PERFORMANCE BENCHMARKING OF RVC BASED MULTIMEDIA SPECIFICATIONS

Junaid Jameel Ahmad [1*] *Shujun Li* [2] *Marco Mattavelli* [1]

[1] École Polytechnique Fédérale de Lausanne (EPFL), Switzerland [2] University of Surrey, UK

ABSTRACT

The Reconfigurable Video Coding (RVC) framework was developed to specify video codecs as abstract and as much as possible implementation-agnostic descriptions, which are supposed to be processed by code synthesis tools to automatically generate implementations for different target languages and platforms. However, there are still questions about if the run-time performance of these automatically generated RVC-based codec implementations is good enough to run efficiently on different target platforms. In this paper, we present a performance benchmarking study on various RVC-based multimedia specifications (H.264/AVC and JPEG codecs, and four multimedia security systems based on these codecs), which covers the following two aspects: 1) the run-time performance against their corresponding non-RVC implementations on a single-core machine; 2) the performance gain these RVC-based implementations on a dual-core machine. Based on our benchmarking results, which show that RVC-based multimedia implementations achieve adequate/acceptable performance, we conclude that RVC has the potential to become a general-purpose but still performance-efficient development framework for many application domains.

1. INTRODUCTION

Recently, ISO/IEC standardized RVC (Reconfigurable Video Coding) [1, 2], which was developed on the principle of dataflow programming paradigm. Initially, the RVC framework was developed to handle the challenges incurred in the development of complicated video codecs [3, 4], which can naturally be modeled as dataflow systems. But the RVC standard actually offers a general development framework and has also been used for the development of other types of media (3-D graphics [5], image [6], audio [7]) codecs and secure computing systems [8–10], as structurally they are all data-driven.

The main idea behind RVC is to specify a system in such a way that that only the functionalities (or behaviors) of the algorithms are specified via their input and output interfaces, and the implementation choice (conversion of abstract implementation-independent solutions to implementation-dependent source codes) can be made only until a specific target platform has been chosen. With this approach, one abstract design can be used to automatically create implementations towards multiple target platforms. However, there are still concerns about if the performance of automatically generated implementations from abstract RVC specifications is good enough so that RVC can be effectively used to develop practical applications that can run efficiently on different target platforms.

In [8], we partially investigated these concerns by conducting a performance benchmarking study on cryptographic primitives, which are quite simple in their structures and consist of only a few number of functional units (FUs). However, multimedia codecs and multimedia security systems are comparatively more complex in their structures and normally consist of a much larger number of FUs. This motivated us to further investigate if the conclusions of our previous performance benchmarking study can be generalized to more complex application domains e.g. multimedia codecs and multimedia security systems. It deserves mentioning that except one MPEG contribution [11], which investigates frame decoding rates of video decoders, there is no any other work we know, which reports the run-time performance of complex RVC-based implementations on single-core and multi-core platforms.

In this paper, we present a performance benchmarking study of RVC-based C implementations of an H.264/AVC intra codec, a JPEG codec and four multimedia security systems working with the H.264/AVC intro codec and the JPEG codec on both single-core and dual-core machines. For our benchmarking on the single-core machine, we present a side-by-side evaluation of these RVC-based implementations against some selected non-RVC reference implementations and the results show that, the run-time performance of RVC-based implementations is aquequate/comparable to their corresponding non-RVC implementations. In addition, we evaluated the amount of performance gain (i.e. execution speed-up) these benchmarked RVC-based implementations achieved while running on a dual-core machine and the results show that they can achieve a performance gain up to 173%.

The rest of the paper is organized as follow. Section 2 briefly covers the evaluated multimedia security systems and the details of our experimental setup. The results of our benchmarking study on single-core and dual-core machines are discussed in Sec. 3 and Sec. 4, respectively. Conclusions are summarized in Sec. 5.

