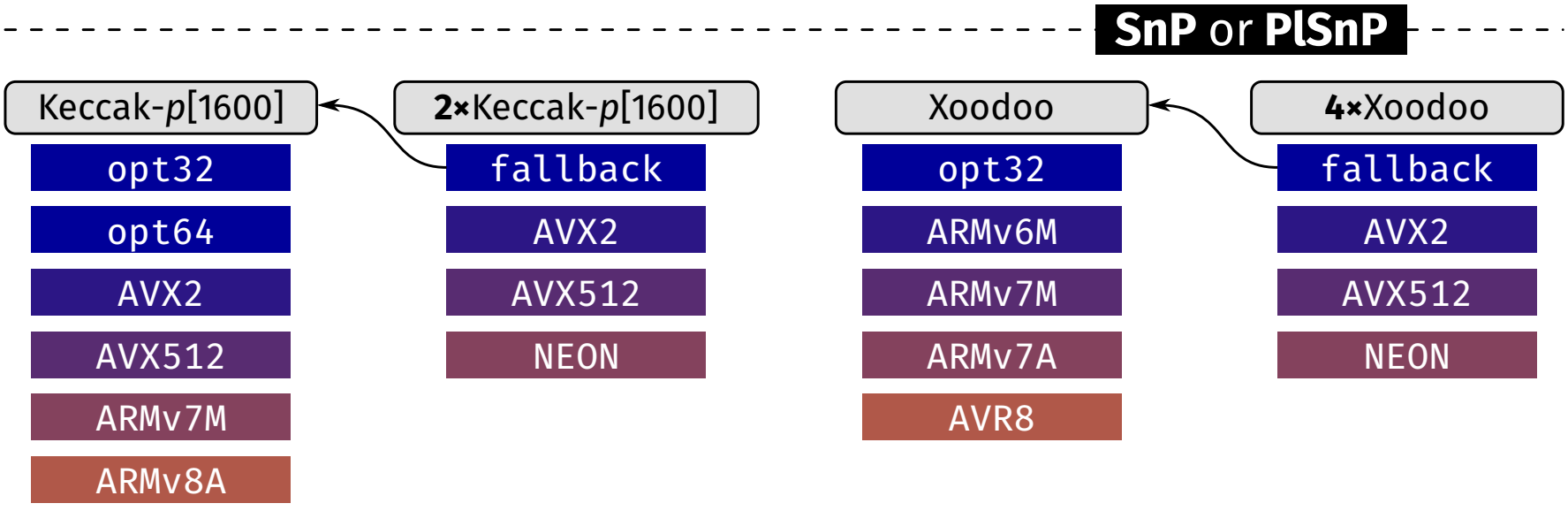
Parallel processing



- SnP = State and Permutation
- PlSnP = Parallel States and Permutations

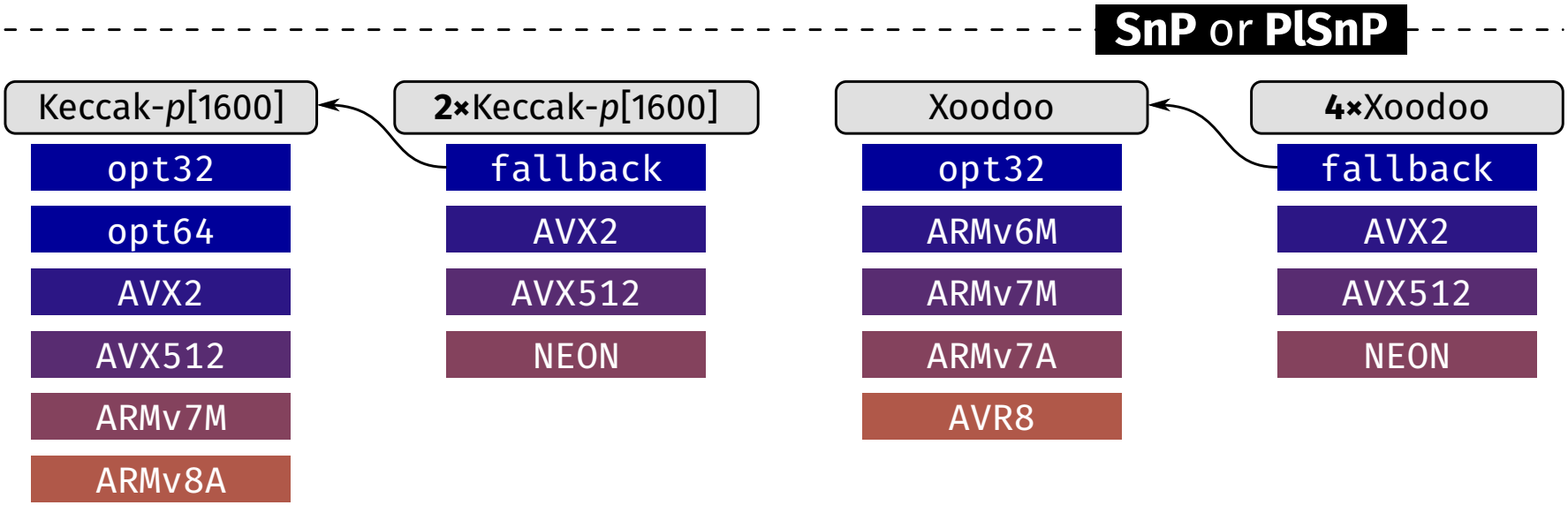
Outline

Multiple implementations



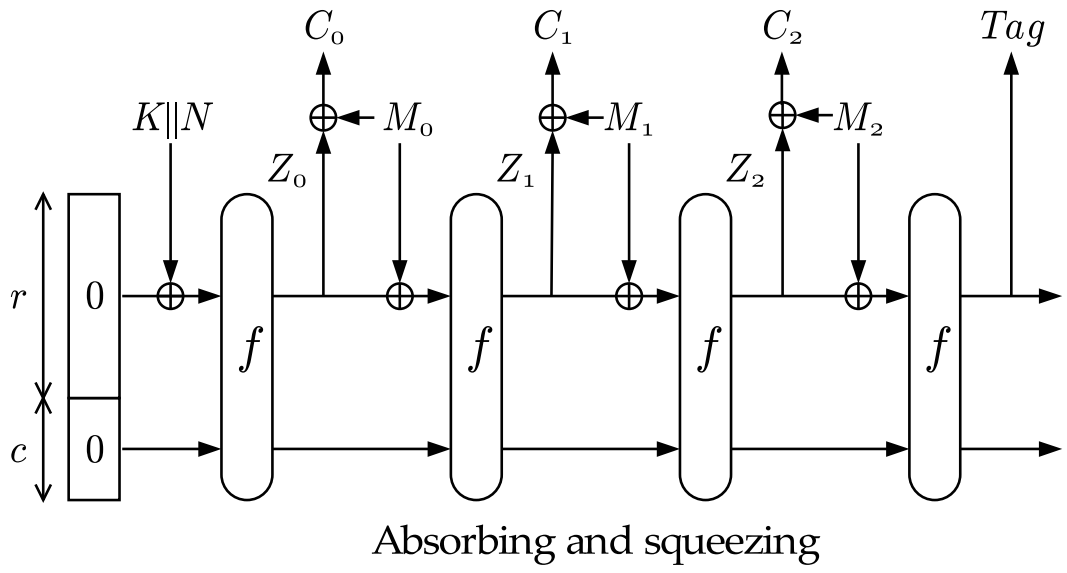
Assumption: **the state representation is opaque!**

Multiple implementations



Assumption: **the state representation is opaque!**

Operations on the state



Absorbing and squeezing

- initialize the state
- apply the permutation f
- XOR/overwrite bytes into the state
- extract bytes from the state
  - and optionally XOR them

Example for KECCAK-*p*[1600]

Declarations in **KeccakP-1600-SnP.h**:

- `#define KeccakP1600_implementation "generic 64-bit optimized implementation"`
- `#define KeccakP1600_stateSizeInBytes 200`
- `#define KeccakP1600_stateAlignment 8`

Typical functions to be implemented (or macro'ed):

- `KeccakP1600_Initialize`
- `KeccakP1600_AddBytes`
- `KeccakP1600_OverwriteBytes`
- `KeccakP1600_Permute_Nrounds`
- `KeccakP1600_ExtractBytes`
- `KeccakP1600_ExtractAndAddBytes`

