Simulation of ARM and x86 Microprocessors Using In-order and Out-of-order CPU Models with Gem5 Simulator

Anas Ahmad Abudaqa, Talal M. Al-Kharoubi, Muhamed F. Mudawar, Armin Kobilica
Dept. of Computer Engineering
King Fahd University for Petroleum and Minerals
Dhahran, KSA
e-mail: {g201202060, talalkh, mudawar, g201403920}@kfupm.edu.sa

***Abstract*—In the past, ARM and Intel x86 computer processors were leaders in the different specialized areas with no competing interest. While Intel was building processors for the PCs and servers, ARM was concentrated on the handheld devices. Nowadays, with no hard limits in computing power between the different sizes of devices, ARM and Intel-based microprocessors are competitors. In this paper, a comparative study is done to evaluate the performance of the two microprocessors. Both, the in-order and the out-of-order, CPU models are used within each architecture, and four performance metrics (total consumed energy, throughput, average cycles per instruction (CPI), and L2 cache miss rate) are used to evaluate the work. The well-known computer architecture simulator, Gem5, is used to accomplish the work. Results show that ARM microprocessor outperforms x86 microprocessor in the most cases.**

Keywords-ARM; x86; Gem5; CPI; O3, in-order

I. INTRODUCTION

Computers are computational machines executing different sets of instructions. Different architectures of these computational machines requires different sets of instructions. For example, the binary code assigned to the opcode field varies based on its different implementations in different processors. Likewise, although a standard exists, the symbolic name given to instructions varies for different processors 0. Next to these minor differences, there are two major types of instruction set architectures that differ markedly in the relationship of hardware to software: complex instruction set computers (CISCs)[2], and reduced instruction set computers (RISCs) [2]. CISCs (Intel x86) provide hardware support for high-level language operations and have compact programs, whereas, RISCs (ARM [1]) emphasize simple instructions and flexibility [3].

The main objective of RISC architecture is to increase throughput and to have faster execution. This is addressed by, not considering fetching instructions, complete avoidance of accesses to memory. On the other hand, the goal of the CISC architecture is to match more closely the operations used in programming languages and to provide instructions that facilitate compact programs and conserve memory.

Since the invention of the RISC-based computer architecture, the performance bar has been raised, forcing prior architectures to compete by adaption or to risk outdated performance and disappearance from processors' market.

To adapt to new challenges and adaption to RISC-like architectures, Intel extends 80x86 processor by translating its instructions to RISC-like internally allowing it to adopt many of the innovations first pioneered in the RISC designs [4]. Table I shows a comparison between ARM and x86 ISAs.

To analyze the performance of these two architectures, a comparative study is conducted by simulating a group of benchmarks applications, i.e., Mibench [5] on both ARM and Intel x86 architectures. Both, in-order and out-of-order, CPU models are used with each architecture. The well-known computer architecture simulator, Gem5 [6], is used to accomplish the work.

The contribution of this paper is twofold: (1) We fixed the bugs of Mibench *.c* files, and we have successfully compiled these files for both ARM and x86 architectures, these fixed files could be requested by email[2], hence they ready to be used easily within Gem5 simulator. Moreover, by default, the in-order CPU model is not configured for the x86 processor; therefore, we do our configuration to suppress this problem. (2) A well-chosen four performance metrics are used to evaluate the architectures; these metrics are total consumed energy (mJ), throughput (bytes/s), average cycles per instruction (CPI), and L2 cache miss rate.

TABLE I. ARM ARCHITECTURE VS. X86 ARCHITECTURE

	ARM	**X86**
Instruction Size	Sizes of instructions are unified and execution is done within single clock cycle.	Sizes of instructions vary based on different complexities.
Fetching and Decoding	The process of fetching instructions is much simpler. Additionally, access of opcodes and operands is possible in the simultaneous manner.	The process of fetching is complex including different formats and several addressing modes.
Power Consumption	Design of control unit is simplified with better power efficiency.	Due to complex design, control unit has higher level of power consumption.
Program Size	Specific function requires many simple instructions creating programs larger in size.	Similar function demands less number of instructions adding more to space efficiency.