2. EXPERIMENTAL SETUP

We evaluated RVC-based implementations of the following four multimedia security systems[1] working with H.264/AVC and JPEG codecs:

- **Joint Video Encryption-Encoding (JVEE):** As an example of a JVEE system, we specified joint sign bit encryption and decryption of H.264/AVC videos, where the ARC4 stream cipher was used as the underlying cryptosystem to flip the sign bits of all DCT coefficients.
- **Joint Image Encryption-Encoding (JIEE):** As an example of a JIEE system, we specified DC encryption and decryption of JPEG images, where the ARC4 stream cipher was again used as the underlying cryptosystem to encrypt/decrypt the fixed-length bits of DC coefficients.

*This work was performed when the first author was studying at the University of Konstanz, Germany and was partly supported by a doctoral fellowship offered by the Zukunftskolleg and a doctoral scholarship offered by the Baden-Württemberg Landesgraduiertenförderung (LGFG), Germany.

[1]Greater details on the RVC specifications of these systems can be found in [12, 10].

- **Compressed Domain JPEG Image Watermarking Embedding (JIWE):** As an example of a JIWE system, we chose a compressed domain image watermarking scheme proposed in [13] for H.264/AVC videos but we ported this watermarking scheme to work with JPEG images. In this watermarking scheme, a number of macroblocks are randomly selected for watermark embedding and in each selected macroblock one watermark bit is embedded in exactly one quantized AC coefficient. The random paths at the macroblock level and the AC coefficients level are both driven by a stream cipher, so an attacker does not have any knowledge about the locations where the watermark bits are embedded [13].
- **Compressed Domain JPEG Image Steganography Embedding (JISE):** As an example of a JISE system, we specified the JPEG compressed-domain image steganography scheme called F5 [14]. In this scheme, the secret message is embedded into the non-zero AC coefficients of the whole image using a matrix embedding scheme. Before the matrix embedding operation, the non-zero AC coefficients are permuted using some pseudo random generator so that the attacker cannot find the sequence in which the AC coefficients are used by the underlying matrix embedding scheme. In [14], the matrix embedding scheme used in F5 is based on the $(1, n, k)$ Hamming distance code, where $n = 2^k - 1$. Hence, to embed every k bits of the message, the embedding scheme changes at most one element in each set of n non-zero AC coefficients. The parameters k and n for each steganographic operation are computed as a function of the size of the secret message being embedded and the number of non-zero AC coefficients such that the message just fits into the carrier image.

For all evaluated RVC specifications, we generated C source code using CAL2C code generation backend of ORCC [15]. Their corresponding non-RVC reference implementations of the H.264/AVC codec and the JPEG codec were developed based on JM [16], the reference software of H.264/AVC standard, and IJG's JPEG codec written in C [17], respectively. Regarding the C compiler, we used the one in Microsoft Visual Studio 2008.

As a test machine for this benchmarking study, we used a general-purpose desktop PC (HP Compaq 8000 Elite Convertible Minitower with an Intel Pentium Dual-Core E5400 2.70GHz CPU and 2.0 GB main memory) and conducted all experiments under the safe-mode command prompt of Windows 7 Professional. All benchmarking results are reported in time units, measured by the high-resolution performance counters `QueryPerformanceCounter()` and `QueryPerformanceFrequency()` in Win32 API [18].

For experiments on the single-core machine, we configured our test machine to run only one of its CPU cores to simulate a real single-core machine. For experiments on the dual-core machine, the performance gain of all the evaluated implementations was measured with reference to the performance achieved on experiments on the single-core machine. For the results reported in this paper, all the FUs of any given RVC specification are manually categorized into two partitions[2], which run as independent threads with each thread running on its designated CPU-core of the system. This manual categorization/partitioning of FUs was performed based on the following criteria: 1) whenever possible we kept closely-dependent FUs in the same partition, 2) maximizing the performance by making the partitions to share the workload as equally as possible[3].

For applications based on the H.264/AVC codec, we report the run-time performance for encoding and decoding of only the first 99 frames of three test videos – foreman, highway and suzie (all with QCIF resolution). All videos were encoded with $QP = 20$. For applications based on JPEG codec, we report the run-time performance for encoding and decoding of three test images – airplane (512×512 resolution), Lena (512×512 resolution) and yacht (512×480 resolution). All images were encoded with the example quantization table of JPEG [20, Table K.1] and the quality factor 80.