Example for KECCAK-*p*[1600]

Declarations in **KeccakP-1600-SnP.h**:

- `#define KeccakP1600_implementation "generic 64-bit optimized implementation"`
- `#define KeccakP1600_stateSizeInBytes 200`
- `#define KeccakP1600_stateAlignment 8`

Typical functions to be implemented (or macro'ed):

- `KeccakP1600_Initialize`
- `KeccakP1600_AddBytes`
- `KeccakP1600_OverwriteBytes`
- `KeccakP1600_Permute_Nrounds`
- `KeccakP1600_ExtractBytes`
- `KeccakP1600_ExtractAndAddBytes`

Example for KECCAK-p[1600]

Declarations in **KeccakP-1600-SnP.h**:

- `#define KeccakP1600_implementation "generic 64-bit optimized implementation"`
- `#define KeccakP1600_stateSizeInBytes 200`
- `#define KeccakP1600_stateAlignment 8`

Typical functions to be implemented (or macro'ed):

- `KeccakP1600_Initialize`
- `KeccakP1600_AddBytes`
- `KeccakP1600_OverwriteBytes`
- `KeccakP1600_Permute_Nrounds`
- `KeccakP1600_ExtractBytes`
- `KeccakP1600_ExtractAndAddBytes`

Example for KECCAK-p[1600]

Declarations in **KeccakP-1600-SnP.h**:

- `#define KeccakP1600_implementation "generic 64-bit optimized implementation"`
- `#define KeccakP1600_stateSizeInBytes 200`
- `#define KeccakP1600_stateAlignment 8`

Typical functions to be implemented (or macro'ed):

- `KeccakP1600_Initialize`
- `KeccakP1600_AddBytes`
- `KeccakP1600_OverwriteBytes`
- `KeccakP1600_Permute_Nrounds`
- `KeccakP1600_ExtractBytes`
- `KeccakP1600_ExtractAndAddBytes`

Example for KECCAK-*p*[1600]

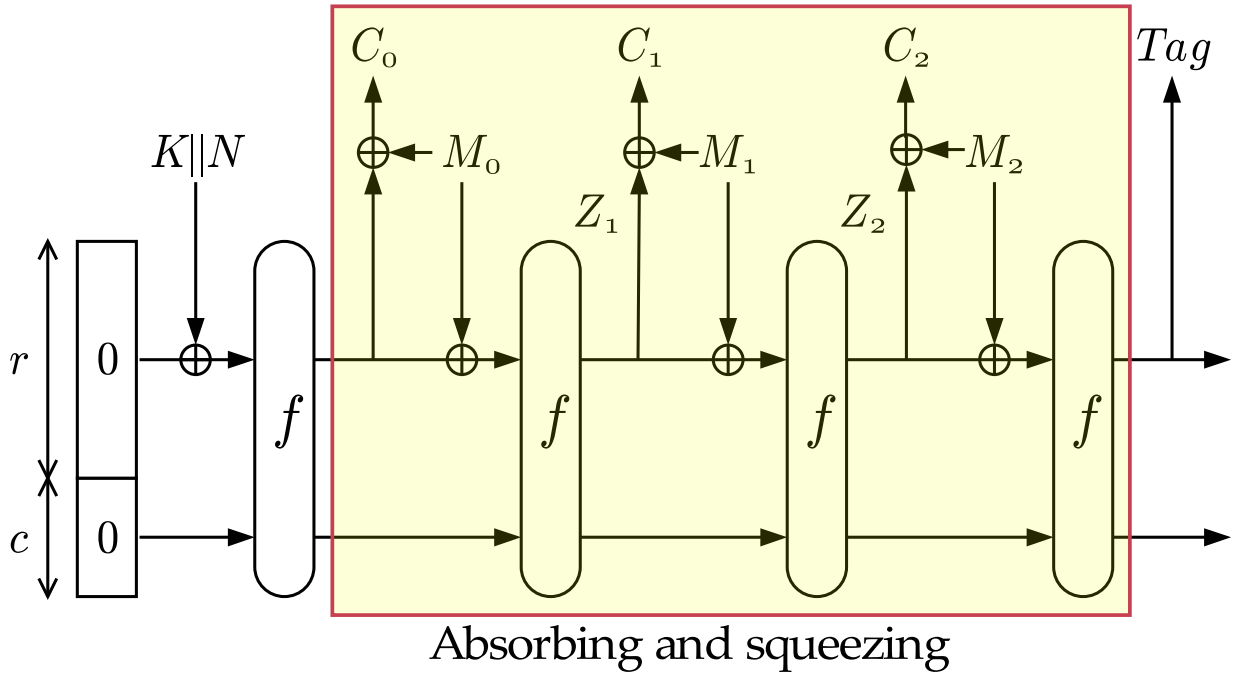
Declarations in **KeccakP-1600-SnP.h**:

