IDEA
Past - Present - Future

Pascal Junod
(joint work with Marco Macchetti, Nagravision SA)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

ESC'10 - January 14, 2010
Remich (Luxembourg)

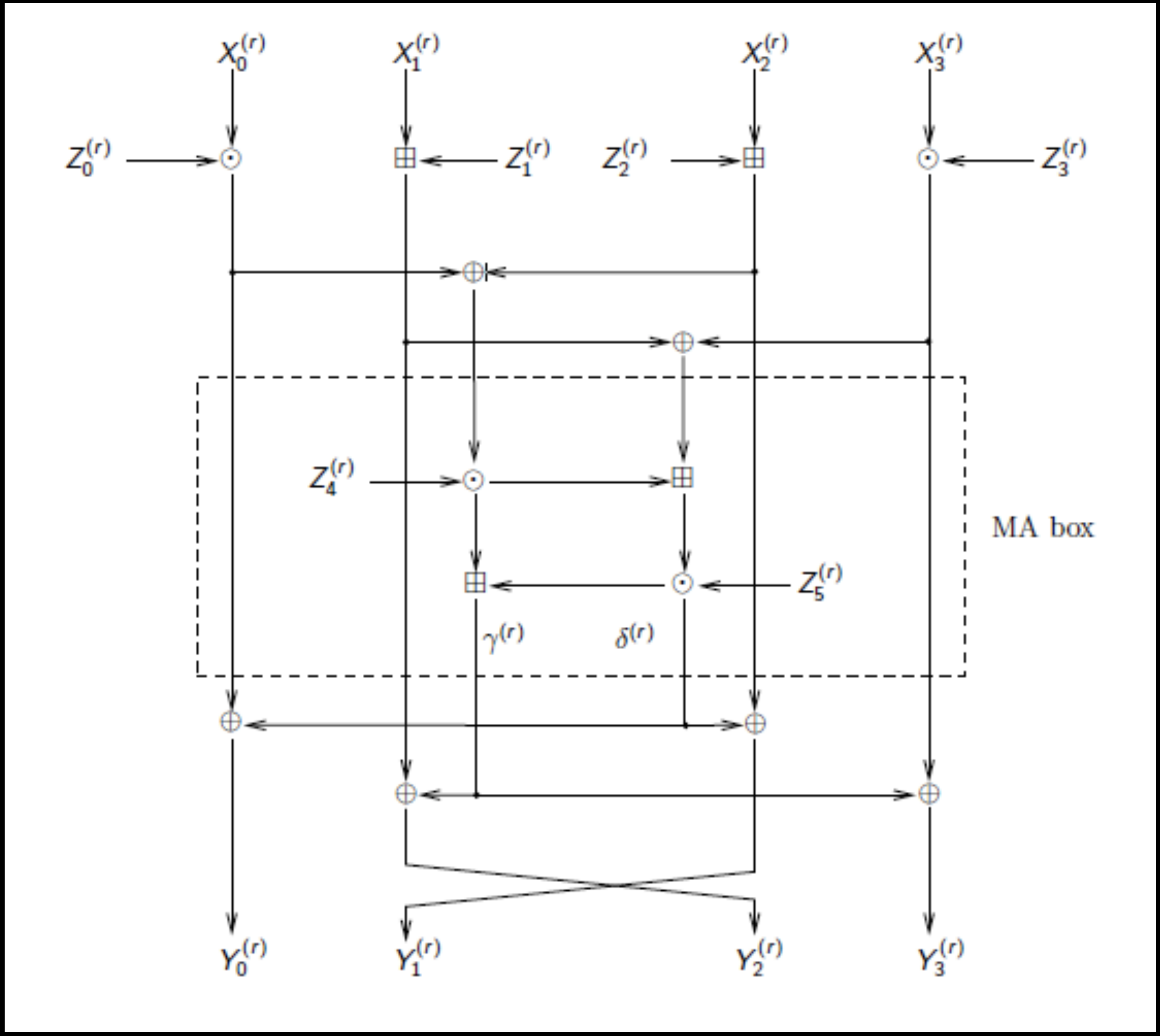
Outline

> IDEA
> WIDEA
> HIDEA

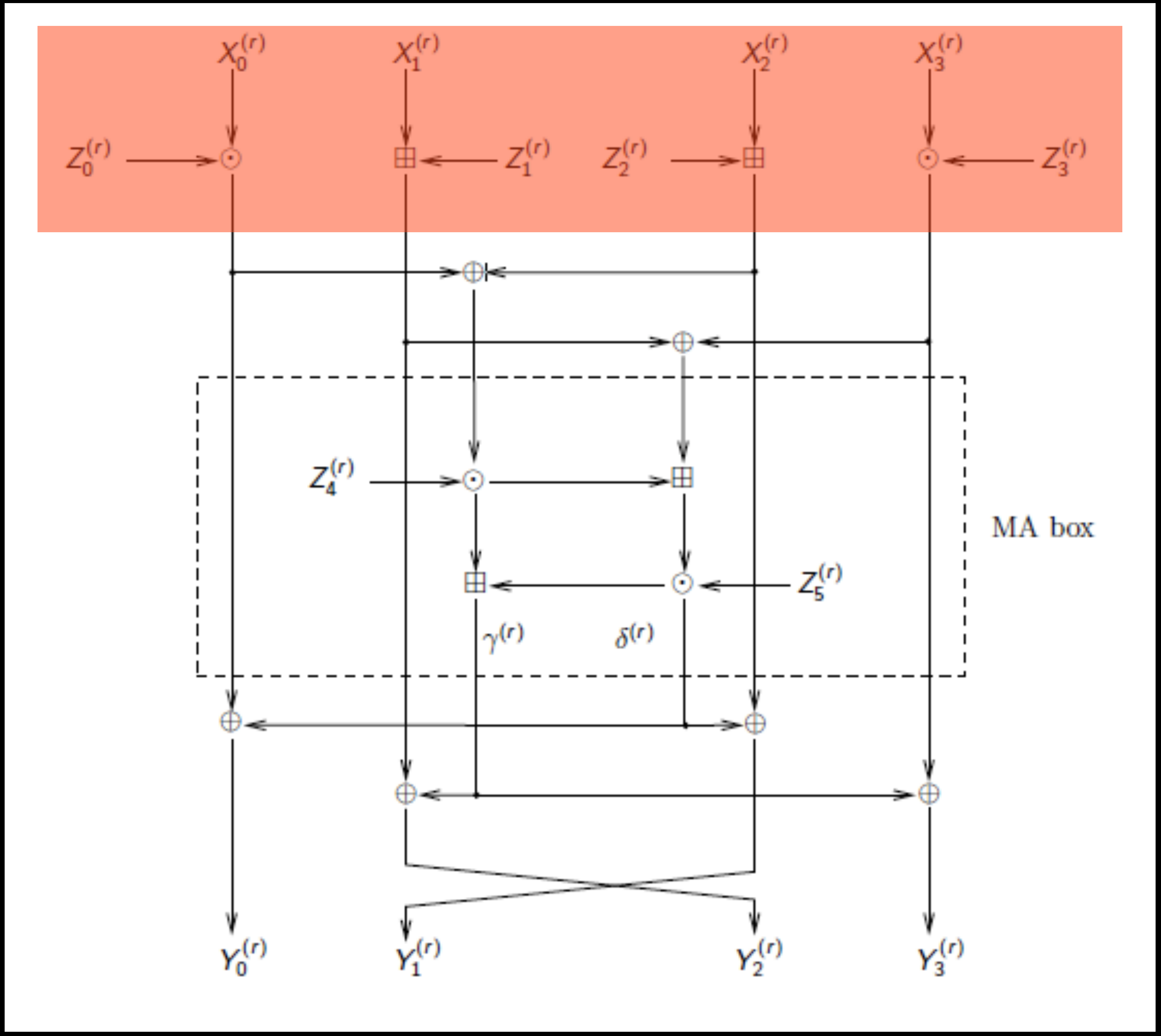
IDEA // A Bit of History

> Block cipher designed by Lai and Massey in 1990 on behalf of Ascom AG
> 64-bit block, 128-bit key
> Very simple philosophy
  > Mix three different and algebraically incompatible group laws on 16-bit words
    > $\oplus$ ... addition over $(\mathbb{Z}/2\mathbb{Z})^{16}$
    > $\boxplus$ ... addition over $\mathbb{Z}/(2^{16}\mathbb{Z})$
    > $\odot$ ... multiplication over $\mathbb{Z}^*_{2^{16}+1}$
> Simple, fully linear bit-selecting key-schedule algorithm
> Quite popular during the 90's thanks to PGP
> Mostly used today as a LUF (*L*egally *U*nclonable *F*unction)