[1] Abbreviated to Advanced Risk Machines.

[2] g201202060@kfupm.edu.sa

The rest of this paper is organized as follows: section II gives a detailed background about the Gem5 simulator and Mibench. In section II we discuss the related work. Section IV shows the configuration and experimental work, meanwhile, results and analysis are presented in section V. Conclusion and future work are presented in section VI.

II. BACKGROUND

A. Gem5 Simulator

The Gem5 simulator is a modular platform used in research community for measuring performance and analysis of computer architecture. Gem5 originated as a merge of two well-known projects: M5[7] and Gems [8]. M5 was known for CPU simulation while Gem was focusing on memory system.

Gem5 supports multiple architectures like ALPHA, ARM, SPARC, MIPS, x86 and POWER, and has two modes of operation: system emulation (SE) and full-system (FS) emulation; the former acts as a functional simulator with a console interface as a mean of communication to the user, the latter allows user interaction with the simulated operating system.

One of the main characteristics of Gem5 its Ability to simulate a complete system, including peripherals and network connections. Further, Gem5 has two different approaches to the memory system simulation. While classic approach focuses on the pipeline simulation, ruby approach supports configurable cache coherence protocols providing more detailed memory hierarchy simulation. Table II shows a comparison of Gem5 among different simulators. The full detailed comparison can be found in [9].

TABLE II. COMPUTER ARCHITECTURE SIMULATORS FEATURES

	Documentation	**Popularity in Research**	**Accuracy Results**	**Full System Simulation**	**License**
CACTI [10]	-	-	-	N/A	private
Multi2-Sim [11]	+	+	+	+	Free
Gem5 [6]	+	+	+	+	Free
QEMU [12]	+	-	-	-	Free
Ruby [13]	-	+	+	N/A	Free

B. Mibench Benchmark

Mibench, pronounced as "My Bench", is a set of thirty-five applications divided into six categories. Their source code is publicly available for free. The applications that are used in this paper and their details are shown in Table III.

TABLE III. THE MIBENCH BENCHMARKS

	Description	**# of Instructions**	**Category**
Basicmath	Performs simple math calculation that usually doesn't have dedicated hardware support in embedded processor.	281591461	Automotive
Dijkstra	An algorithm which finds the smallest path between every pair of network nodes.	45088085	Networking
Qsort	A well-known algorithm which sorts elements of a large array into ascending order using.	15284694	
Patricia	Specially designed data structure to reduce time complexity in full trees with high sparsity in the leaf nodes.	6012	Networking
Bitcount	Tests how the processor is able to count the number of bits in an array of integers	5045	Automotive

III. RELATED WORK

R. Aroca et al. [14] conduct a comparison of server applications running on x86 and ARM architectures. The performance metrics are power usage, CPU load, temperature, and request latencies. They analyzed and compared several ARM and x86 devices on web server, database servers and floating point computation. For tests that are done on HTTP and SQL servers, ARM device are 3 to 4 time less energy consumption than x86. However, x86 is superior in floating point computation.

F. Endo et al. [15] used Gem5 simulator to simulate in-order and out-of-order ARM microprocessors. Real hardware was used for comparison of execution time of ten benchmark applications. On average, these models estimate the execution time with absolute errors around 7% to 17%. Other simulators than Gem5, show average absolute errors between 15 and 37%.

H. Rawat et al. [16] use the Gem5 simulator to evaluate their proposed crypto-instruction set for the Keccak sponge function (Keccak-f) and for the Keccak duplex construction (Keccak-p). A set of six new instructions are proposed based on NEON instruction set in ARMv7 ISA, and Intel AVX. They show performance improvements in instructions committed per byte of input data, and provide hardware cost of the proposed extensions.

IV. CONFIGURATION AND EXPERIMENTAL WORK

To evaluate the performance of the x86 and ARM, the following configuration characteristics are used within Gem5 simulator: First we enable and activate the in-order CPU model for the x86 architecture. Information about how to enable this configuration can be found in Appendix A. Both, in-order and out-of-order, CPU models are used for each architecture. For each cache level (L1 and L2) two configurations are used for every CPU model.