3. BENCHMARKING ON SINGLE-CORE MACHINE

In this section, we present the benchmarking results for RVC-based implementations of the evaluated multimedia specifications against their corresponding non-RVC reference implementations.

3.1. H.264/AVC Applications

Figures 1a and 1b show the performance benchmarking results for evaluated implementations based on both RVC and JM codecs. We can observe that the run-time performance of current implementations of RVC-based intra encoder and JVEE system are (more than 100%) faster than their corresponding JM based implementations. This is may be due to the fact that the evaluated RVC-based H.264/AVC encoder is still not complex – as it does not yet support all encoding options (e.g., profiles, levels, inter encoding of P and B frames, etc.). Moreover, the run-time performances of RVC- and JM-based implementations are quite close and comparable to each other.

It should be noted that, the sign-bit flipper module added to H.264/AVC JVEE and JVDD systems (for encryption and decryption, respectively) makes them consume some extra time to run than simpler encoder and decoder. In order to have an idea of how much overhead JVEE and JVDD systems introduce to the video codec, Table 1 shows the percentage of the total time consumed by the sign-bit flipper module for both JM and RVC based implementations of JVEE and JVDD systems.

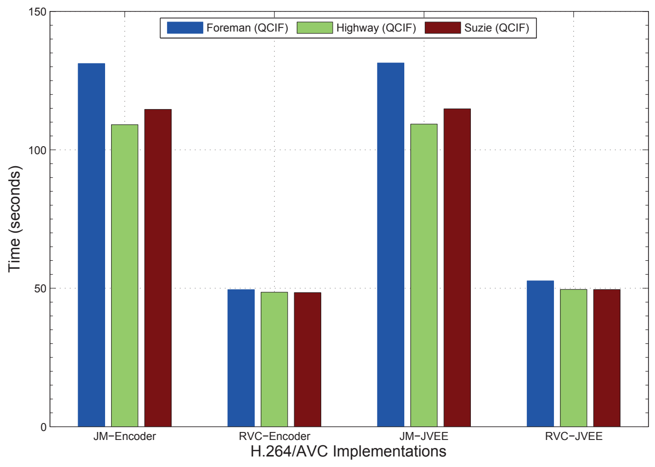
Our results show that, the time overheads caused by RVC-based implementation of sign bits encryption and decryption are always below 7.0%. On the other hand, the time overheads caused by JM-based implementation of sign bits encryption and decryption are significantly smaller (less than 0.20% and 3.0%, respectively). Note that, the magnitudes of the time overheads for the JM-JVEE and JM-JVDD vary a lot. This can be explained by the fact that the JM encoder takes far more time than the JM decoder (see Fig. 1), while the the sign-bit flipper depletes roughly the same absolute time regardless if its working with the JM encoder or the decoder. As a result, the percentage of overhead time for the JM-JVEE becomes very small. Overall, we can conclude that, even in terms of the overhead time for sign bits encryption and decryption, the cost incurred by RVC-based implementations is comparable to the JM-based implementations.

3.2. JPEG Applications

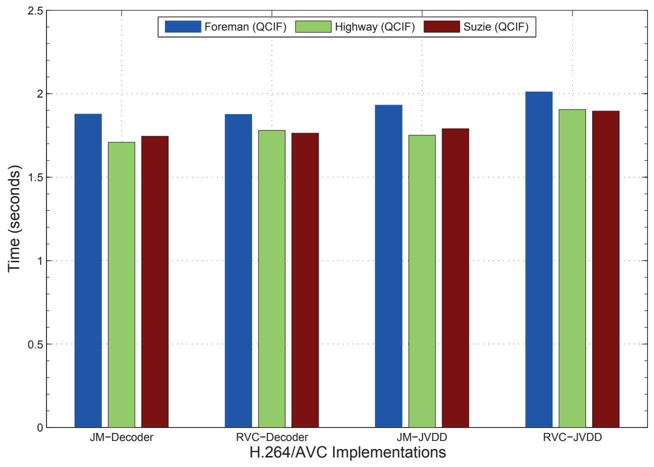
Figures 2a and 2b show the performance benchmarking results of the evaluated implementations based on RVC and IJG codecs. For all implementations, one can observe that the run-time performance of

[2]In order to keep our discussion more focused towards benchmarking results, partitions of evaluated specifications are not presented in this paper.

