r5embed numbers 20200410210300

Changelog	
2020-01-31	Markku-Juhani O. Saarinen <mjos@pqshield.com>
	First measurements for new parameters.
2020-04-10	*TupleHash, optimizations.*

1 Round5 on Embedded, Mobile, and IoT

A separate source distribution of Round5 without external library dependencies is available from: `https://github.com/r5embed/r5embed`.

This codebase is smaller and more portable than the NIST reference code and has been tested on various architectures (32/64 bit targets, big-endian, etc). It forms the basis for our embedded testing and hardware-software codesign (although those hardware drivers are not included in the above distribution).

1.1 IoT Embedded Performance (Cortex M4)

We used the ST STM32F407 Discovery board and the "r5embed" code base for our for Cortex M4 performance and size measurements. This implementation includes some ARMv7 assembler code for speeding up core ring and matrix arithmetic for this ARM Cortex M4 (ARMv7E-M) reference target.

This particular MCU (and Cortex M4 in general) does not deploy a data cache – a separate flash cache exists on the instruction bus. Code execution paths are therefore designed to be constant, while there is some variation in the data access paths. This implementation may be considered *hardened* or *resistant* against a time-channel attack, while not strictly constant time.

This updated implementation will be made available to the PQM4 joint Cortex M4 benchmarking effort[1]. We note that during the second round of the NIST competition Round5 had overall leading embedded performance according to PQM4 data, although the margin is not particularly large. Compared to some other candidates, the "r5embed" implementation is much smaller and more of it is written in a high-level language, as it was written for portability.

Table 1 summarizes performance on the Cortex M4 target. Note that the `R5N1_3CCA_0smallCT` target has been excluded due to its relatively large RAM requirement. Table 2 summarizes the RAM, ROM (Flash), and bandwidth requirements of the ARMv7 implementation.

1.2 Hardware-Software Codesign

The lattice and ring arithmetic coprocessor in PQSoC[2] can support Round5 directly. PQSoC is a self-contained SoC platform with a RISC-V Core (RV32IMC "Pluto") and other peripherals such as a very fast SHA-3 Keccak core.

We synthesized PQSoC with and without the ring arithmetic accelerator on a Xilinx Artix-7 platform (Digilent Arty A7-35T board). Table 3 indicates its relative size, which is very small. Even with the CPU the implementation

[1] PQM4: Cortex-M4 post-quantum cryptography library: `https://github.com/mupq/pqm4`.
[2] PQSoC: Post-Quantum Crypto IP Data Sheet `https://pqsoc.com`

Variant	KG	Enc	Dec	Tot
R5ND_1CPA_5d	425	607	247	1,280
R5ND_3CPA_5d	839	1,136	400	2,376
R5ND_5CPA_5d	1,455	1,976	697	4,129
R5ND_1CCA_5d	384	593	755	1,734
R5ND_3CCA_5d	840	1,253	1,567	3,661
R5ND_5CCA_5d	1,386	2,011	2,563	5,961
R5N1_1CPA_0d	6,625	4,488	1,209	12,323
R5N1_3CPA_0d	9,935	7,063	1,849	18,849
R5N1_5CPA_0d	34,833	20,891	4,455	60,181
R5N1_1CCA_0d	4,166	3,723	4,051	11,942
R5N1_3CCA_0d	17,094	11,986	13,572	42,654
R5N1_5CCA_0d	23,206	16,681	17,949	57,837
R5ND_0CPA_2iot	393	517	195	1,106
R5ND_1CPA_4longkey	423	628	271	1,324

Table 1: Round5 performance on ARM Cortex M4 (STM32F407 Discovery) clocked at 24 Mhz. All of these numbers are in 1000s of cycles; KG = keypair generation, Enc = encapsulation, Dec = decapsulation, Tot = KG+Enc+Dec measured as a whole (both sides of an ephemeral key exchange).