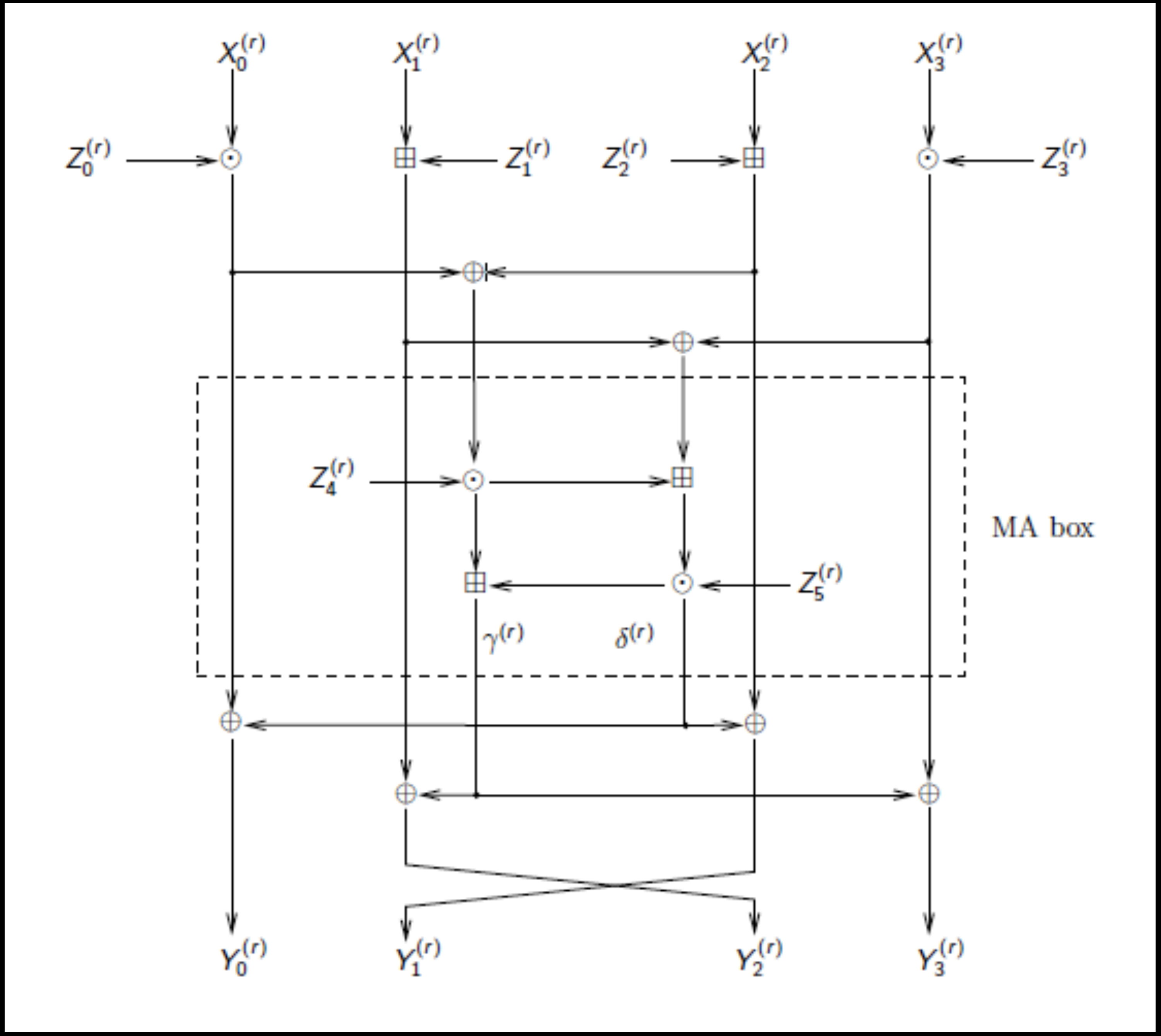
IDEA // Round Function



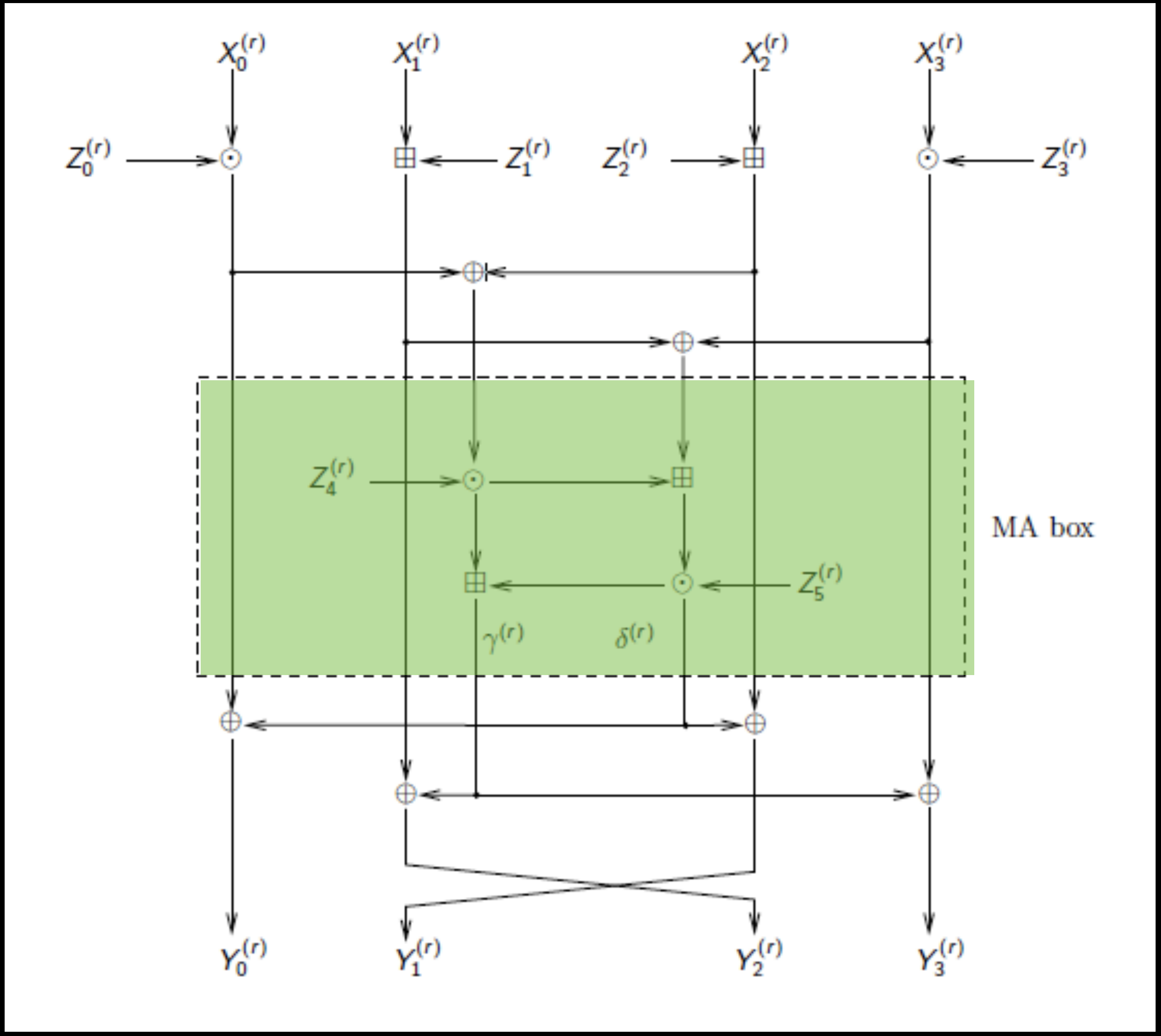
IDEA // Round Function



IDEA // Round Function



IDEA // Round Function



IDEA // Current Security Level

> Designed to resist differential cryptanalysis
> Extensively cryptanalyzed
  > More than 15 published paper so far
> As of today, the best attack by Sun and Lai `[SunLai09]` breaks 6 out of 8.5 rounds with help of 2^{49} chosen plaintexts and $2^{112.1}$ encryption operations (in a classical scenario)
> Virtually all the attacks largely exploit properties of the fully linear key scheduling

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0\ldots15]}$	$Z_{[16\ldots31]}$	$Z_{[32\ldots47]}$	$Z_{[48\ldots63]}$	$Z_{[64\ldots79]}$	$Z_{[80\ldots95]}$
2	$Z_{[96\ldots111]}$	$Z_{[112\ldots127]}$	$Z_{[25\ldots40]}$	$Z_{[41\ldots56]}$	$Z_{[57\ldots72]}$	$Z_{[73\ldots88]}$
3	$Z_{[89\ldots104]}$	$Z_{[105\ldots120]}$	$Z_{[121\ldots8]}$	$Z_{[9\ldots24]}$	$Z_{[50\ldots65]}$	$Z_{[66\ldots81]}$
4	$Z_{[82\ldots97]}$	$Z_{[98\ldots113]}$	$Z_{[114\ldots1]}$	$Z_{[2\ldots17]}$	$Z_{[18\ldots33]}$	$Z_{[34\ldots49]}$
5	$Z_{[75\ldots90]}$	$Z_{[91\ldots106]}$	$Z_{[107\ldots122]}$	$Z_{[123\ldots10]}$	$Z_{[11\ldots26]}$	$Z_{[27\ldots42]}$
6	$Z_{[43\ldots58]}$	$Z_{[59\ldots74]}$	$Z_{[100\ldots115]}$	$Z_{[116\ldots3]}$	$Z_{[4\ldots19]}$	$Z_{[20\ldots35]}$
7	$Z_{[36\ldots51]}$	$Z_{[52\ldots67]}$	$Z_{[68\ldots83]}$	$Z_{[84\ldots99]}$	$Z_{[125\ldots12]}$	$Z_{[13\ldots28]}$
8	$Z_{[29\ldots44]}$	$Z_{[45\ldots60]}$	$Z_{[61\ldots76]}$	$Z_{[77\ldots92]}$	$Z_{[93\ldots108]}$	$Z_{[109\ldots124]}$
9	$Z_{[22\ldots37]}$	$Z_{[38\ldots53]}$	$Z_{[54\ldots69]}$	$Z_{[70\ldots85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0...15]}$	$Z_{[16...31]}$	$Z_{[32...47]}$	$Z_{[48...63]}$	$Z_{[64...79]}$	$Z_{[80...95]}$
2	$Z_{[96...111]}$	$Z_{[112...127]}$	$Z_{[25...40]}$	$Z_{[41...56]}$	$Z_{[57...72]}$	$Z_{[73...88]}$
3	$Z_{[89...104]}$	$Z_{[105...120]}$	$Z_{[121...8]}$	$Z_{[9...24]}$	$Z_{[50...65]}$	$Z_{[66...81]}$
4	$Z_{[82...97]}$	$Z_{[98...113]}$	$Z_{[114...1]}$	$Z_{[2...17]}$	$Z_{[18...33]}$	$Z_{[34...49]}$
5	$Z_{[75...90]}$	$Z_{[91...106]}$	$Z_{[107...122]}$	$Z_{[123...10]}$	$Z_{[11...26]}$	$Z_{[27...42]}$
6	$Z_{[43...58]}$	$Z_{[59...74]}$	$Z_{[100...115]}$	$Z_{[116...3]}$	$Z_{[4...19]}$	$Z_{[20...35]}$
7	$Z_{[36...51]}$	$Z_{[52...67]}$	$Z_{[68...83]}$	$Z_{[84...99]}$	$Z_{[125...12]}$	$Z_{[13...28]}$
8	$Z_{[29...44]}$	$Z_{[45...60]}$	$Z_{[61...76]}$	$Z_{[77...92]}$	$Z_{[93...108]}$	$Z_{[109...124]}$
9	$Z_{[22...37]}$	$Z_{[38...53]}$	$Z_{[54...69]}$	$Z_{[70...85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0\ldots15]}$	$Z_{[16\ldots31]}$	$Z_{[32\ldots47]}$	$Z_{[48\ldots63]}$	$Z_{[64\ldots79]}$	$Z_{[80\ldots95]}$
2	$Z_{[96\ldots111]}$	$Z_{[112\ldots127]}$	$Z_{[25\ldots40]}$	$Z_{[41\ldots56]}$	$Z_{[57\ldots72]}$	$Z_{[73\ldots88]}$
3	$Z_{[89\ldots104]}$	$Z_{[105\ldots120]}$	$Z_{[121\ldots8]}$	$Z_{[9\ldots24]}$	$Z_{[50\ldots65]}$	$Z_{[66\ldots81]}$
4	$Z_{[82\ldots97]}$	$Z_{[98\ldots113]}$	$Z_{[114\ldots1]}$	$Z_{[2\ldots17]}$	$Z_{[18\ldots33]}$	$Z_{[34\ldots49]}$
5	$Z_{[75\ldots90]}$	$Z_{[91\ldots106]}$	$Z_{[107\ldots122]}$	$Z_{[123\ldots10]}$	$Z_{[11\ldots26]}$	$Z_{[27\ldots42]}$
6	$Z_{[43\ldots58]}$	$Z_{[59\ldots74]}$	$Z_{[100\ldots115]}$	$Z_{[116\ldots3]}$	$Z_{[4\ldots19]}$	$Z_{[20\ldots35]}$