- `#define KeccakP1600_implementation "generic 64-bit optimized implementation"`
- `#define KeccakP1600_stateSizeInBytes 200`
- `#define KeccakP1600_stateAlignment 8`

Typical functions to be implemented (or macro'ed):

- `KeccakP1600_Initialize`
- `KeccakP1600_AddBytes`
- `KeccakP1600_OverwriteBytes`
- `KeccakP1600_Permute_Nrounds`
- `KeccakP1600_ExtractBytes`
- `KeccakP1600_ExtractAndAddBytes`

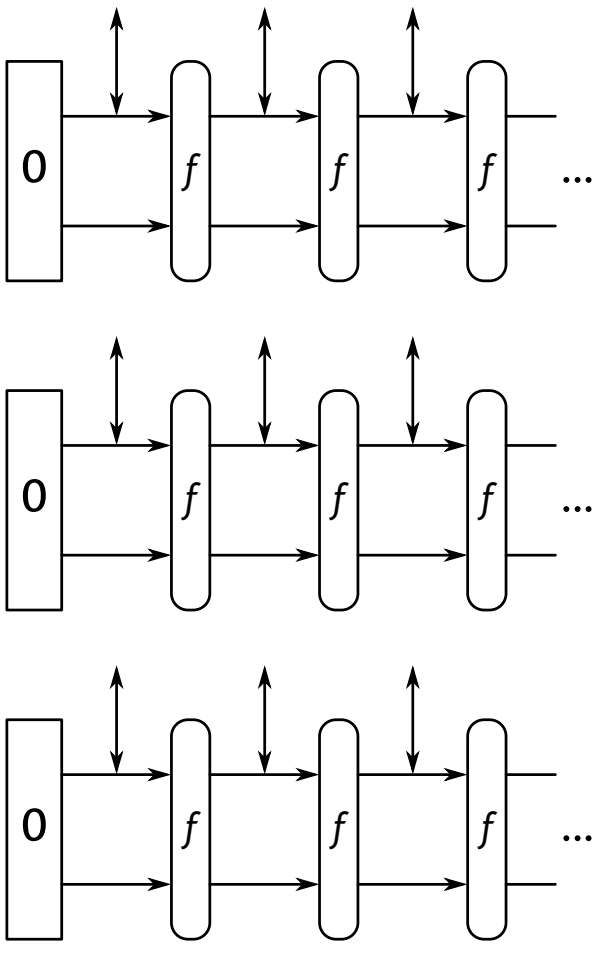
Fast loop optimization



Specialized repeated application of some operations

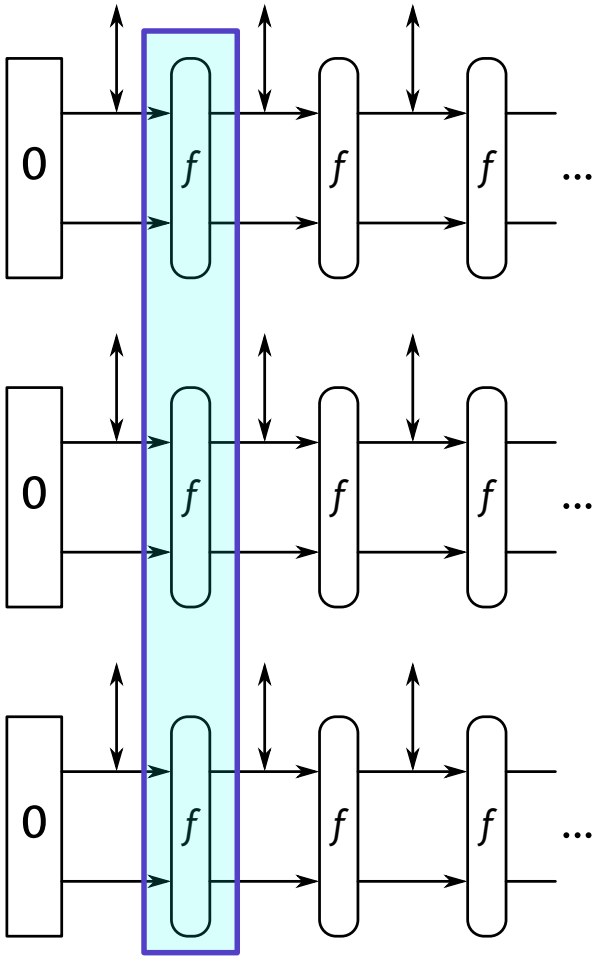
(optional)

Parallel operations on the states



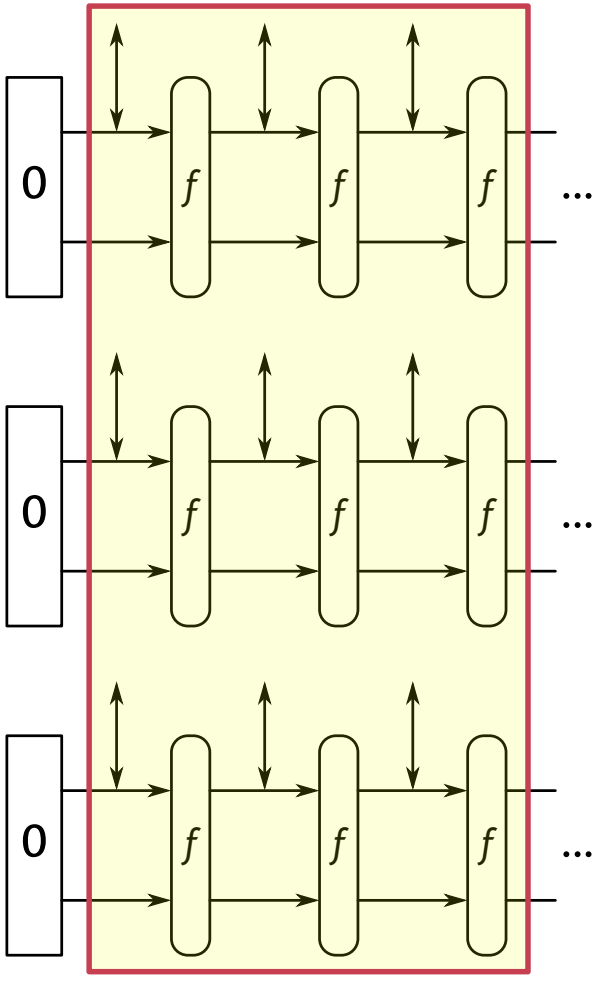
- Functions on individual instances
- Functions on **all instances**
  - Parallel application of f
  - XOR blocks into state
- Optional **fast loop** optimization

Parallel operations on the states



- Functions on individual instances
- Functions on **all instances**
  - Parallel application of f
  - XOR blocks into state
- Optional **fast loop** optimization

Parallel operations on the states



- Functions on individual instances
- Functions on **all instances**
  - Parallel application of f
  - XOR blocks into state
- Optional **fast loop** optimization

Outline

Making targets

- Making a library (**.a** or **.so**)
  - `make generic64/libXKCP.a`
  - `make generic32/libXKCP.a`
  - `make Skylake/libXKCP.a`
  - `make ARMv7A/libXKCP.a`
  - `make compact/libXKCP.a`
- Extracting the source files
  - `make generic64/libXKCP.a.pack`
- Running the unit tests
  - `make generic64/UnitTests`
  - `UnitTests --SnP --KangarooTwelve --Xoofff`
- And more: benchmarks, KeccakSum utility