	RAM (Stack) Usage in			Bandwidth for		ROM
Variant	**KG**	**Enc**	**Dec**	**PK**	**CT**	**Code**
R5ND_1CPA_5d	3,862	4,317	2,124	445	549	5,894
R5ND_3CPA_5d	5,598	6,173	2,708	780	859	7,332
R5ND_5CPA_5d	7,038	7,717	3,468	972	1,063	4,828
R5ND_1CCA_5d	3,958	4,405	5,028	461	620	6,248
R5ND_3CCA_5d	5,638	6,229	7,164	780	934	7,788
R5ND_5CCA_5d	7,078	7,781	9,068	978	1,285	5,412
R5N1_1CPA_0d	19,286	19,501	12,996	5,214	5,236	3,796
R5N1_3CPA_0d	26,950	27,221	19,556	8,834	8,866	3,804
R5N1_5CPA_0d	40,934	41,213	32,324	14,264	14,288	3,910
R5N1_1CCA_0d	19,878	20,109	25,900	5,740	5,788	3,892
R5N1_3CCA_0d	30,206	30,485	40,204	9,660	9,716	4,228
R5N1_5CCA_0d	37,638	37,933	52,644	14,636	14,708	4,164
R5ND_0CPA_2iot	3,198	3,501	1,740	342	394	3,618
R5ND_1CPA_4longkey	3,830	4,357	2,180	453	563	5,764

Table 2: Round5 RAM (Stack) / ROM (Flash) and bandwidth usage on Cortex M4 (or any ARMv7). All numbers are in bytes: KG = keypair generation RAM, Enc = encapsulation RAM, Dec = decapsulation RAM, PK = public key (transmit), CT ciphertext (transmit), Code = firmware size excluding Keccak and other standard components (ROM or Flash).

Artix-7 FPGA	Coprocessor?		Area
Component	**No**	**Yes**	**Delta**
SLICEL	1,525	1,533	+8
SLICEM	626	691	+64
LUT as Logic	7,064	7,720	+656
LUT as Memory	48	48	+0
Slice Registers	2,626	3,336	+710
DSP48E1	4	5	+1

Table 3: Synthesis of PQSoC with and without the Round5 lattice coprocessor on a Xilinx Artix-7 (XC7A35) FPGA platform (Arty 7), indicating its size.

	With coprocessor			Plain C Code			Speed
Variant	**KG**	**Enc**	**Dec**	**KG**	**Enc**	**Dec**	**Factor**
`R5ND_1CPA_5d`	137	195	89	888	1,578	724	7.59
`R5ND_3CPA_5d`	205	273	105	1,937	3,222	1,329	11.12
`R5ND_5CPA_5d`	346	450	164	4,001	6,595	2,663	13.82
`R5ND_1CCA_5d`	130	195	276	789	1,387	1,997	6.94
`R5ND_3CCA_5d`	207	290	401	1,939	3,248	4,575	10.86
`R5ND_5CCA_5d`	338	458	627	3,794	6,257	8,758	13.22
`R5ND_0CPA_2iot`	127	166	73	761	1,193	473	6.64
`R5ND_1CPA_4longkey`	132	196	91	884	1,653	801	7.96

Table 4: Performance of Round5 on a lightweight RISC-V SoC with the lattice coprocessor and without. The numbers are in 1000s of cycles at 100 MHz (equivalent to 10 μs). Neither of the implementations uses assembly language. The hardware accelerator can be used with other CPU architectures as well.

is smaller than some proposed hardware modules; as a codesign it is minuscule. From Table 4 we see that the performance advantage gained from the coprocessor is between 664% and 1382%, depending on the variant.

Four DSP units are used by the multiplication unit of the RV32 CPU core, and 1 by the the lattice unit. However, that is actually needed when executing Round5 but are required by an another PQC algorithms. The ring multiplication in Round5 is constructed entirely from additions and subtractions.

The design achieves timing closure at 100 MHz – that was also the clock frequency used in timing runs. As can be seen, depending on (ring variant) security level key generation requires $1.3 - 3.4$ ms, encapsulation $1.7 - 4.5$ ms, and decapsulation $0.9 - 6.2$ ms. What is noteworthy in comparison to monolithic hardware implementations is that all of these options are available *simultaneously* via the same hardware module with very small additional cost.