All of the five benchmarks are simulated four times to cover all the configuration combinations, and the four performance metrics, total energy (mJ), throughput (kB/s), average cycles per instruction (CPI), and L2 cache miss rate, are calculated. The System Emulation (SE) operation mode is used. TABLE **IV** summarizes the configuration parameters that are used to complete the simulation process.

TABLE IV. CONFIGURATION PARAMETERS

Parameter	Value
CPU Model	in-order, out-of-order
Core clock Frequency	1GHz
L2 cache size	512KB, 1MB
L2 Associativity	8
L1-I cache size	32kB, 64kB
L1-D cache size	32kB, 64kB
L1 Associativity (both caches)	2
Theoretical Peak memory bandwidth	12800 MiB/s

V. RESULTS AND ANALYSIS

In this section, we show the corresponding results. The following subsections present the performance metrics and their results and analysis.

A. Average Cycles per Instruction (CPI)

For the CPI test, the results came as expected. For all benchmarks the out-of-order CPU model has less CPI than in- order model. Also, ARM always has less CPI than x86. However, the difference between the two architectures when the CPU model is out-of-order is very small. CPI results are depicted in Fig. 1.

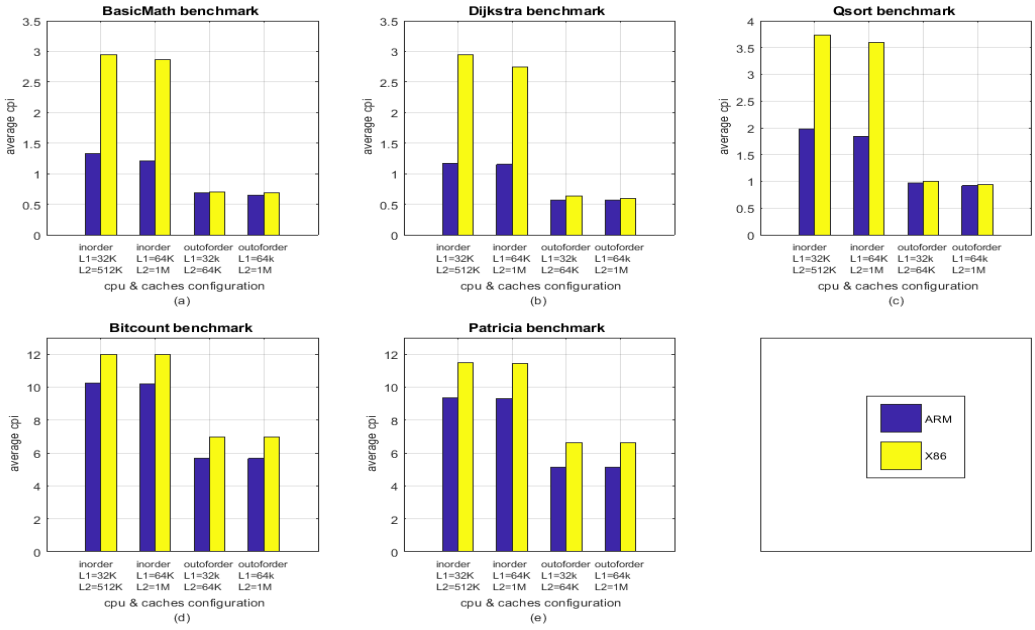

Figure 1. Average cycles per instruction (CPI) test for (a) BasicMath, (b) Dijkstra, (c) Qsort, (d) Bitcount, and (e) Patricia benchmarks.

B. L2 Cache Miss Rate

The results for L2 cache miss rate are depicted in Fig. 2. The results show that as the sizes of L1 and L2 caches are doubled, L2 cache miss rate becomes higher but this doesn't make sense because it's known that the larger caches sizes the less caches miss rate. The only reasonable result is depicted in Fig. 2(c) for Qsort benchmark. A possible reason for these inaccurate results is that the Gem5 simulator was working under the SE mode.

C. Total Energy

Results that are depicted in Fig. 3 show that energy is directly proportional to the code size of the program. Also, the results show that the in-order CPU model consumes much more energy. ARM is always more energy efficient than x86 except for the basicmath benchmark when the CPU model is out-of-order.