[3]In future, we plan to use design space exploitation tools [19] to automatically suggest us more intelligent partitioning layouts of FUs.

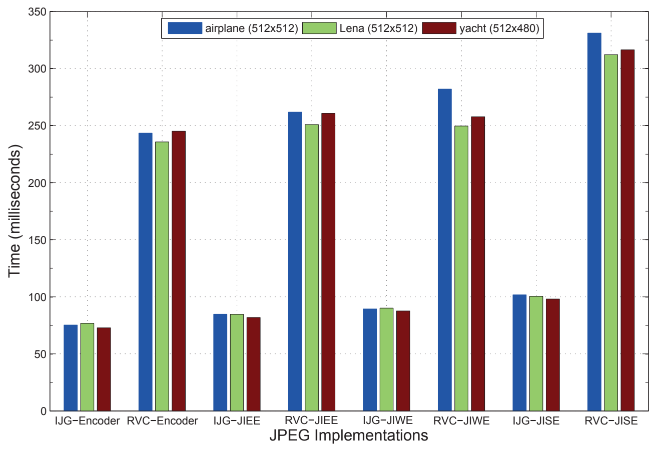


(a) Encoding and encryption-encoding

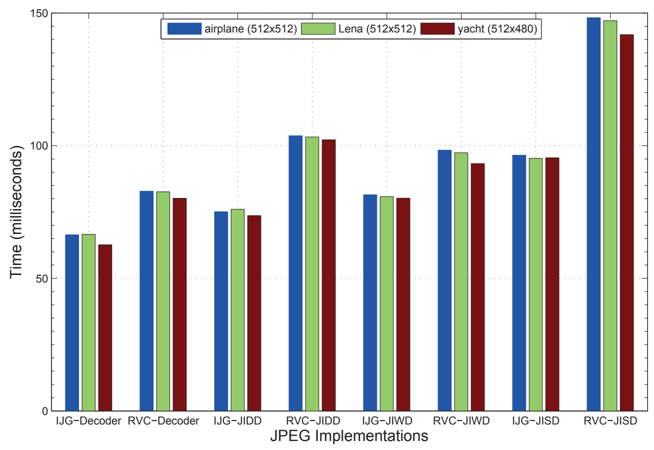


(b) Decoding and decryption-decoding

Fig. 1: Benchmarking results of H.264/AVC implementations.



(a) Encoder, encryption-encoder, watermark-embedder and steganographic-embedder



(b) Decoder, decryption-decoder, watermark-detector and steganographic-detector

Fig. 2: Benchmarking results of JPEG implementations.

Table 2: Overhead time consumed by the added security module in JPEG security systems.

(a) RVC

Test Images	**JIEE**	**JIDD**	**JIWE**	**JIWD**	**JISE**	**JISD**
airplane	7.02%	20.13%	13.67%	15.73%	26.46%	44.11%
Lenna	6.06%	20.00%	5.60%	15.13%	24.50%	43.85%
peppers	6.00%	21.59%	4.91%	14.09%	22.52%	43.52%

(b) IJG

Test Images	**JIEE**	**JIDD**	**JIWE**	**JIWD**	**JISE**	**JISD**
airplane	11.25%	11.55%	15.84%	18.50%	26.08%	31.12%
Lenna	9.28%	12.50%	14.80%	17.66%	23.52%	30.11%
peppers	11.13%	15.01%	16.85%	21.98%	27.77%	34.39%

RVC-based encoder implementations are (more than 200%) slower than their corresponding IJG-based counterparts. This is due to the

Table 1: Overhead time consumed by the sign-bit flipper module.