7	$Z_{[36\ldots51]}$	$Z_{[52\ldots67]}$	$Z_{[68\ldots83]}$	$Z_{[84\ldots99]}$	$Z_{[125\ldots12]}$	$Z_{[13\ldots28]}$
8	$Z_{[29\ldots44]}$	$Z_{[45\ldots60]}$	$Z_{[61\ldots76]}$	$Z_{[77\ldots92]}$	$Z_{[93\ldots108]}$	$Z_{[109\ldots124]}$
9	$Z_{[22\ldots37]}$	$Z_{[38\ldots53]}$	$Z_{[54\ldots69]}$	$Z_{[70\ldots85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0...15]}$	$Z_{[16...31]}$	$Z_{[32...47]}$	$Z_{[48...63]}$	$Z_{[64...79]}$	$Z_{[80...95]}$
2	$Z_{[96...111]}$	$Z_{[112...127]}$	$Z_{[25...40]}$	$Z_{[41...56]}$	$Z_{[57...72]}$	$Z_{[73...88]}$
3	$Z_{[89...104]}$	$Z_{[105...120]}$	$Z_{[121...8]}$	$Z_{[9...24]}$	$Z_{[50...65]}$	$Z_{[66...81]}$
4	$Z_{[82...97]}$	$Z_{[98...113]}$	$Z_{[114...1]}$	$Z_{[2...17]}$	$Z_{[18...33]}$	$Z_{[34...49]}$
5	$Z_{[75...90]}$	$Z_{[91...106]}$	$Z_{[107...122]}$	$Z_{[123...10]}$	$Z_{[11...26]}$	$Z_{[27...42]}$
6	$Z_{[43...58]}$	$Z_{[59...74]}$	$Z_{[100...115]}$	$Z_{[116...3]}$	$Z_{[4...19]}$	$Z_{[20...35]}$
7	$Z_{[36...51]}$	$Z_{[52...67]}$	$Z_{[68...83]}$	$Z_{[84...99]}$	$Z_{[125...12]}$	$Z_{[13...28]}$
8	$Z_{[29...44]}$	$Z_{[45...60]}$	$Z_{[61...76]}$	$Z_{[77...92]}$	$Z_{[93...108]}$	$Z_{[109...124]}$
9	$Z_{[22...37]}$	$Z_{[38...53]}$	$Z_{[54...69]}$	$Z_{[70...85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0\ldots15]}$	$Z_{[16\ldots31]}$	$Z_{[32\ldots47]}$	$Z_{[48\ldots63]}$	$Z_{[64\ldots79]}$	$Z_{[80\ldots95]}$
2	$Z_{[96\ldots111]}$	$Z_{[112\ldots127]}$	$Z_{[25\ldots40]}$	$Z_{[41\ldots56]}$	$Z_{[57\ldots72]}$	$Z_{[73\ldots88]}$
3	$Z_{[89\ldots104]}$	$Z_{[105\ldots120]}$	$Z_{[121\ldots8]}$	$Z_{[9\ldots24]}$	$Z_{[50\ldots65]}$	$Z_{[66\ldots81]}$
4	$Z_{[82\ldots97]}$	$Z_{[98\ldots113]}$	$Z_{[114\ldots1]}$	$Z_{[2\ldots17]}$	$Z_{[18\ldots33]}$	$Z_{[34\ldots49]}$
5	$Z_{[75\ldots90]}$	$Z_{[91\ldots106]}$	$Z_{[107\ldots122]}$	$Z_{[123\ldots10]}$	$Z_{[11\ldots26]}$	$Z_{[27\ldots42]}$
6	$Z_{[43\ldots58]}$	$Z_{[59\ldots74]}$	$Z_{[100\ldots115]}$	$Z_{[116\ldots3]}$	$Z_{[4\ldots19]}$	$Z_{[20\ldots35]}$
7	$Z_{[36\ldots51]}$	$Z_{[52\ldots67]}$	$Z_{[68\ldots83]}$	$Z_{[84\ldots99]}$	$Z_{[125\ldots12]}$	$Z_{[13\ldots28]}$
8	$Z_{[29\ldots44]}$	$Z_{[45\ldots60]}$	$Z_{[61\ldots76]}$	$Z_{[77\ldots92]}$	$Z_{[93\ldots108]}$	$Z_{[109\ldots124]}$
9	$Z_{[22\ldots37]}$	$Z_{[38\ldots53]}$	$Z_{[54\ldots69]}$	$Z_{[70\ldots85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0\ldots15]}$	$Z_{[16\ldots31]}$	$Z_{[32\ldots47]}$	$Z_{[48\ldots63]}$	$Z_{[64\ldots79]}$	$Z_{[80\ldots95]}$
2	$Z_{[96\ldots111]}$	$Z_{[112\ldots127]}$	$Z_{[25\ldots40]}$	$Z_{[41\ldots56]}$	$Z_{[57\ldots72]}$	$Z_{[73\ldots88]}$
3	$Z_{[89\ldots104]}$	$Z_{[105\ldots120]}$	$Z_{[121\ldots8]}$	$Z_{[9\ldots24]}$	$Z_{[50\ldots65]}$	$Z_{[66\ldots81]}$
4	$Z_{[82\ldots97]}$	$Z_{[98\ldots113]}$	$Z_{[114\ldots1]}$	$Z_{[2\ldots17]}$	$Z_{[18\ldots33]}$	$Z_{[34\ldots49]}$
5	$Z_{[75\ldots90]}$	$Z_{[91\ldots106]}$	$Z_{[107\ldots122]}$	$Z_{[123\ldots10]}$	$Z_{[11\ldots26]}$	$Z_{[27\ldots42]}$
6	$Z_{[43\ldots58]}$	$Z_{[59\ldots74]}$	$Z_{[100\ldots115]}$	$Z_{[116\ldots3]}$	$Z_{[4\ldots19]}$	$Z_{[20\ldots35]}$
7	$Z_{[36\ldots51]}$	$Z_{[52\ldots67]}$	$Z_{[68\ldots83]}$	$Z_{[84\ldots99]}$	$Z_{[125\ldots12]}$	$Z_{[13\ldots28]}$
8	$Z_{[29\ldots44]}$	$Z_{[45\ldots60]}$	$Z_{[61\ldots76]}$	$Z_{[77\ldots92]}$	$Z_{[93\ldots108]}$	$Z_{[109\ldots124]}$
9	$Z_{[22\ldots37]}$	$Z_{[38\ldots53]}$	$Z_{[54\ldots69]}$	$Z_{[70\ldots85]}$		