D. Throughput

The throughput results are depicted in Fig. 4. The results show that the throughput is increasing when the CPU model is out-of-order. This is due to the fact that in the out-of-order pipeline the caches misses increase.

VI. CONCLUSION AND FUTURE WORK

In this paper, a comparative study is done to evaluate the performance of ARM and x86 architectures. A well-known, five benchmarks, selected from the Mibench are used as a test data and four performance metrics, which are: average cycles per instruction (CPI), L2 cache miss rate, total energy, and throughput, are evaluated. The evaluation is done using Gem5 simulator. The results show that ARM outperforms x86 in the most cases. X86 architecture works much better when the CPU model is out-of-order.

Since we use Gem5 under the SE mode, some results are not reasonable. For the future work, we advise to use the Gem5 under the Full-System (FS) mode to evaluate the performance of these architectures.

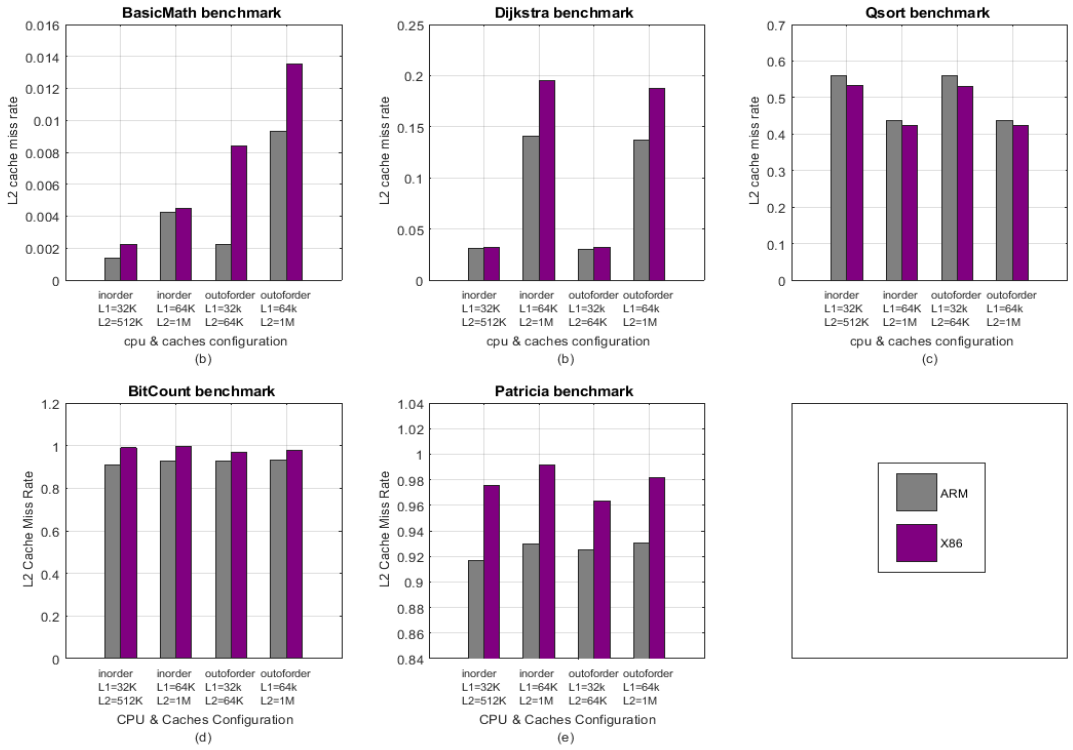

Figure 2. L2 cache miss rate test for (a) BasicMath, (b) Dijkstra, (c) Qsort, (d) Bitcount, and (e) Patricia benchmarks