Test Videos	JM-JVEE	JM-JVDD	RVC-JVEE	RVC-JVDD
foreman	0.16%	2.77%	6.01%	6.73%
highway	0.18%	2.41%	1.98%	6.55%
suzie	0.17%	2.54%	2.25%	6.97%

fact that the evaluated RVC-based JPEG encoder is less optimized than the IJG one (which has been carefully optimized for many years). However, the run-time performance of RVC- and IJG-based decoder implementations are comparable to each other. This suggests that the RVC-based JPEG decoder is already good enough and may not require too much further optimization.

Like Table 1, Table 2 shows the percentage of the total time consumed by the added security module to both IJG and RVC based implementations of image encryption encoder/decoder, watermarking embedder/detector and steganographic embedder/detector systems.

The time overheads caused by RVC-based JPEG DC encryption and decryption are less than 8% and 22%, respectively. Similarly, the time overheads for RVC-based JPEG image watermark embedder and detector are less than 6.00% (with airplane consuming comparatively an odd amount of 13.67%) and 16%, respectively. Moreover, RVC-based JPEG image steganographic embedder and detector have an exceptionally higher overhead time: 22% to 27% and 43% to 45%, respectively. This can be explained by the fact that the steganographic technique being evaluated requires buffering the whole image in order to pseudo-randomly permute it and then perform the matrix embedding operation to actually embed the secret message into the carrier image. This activity results in consuming an exceptionally higher amount of overhead time.

The time overheads caused by the IJG-based implementations generally follow the trends observed for RVC-based implementations (except for JPEG DC encryption and decryption, which consumes higher overhead time than the corresponding RVC-based implementations). Similarly, JPEG steganographic embedder and detector schemes also consume higher amount of overhead time because of the same reasons reported above.

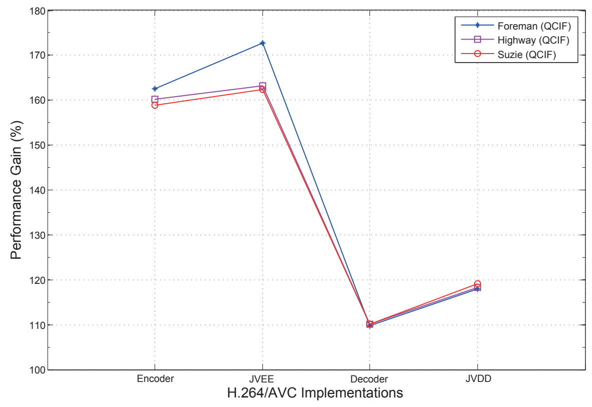
4. BENCHMARKING ON DUAL-CORE MACHINE

In this section, we present the performance benchmarking results of evaluating the amount of performance gain RVC-based implementations of H.264/AVC and JPEG codecs and the four multimedia security systems can achieve on a dual-core machine.

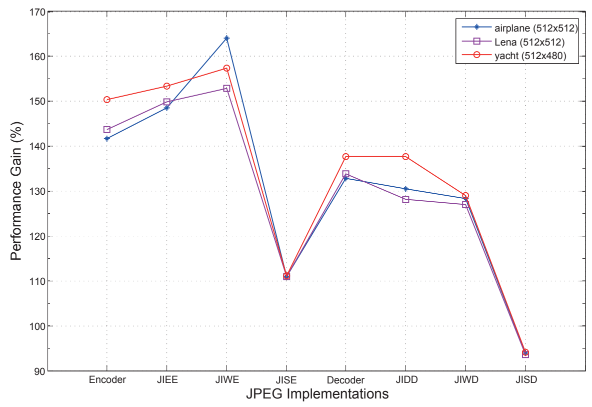
4.1. H.264/AVC Applications

Figure 3a shows the performance gain we observed for RVC-based implementations of H.264/AVC codec and sign bits encryption system. One can observe that both of the encoder implementations achieve a performance gain between 159% and 173%. However, the performance gain achieved by both of the decoder implementations is lower – between 110% to 119%.