IDEA // Key-Schedule Algorithm

Round i	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	$Z_{[0...15]}$	$Z_{[16...31]}$	$Z_{[32...47]}$	$Z_{[48...63]}$	$Z_{[64...79]}$	$Z_{[80...95]}$
2	$Z_{[96...111]}$	$Z_{[112...127]}$	$Z_{[25...40]}$	$Z_{[41...56]}$	$Z_{[57...72]}$	$Z_{[73...88]}$
3	$Z_{[89...104]}$	$Z_{[105...120]}$	$Z_{[121...8]}$	$Z_{[9...24]}$	$Z_{[50...65]}$	$Z_{[66...81]}$
4	$Z_{[82...97]}$	$Z_{[98...113]}$	$Z_{[114...1]}$	$Z_{[2...17]}$	$Z_{[18...33]}$	$Z_{[34...49]}$
5	$Z_{[75...90]}$	$Z_{[91...106]}$	$Z_{[107...122]}$	$Z_{[123...10]}$	$Z_{[11...26]}$	$Z_{[27...42]}$
6	$Z_{[43...58]}$	$Z_{[59...74]}$	$Z_{[100...115]}$	$Z_{[116...3]}$	$Z_{[4...19]}$	$Z_{[20...35]}$
7	$Z_{[36...51]}$	$Z_{[52...67]}$	$Z_{[68...83]}$	$Z_{[84...99]}$	$Z_{[125...12]}$	$Z_{[13...28]}$
8	$Z_{[29...44]}$	$Z_{[45...60]}$	$Z_{[61...76]}$	$Z_{[77...92]}$	$Z_{[93...108]}$	$Z_{[109...124]}$
9	$Z_{[22...37]}$	$Z_{[38...53]}$	$Z_{[54...69]}$	$Z_{[70...85]}$		

IDEA // Philosophy Recycling

> On May 16th, 2011, IDEA will fall into the public domain
> The «IDEA way» to build a cipher looks like to be valid in terms of security
> Existing derivates (like MESH ciphers) are not very competitive in terms of speed
> Can we recycle this approach to design something new and fast, with a look at hash functions and authenticated encryption ?