Making targets

- Making a library (**.a** or **.so**)
  - `make generic64/libXKCP.a`
  - `make generic32/libXKCP.a`
  - `make Skylake/libXKCP.a`
  - `make ARMv7A/libXKCP.a`
  - `make compact/libXKCP.a`
- Extracting the source files
  - `make generic64/libXKCP.a.pack`
- Running the unit tests
  - `make generic64/UnitTests`
  - `UnitTests --SnP --KangarooTwelve --Xoofff`
- And more: benchmarks, KeccakSum utility

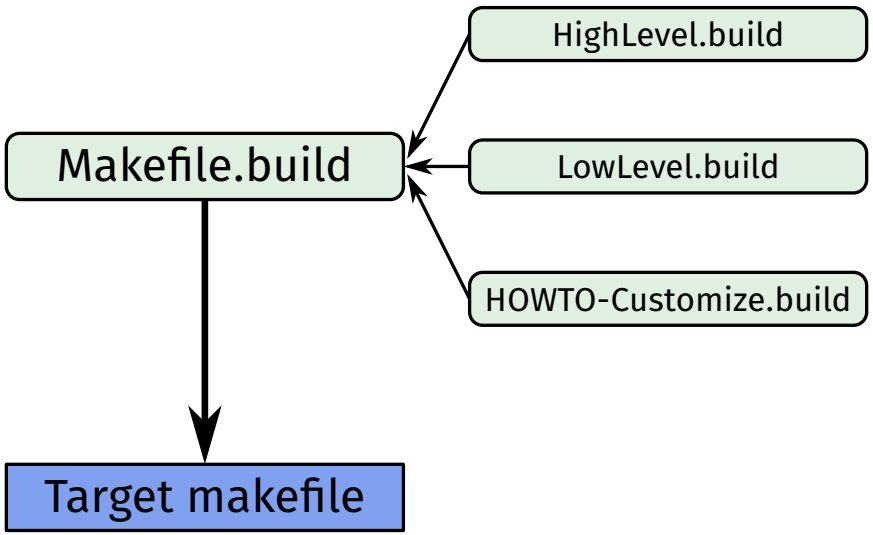
Making targets

- Making a library (**.a** or **.so**)
  - `make generic64/libXKCP.a`
  - `make generic32/libXKCP.a`
  - `make Skylake/libXKCP.a`
  - `make ARMv7A/libXKCP.a`
  - `make compact/libXKCP.a`
- Extracting the source files
  - `make generic64/libXKCP.a.pack`
- Running the unit tests
  - `make generic64/UnitTests`
  - `UnitTests --SnP --KangarooTwelve --Xoofff`
- And more: benchmarks, KeccakSum utility

Making targets

- Making a library (**.a** or **.so**)
  - `make generic64/libXKCP.a`
  - `make generic32/libXKCP.a`
  - `make Skylake/libXKCP.a`
  - `make ARMv7A/libXKCP.a`
  - `make compact/libXKCP.a`
- Extracting the source files
  - `make generic64/libXKCP.a.pack`
- Running the unit tests
  - `make generic64/UnitTests`
  - `UnitTests --SnP --KangarooTwelve --Xoofff`
- And more: benchmarks, KeccakSum utility

XML-driven Makefile



In `*.build`, one defines **fragments** that ...

- ... are sets of files and compilation options
- ... represent
  - a concrete service, mode, construction
  - the implementation of a permutation

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3"/>`
- `<target name="K12onAVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3"/>`
- `<target name="K12onAVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak"/>`
- `<target name="K12onAVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak optimizedARMv7M"/>`
- `<target name="K12onAVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak optimizedARMv7M"/>`
- `<target name="K12onAVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak optimizedARMv7M"/>`
- `<target name="K12onAVX2" inherits="KangarooTwelve"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak optimizedARMv7M"/>`
- `<target name="K12onAVX2" inherits="KangarooTwelve optimized1600AVX2"/>`

Customizing targets

HOWTO-Customize.build

Examples:

- `<target name="XoodyakCortexM3" inherits="Xoodyak optimizedARMv7M"/>`
- `<target name="K12onAVX2" inherits="KangarooTwelve optimized1600AVX2 SIMD256-AVX2u12"/>`

Any questions?

Thanks for your attention!

`https://keccak.team/`