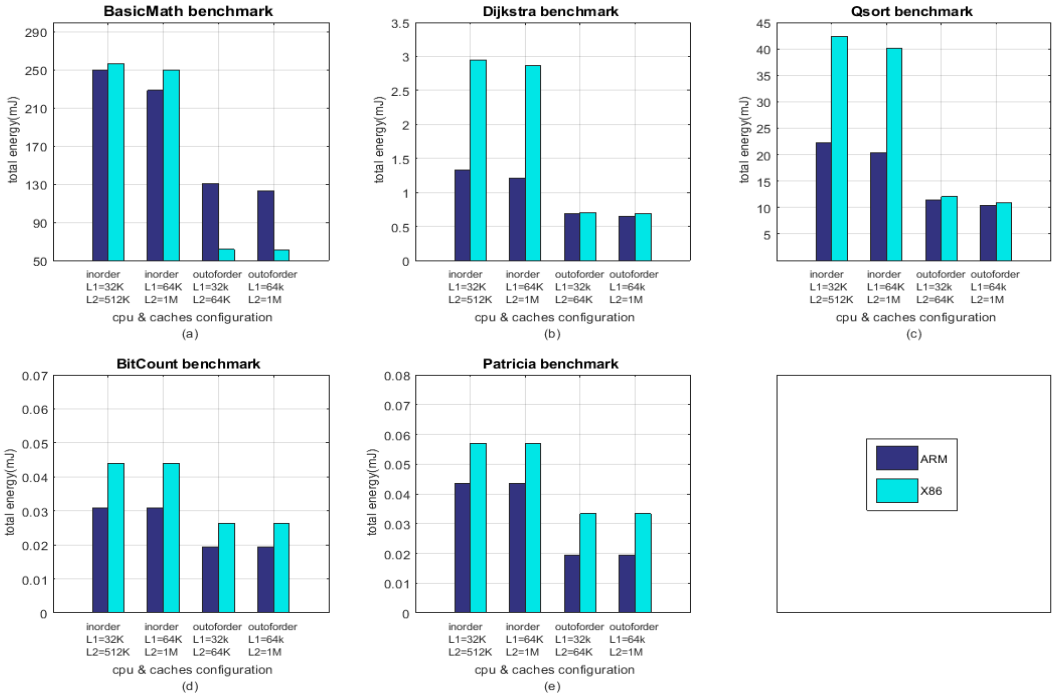

Figure 3. Total energy(mJ) test for (a) BasicMath, (b) Dijkstra, (c) Qsort, (d) Bitcount, and (e) Patricia benchmarks.