IDEA // A Fast Implementation

> Implementation of 8-way IDEA on the `x86_64` architecture using the SSE2 instruction set `[JunMac09]` running at 5.4 clocks/byte on an Intel Core2 CPU

> Motivated the design of WIDEA-8

  > Block cipher with 512-bit block size, 1024-bit key size

  > Fully respect the IDEA philosophy

  > New key-schedule

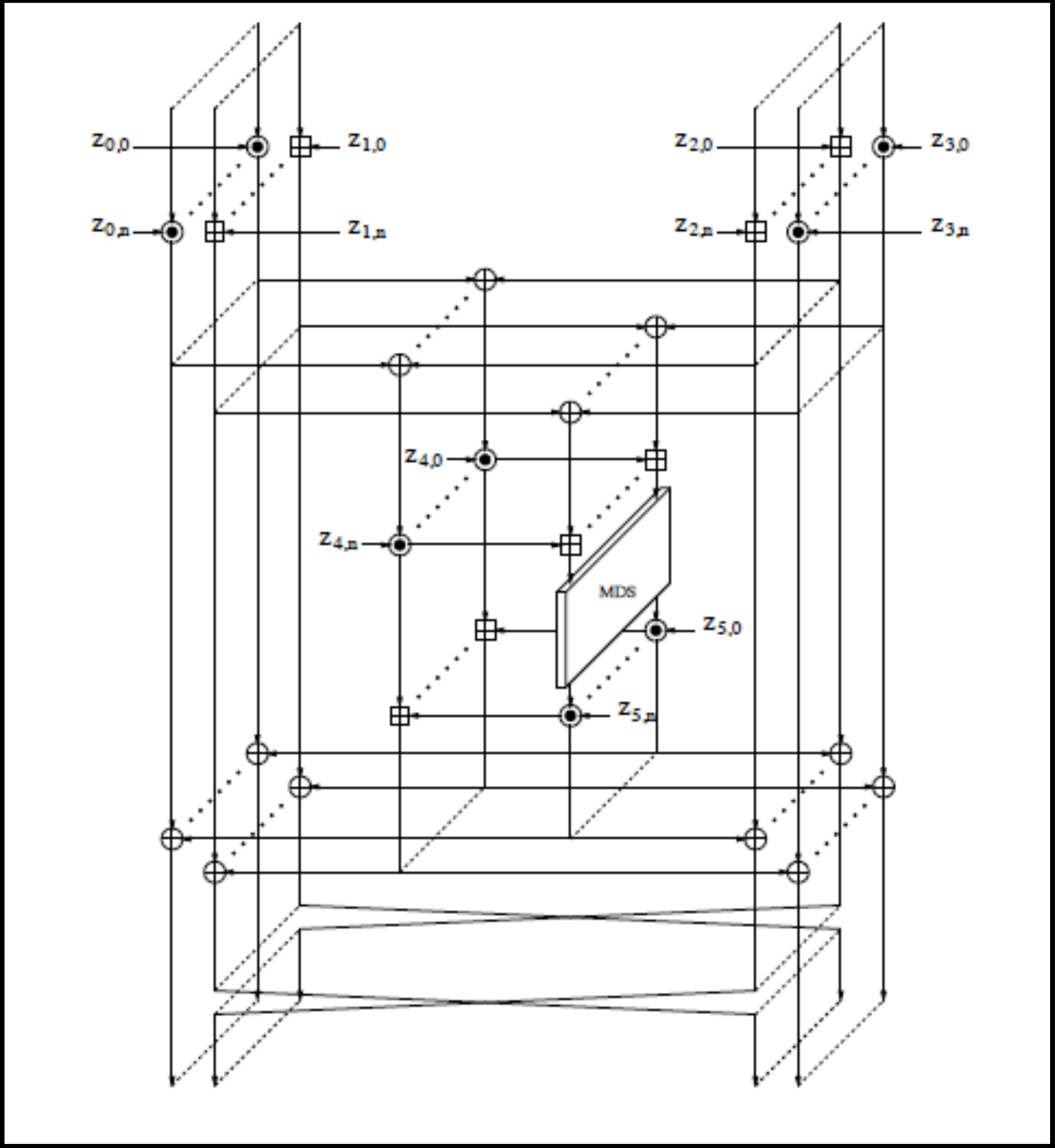
  > Keep highest possible parallelism

  > Inherit all the good security properties of IDEA

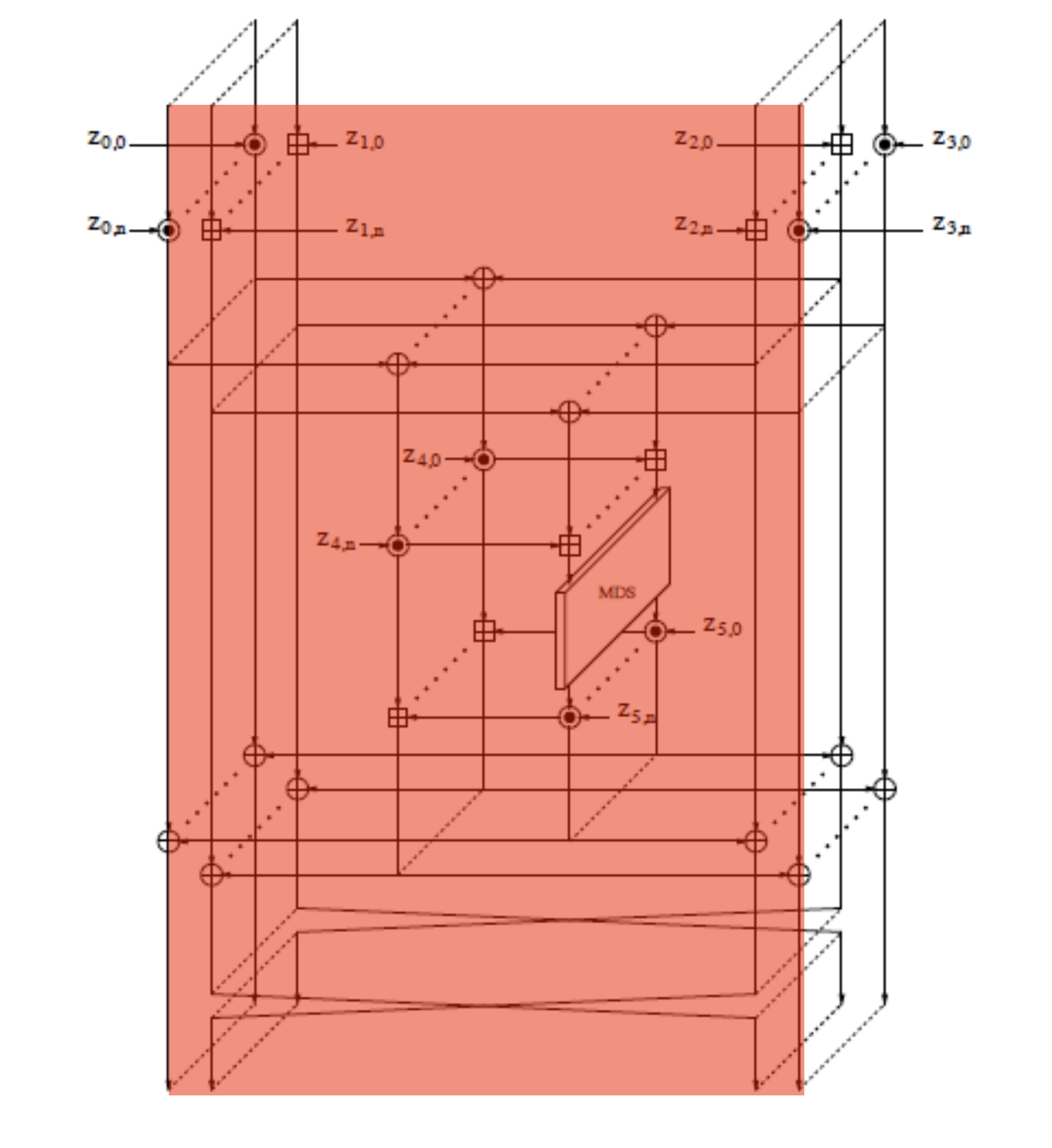
Outline

> IDEA
> WIDEA
> HIDEA

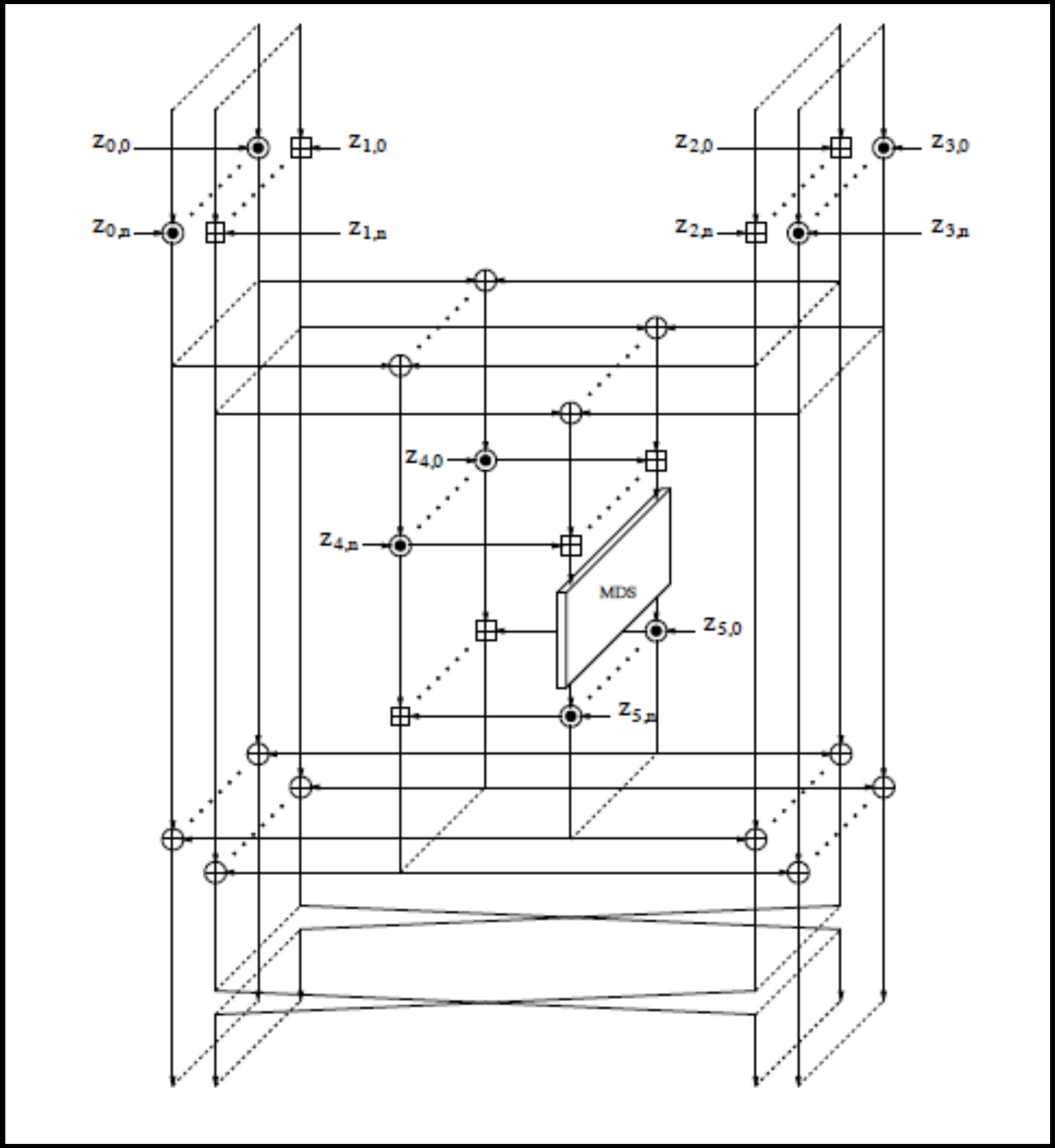
WIDEA // As a Single Picture



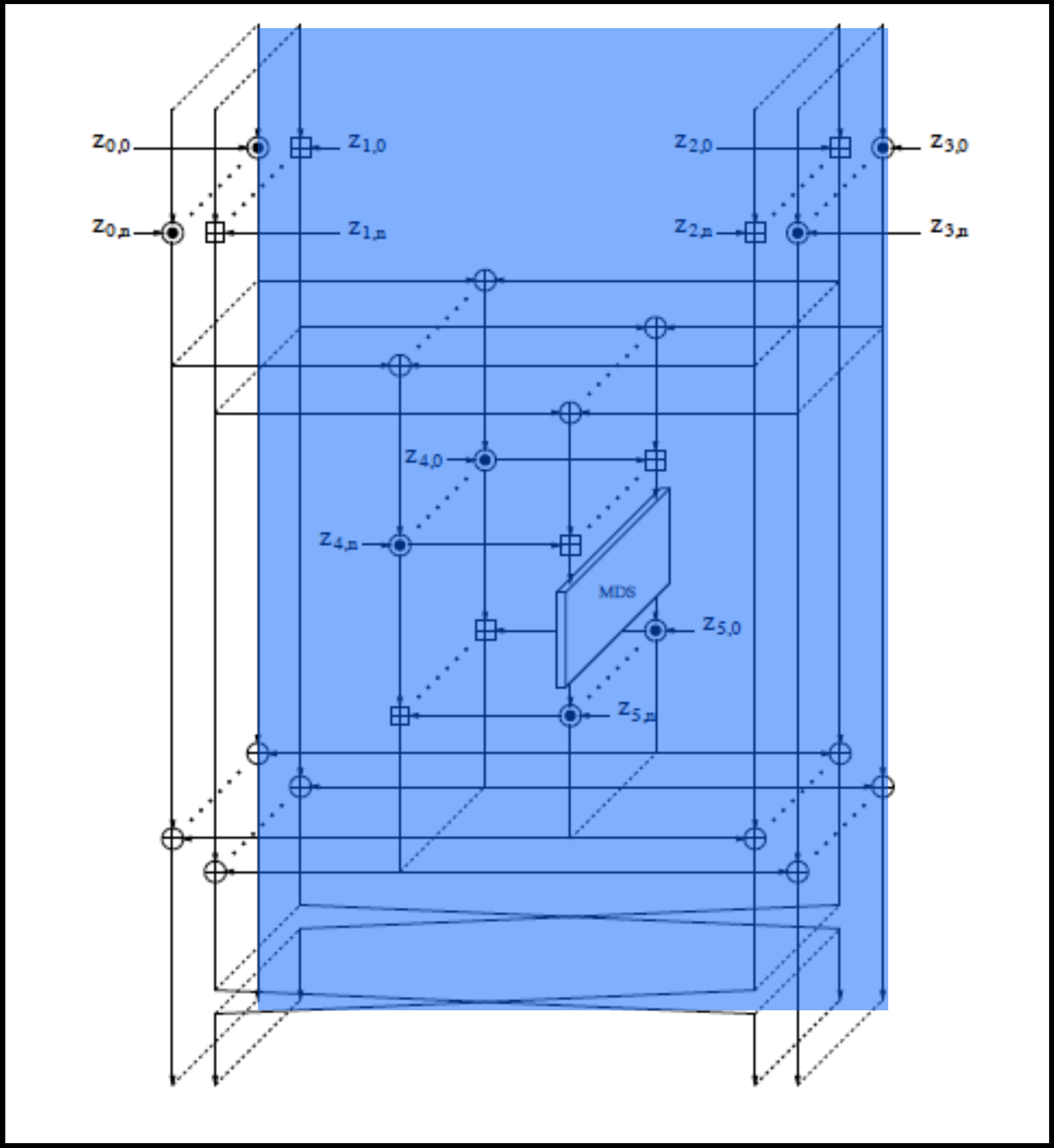
WIDEA // As a Single Picture



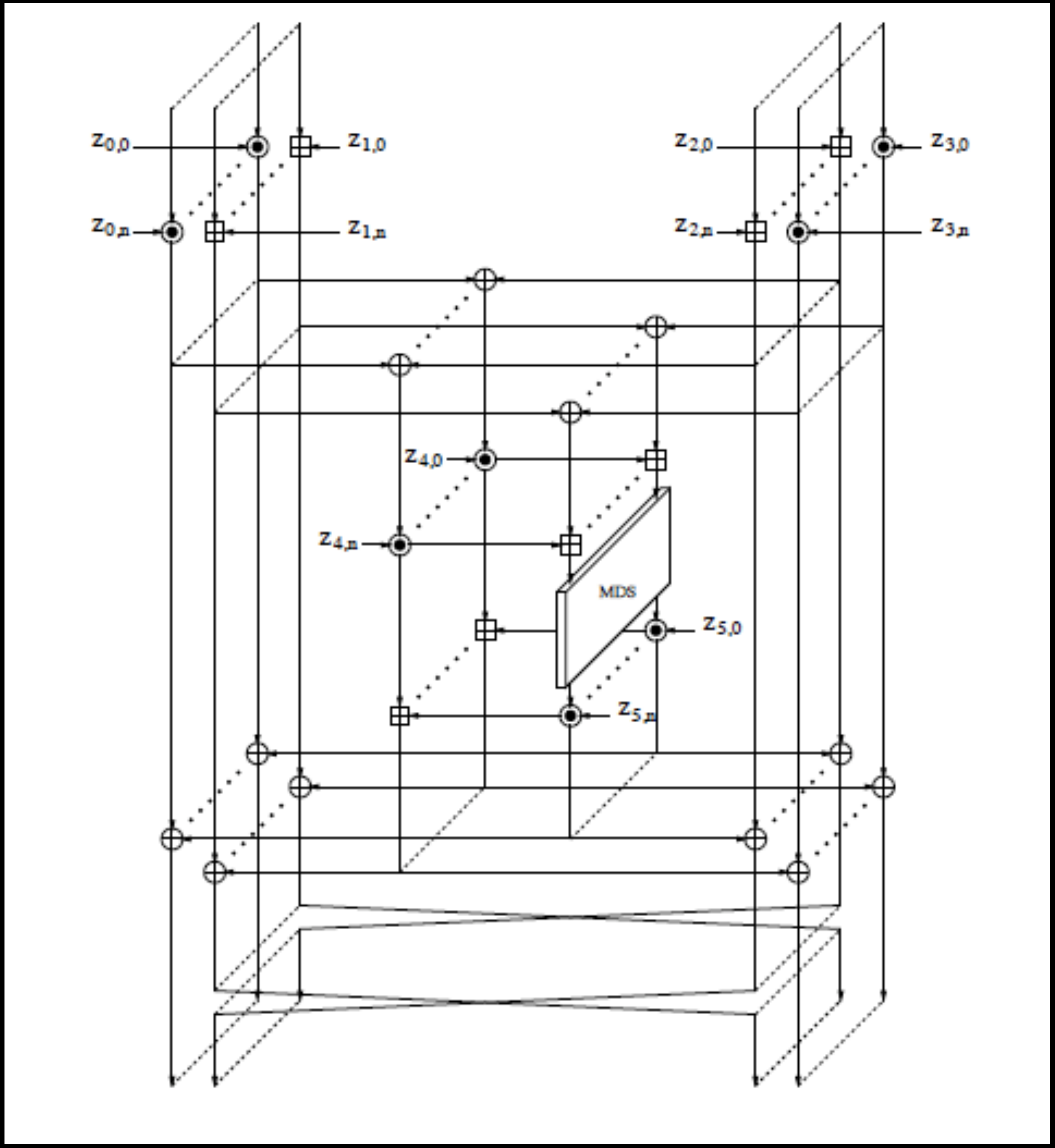
WIDEA // As a Single Picture



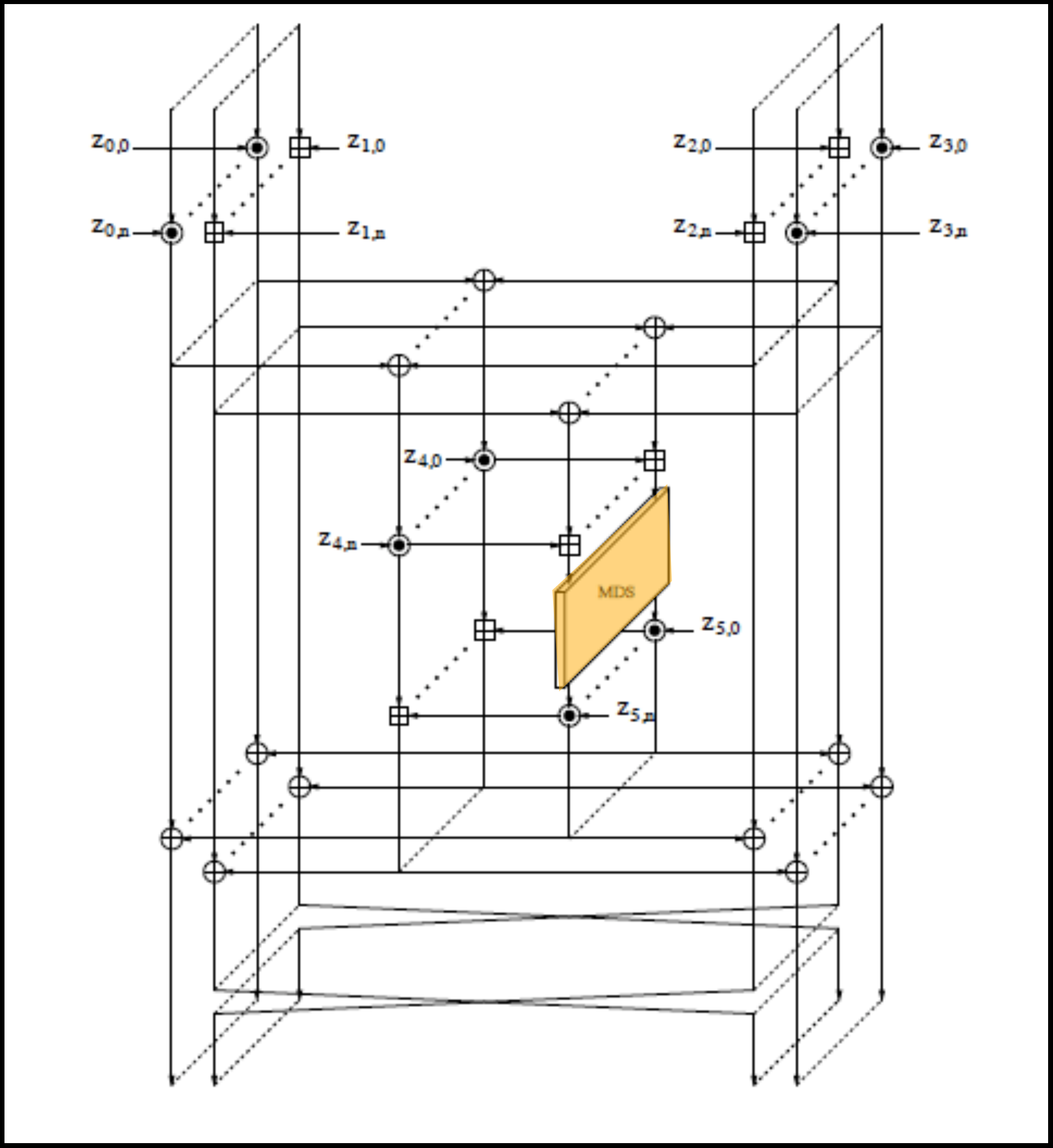
WIDEA // As a Single Picture



WIDEA // As a Single Picture



WIDEA // As a Single Picture



WIDEA // Optimal Cross-Diffusion

> Diffusion across the 8 instances through a $\mathrm{GF}(2)$-linear $(8,8)$-multipermutation over $\mathrm{GF}(2^{16})$

> Only «sequential» step in the whole cipher

> But it is still possible to perform some of the operations in parallel

WIDEA // Key-Schedule Algorithm

> Non-linear feedback shift register
> Fast diffusion (full diffusion after 3 rounds of WIDEA)
> Asymmetry brought through iteration-dependent constants
> Design similar to the Rijndael key-schedule algorithm

$$Z_i = K_i \qquad 0 \leq i \leq 7$$

$$Z_i = ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) \oplus C_{\frac{i}{8}-1} \qquad 8 \leq i \leq 51,\ 8 \mid i$$

$$Z_i = ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) \qquad 8 \leq i \leq 51,\ 8 \nmid i$$

WIDEA // Security Considerations

> Two sequential operations are always algebraically incompatible
> Thanks to the MDS matrix, we get full diffusion after a single round
    > Total of eight full diffusion
    > More than most existing designs
> Differential, linear and integral properties expected to behave the same way than for IDEA
> The new non-linear key-schedule further strengthen the design

WIDEA // Implementation

> WIDEA-8 is fully specified in the FSE'09 paper
> Implemented as a compression function
  > Davies-Meyer mode
  > Merkle-Damgard scheme
  > SSE3 instruction set on an Intel Core 2
  > 5.98 clocks / byte
> Fill the gap from the compression function to a full-flavored hash function
  > HIDEA

Outline

> IDEA
> WIDEA
> HIDEA

HIDEA // Introduction

> FSE'10 anonymous reviewer comment

  `However, given where we are with SHA-3, the authors should provide better justification for why we need yet another hash function proposal.`

> Main goals

  > Recycle the «IDEA philosophy»

  > Get a new toy to play with

  > Propose a somewhat alternative design to the many-iterations-of-a-light-function approach

HIDEA // Basics

- HIDEA (= «Hash» based on IDEA)
- Design relies on Biham and Dunkelman's HAIFA framework
  - Two instances
    - HIDEA-256
      - 256-bit digest
      - 128-bit salt
      - 64-bit counter
      - 10.5-round WIDEA-4 as compression function
    - HIDEA-512
      - 512-bit digest
      - 256-bit salt
      - 128-bit counter
      - 10.5-round WIDEA-8 as compression function