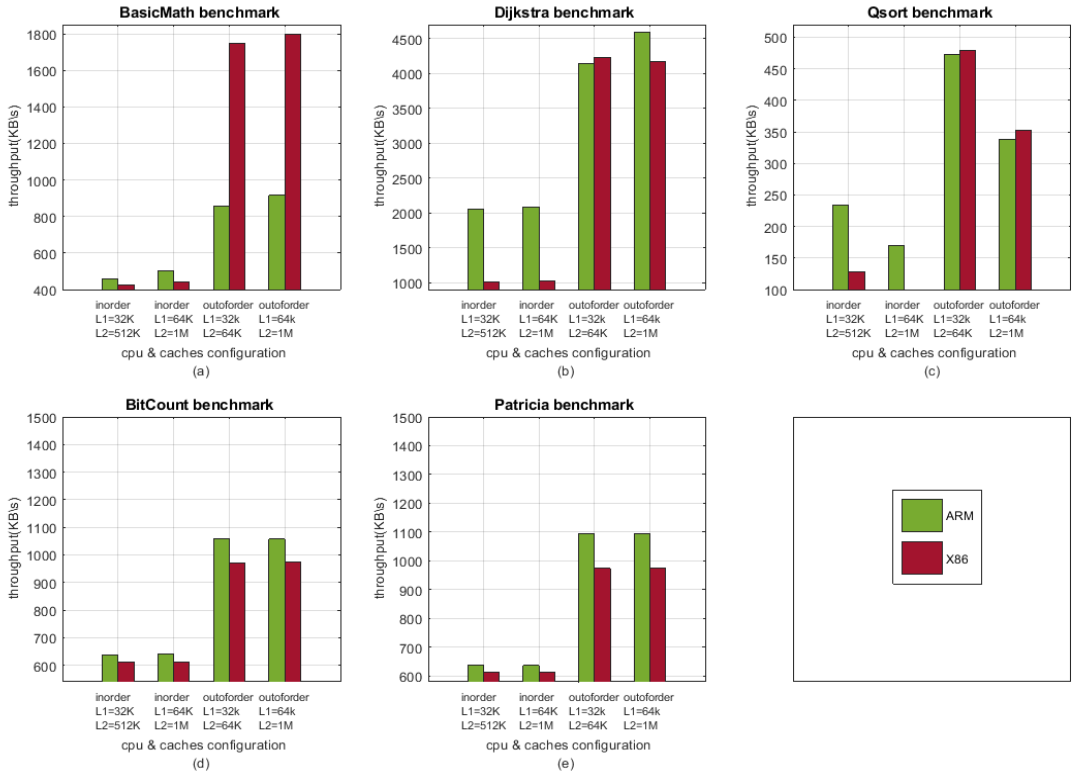

Figure 4. Throughput (KB\s) test for (a) BasicMath, (b) Dijkstra, (c) Qsort, (d) Bitcount, and (e) Patricia benchmark

APPENDIX A

To enable the In-order CPU model for x86 processor follow these steps:

1. Under the gem5 folder open the build_opts folder.
2. Open the x86 text file using gedit program or any other program.
3. The default text in the line two statement is CPU_MODELS='AtomicSimpleCPU, O3CPU, TimingSimpleCPU'
4. Add "MinorCPU" at the end of this statement to be CPU_MODELS='AtomicSimpleCPU, O3CPU, TimingSimpleCPU,MinorCPU'.
5. Save the file and close it.
6. Under the gem5 folder open the build folder, under the build folder open varaiables.
7. Open the x86 text file using gedit program or any other program.
8. Be sure that the second line contains "MinorCPU" CPU_MODELS='AtomicSimpleCPU, O3CPU, TimingSimpleCPU,MinorCPU'.
9. Save the file and close it.
10. Open the terminal, type cd gem5 then press Enter.
11. Type scons build/x86/gem5.opt then press Enter
12. Wait until the build operation is completed.

REFERENCES

[1] I. Intel, "and IA-32 Architectures Software Developer's Manual," *Volume 3A: System Programming Guide, Part*, vol. 1, no. 64, 64.

[2] D. A. Patterson and D. R. Ditzel, "The case for the reduced instruction set computer," ACM SIGARCH Computer Architecture News, vol. 8, no. 6, pp. 25–33, 1980.

[3] D. Brash, "The arm architecture version 6," 2016.

[4] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach. Elsevier, 2011.

[5] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on, 2001, pp. 3–14.

[6] N. Binkert et al., "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol. 39, no. 2, pp. 1–7, 2011.

[7] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator: Modeling networked systems," IEEE Micro, vol. 26, no. 4, pp. 52–60, 2006.

[8] M. M. Martin et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," ACM SIGARCH Computer Architecture News, vol. 33, no. 4, pp. 92–99, 2005.

[9] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system," in Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on, 2012, pp. 1–7.

[10] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," 2001

[11] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, “Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors.,” in Sbac-Pad, 2007, pp. 62–68.

[12] F. Bellard and others, “QEMU open source processor emulator,”. [Online]. Available: https://www.qemu.org/. [Accessed: 15- Oct-2017].

[13] G. Hutton, “A simple guide to the Ruby simulator,” Glasgow University, August, 1991.

[14] R. V. Aroca and L. M. G. Gonçalves, “Towards green data centers: A comparison of x86 and ARM architectures power efficiency,” Journal of Parallel and Distributed Computing, vol. 72, no. 12, pp. 1770–1780, 2012.

[15] F. A. Endo, D. Couroussé, and H.-P. Charles, “Micro-architectural simulation of in-order and out-of-order arm microprocessors with gem5,” in Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on, 2014, pp. 266–273.

[16] H. Rawat and P. Schaumont, "SIMD Instruction Set Extensions for Keccak with Applications to SHA-3, Keyak and Ketje", in Proceedings of the Hardware and Architectural Support for Security and Privacy, 2016, p. 4.