HIDEA // Counter Inclusion

> Cross-diffusion involves also the counter value:

$$Y = \mathrm{MDS}(X) = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} \\ \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} \\ \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} \\ \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} \\ \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} \\ \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x4} \\ \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} & \mathtt{0x1} \\ \mathtt{0x1} & \mathtt{0x4} & \mathtt{0x1} & \mathtt{0x8} & \mathtt{0x5} & \mathtt{0x2} & \mathtt{0x9} & \mathtt{0x1} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \oplus C$$

$$Y = \mathrm{MDS}(X) = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \mathtt{0x2} & \mathtt{0x3} & \mathtt{0x1} & \mathtt{0x1} \\ \mathtt{0x1} & \mathtt{0x2} & \mathtt{0x3} & \mathtt{0x1} \\ \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x2} & \mathtt{0x3} \\ \mathtt{0x3} & \mathtt{0x1} & \mathtt{0x1} & \mathtt{0x2} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \oplus C$$

HIDEA // Seed Inclusion

> Seed is handled as a round subkey and included using the $\boxplus$ operation

$$
\begin{aligned}
Z_{6i} &= K_{5(j-1)} \\
Z_{6i+1} &= S_0 \text{ and } Z_{6i+2} = K_{5(j-1)+1} \text{ if } j \equiv 1 \pmod 2 \\
Z_{6i+1} &= K_{5(j-1)+1} \text{ and } Z_{6i+2} = S_1 \text{ if } j \equiv 0 \pmod 2 \\
Z_{6i+3} &= K_{5(j-1)+2} \\
Z_{6i+4} &= K_{5(j-1)+3} \\
Z_{6i+5} &= K_{5(j-1)+4}.
\end{aligned}
$$

HIDEA // Number of Rounds

> Compared to WIDEA which uses 8.5 rounds, we added two additional rounds

> > Invest two full rounds to inject the seed
> >
> > We still keep rather good performances on high-end CPUs (Xeon CPU, end-of-2009 eBASH numbers)

Hash function	Speed (clock cycles / byte)
BMW-512	4.75
Skein-512	6.63
CubeHash8/32	6.70
HIDEA-512	**7.65**
Shabal-512	8.31
BMW-256	9.32
BLAKE-32	10.39
SIMD-256	12.87
Keccak	13.55
SHAvite-3	27.82
Grostl-512	30.11
Fugue-256	26.41
Hamsi	31.28
ECHO-256	36.35

HIDEA // Number of Rounds

> Compared to WIDEA which uses 8.5 rounds, we added two additional rounds

> > Invest two full rounds to inject the seed
>
> > We still keep rather good performances on high-end CPUs (Xeon CPU, end-of-2009 eBASH numbers)

Hash function	Speed (clock cycles / byte)
BMW-512	4.75
Skein-512	6.63
CubeHash8/32	6.70
HIDEA-512	**7.65**
Shabal-512	8.31
BMW-256	9.32
BLAKE-32	10.39
SIMD-256	12.87
Keccak	13.55
SHAvite-3	27.82
Grostl-512	30.11
Fugue-256	26.41
Hamsi	31.28
ECHO-256	36.35

HIDEA // ATMega 8-bit Microcontroller

> First (rather unoptimized) implementation of HIDEA-256 on ATMega128

- > Code segment size: less than 2 kB
- > SRAM usage: 138 bytes
- > Throughput about 270 cycles / byte

> Still a bit difficult to compare those numbers with SHA-3 due to the lack of available literature.

HIDEA // Concluding Remarks and Open Questions

> Design still very preliminary
> Work in progress
  > Almost no security analysis
  > Resistance to collision attacks still to be assessed
  > Interaction between IDEA multiplication weak values and the key-schedule have still to be seriously assessed

THANK YOU !