These results lead to the following two interesting observations. First, even though both JVEE and JVDD systems includes an added sign bits flipper module, the performance gains achieved by both systems are better than the simpler encoder and decoder, respectively. Second, we know that the H.264/AVC decoder is not as complex as H.264/AVC encoder (i.e., decoder consists of a lesser number of FUs than the encoder) but the performance gains achieved by the decoder



(a) H.264/AVC



(b) JPEG

Fig. 3: Performance gain we achieved for RVC-based implementations on a dual-core machine.

implementations is far lower than the performance gains achieved by the encoder implementations. Based on these two observations, one may hypothesize that, the encoder specifications have more parallelizable components/FUs than the decoder specifications and the increase in the complexity of an RVC specification provides an opportunity to reconfigure the specification in a way to exploit more parallelism. More insights about these aspects can be determined by conducting a low-level profiling of these RVC encoder and decoder specifications. In addition, the low-level profiling may also be helpful in acquiring the best possible load-balancing of the parallelizable components and may further improve the overall performance gain. In future, we plan to study these aspects in more detail.

4.2. JPEG Applications

Figure 3b shows the performance gain we observed for RVC-based implementations of JPEG codec, encryption, watermarking and steganographic systems. The performance gains achieved by the steganographic implementations are quite low (for JISD it is even negative). This can be explained by the algorithmic detail of steganographic technique being evaluated – the embedding and detection of the secret message requires buffering the whole image. Hence, the steganographic modules become the bottleneck and (almost) freezes the dataflow in all other FUs of the system, and as a result this severe performance drop is created.

With the steganographic system as an exception, the performance gains achieved by the encoding implementations are comparatively higher (between 142% and 164%) than the gains achieved by the decoding implementations (between 127% and 148%). This matches to the results we obtained for the H.264/AVC-based implementations and adds more evidence to our hypothesis on why this happens.

5. CONCLUSIONS

In this paper, we report our benchmarking results on the run-time performance of RVC-based implementations of multimedia specifications against their corresponding non-RVC reference implementations on a single-core machine and their performance gain on a dual-core machine. Our benchmarking results showed that automatically generated RVC-based implementations can achieve adequate/acceptable performance, hence answered the concern about the performance of RVC-based implementations. These results encourage the community to further evolve RVC and make it a general-purpose but still performance-efficient development framework for application domains beyond video codecs.

6. REFERENCES

[1] ISO/IEC, "Information technology – MPEG systems technologies – Part 4: Codec configuration representation," ISO/IEC 23001-4, 2nd Edition, December 2011.

[2] ISO/IEC, "Information technology – MPEG video technologies – Part 4: Video tool library," ISO/IEC 23002-4, January 2010.

[3] Ihab Amer, Christophe Lucarz, Ghislain Roquier, Marco Mattavelli, Mickaël Raulet, Jean-François Nezan, and Olivier Déforges, "Reconfigurable Video Coding on multicore: An overview of its main objectives," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 113–123, 2009.

[4] Shuvra Bhattacharyya, Johan Eker, Jörn W. Janneck, Christophe Lucarz, Marco Mattavelli, and Mickaël Raulet, "Overview of the MPEG Reconfigurable Video Coding framework," *J. Signal Processing Systems*, vol. 63, no. 2, pp. 251–263, 2011.

[5] Sinwook Lee, Taehee Lim, E.S. Jang, Ji Hyung Lee, and SeungwookLee, "MPEG Reconfigurable Graphics Coding framework: Overview and applications," in *Proc. 2011 IEEE Visual Communications and Image Processing Conference (VCIP 2011)*. 2011, IEEE.

[6] "RVC implementation of JPEG codec," `http://orc-apps.svn.sourceforge.net/viewvc/orc-apps/trunk/JPEG/`.

[7] Hazem Ismail Abdel Aziz Ali and Mohammad Nazrul Ishlam Patoary, "Design and implementation of an audio codec (AMR-WB) using dataflow programming language CAL in the OpenDF environment," TR: IDE1009, Halmstad University, Sweden, 2010.

[8] Junaid Jameel Ahmad, Shujun Li, Richard Thavot, and Marco Mattavelli, "Secure Computing with the MPEG RVC Framework," *Signal Processing: Image Communication, special issue on Reconfigurable Media Computing*, to appear 2013.

[9] Junaid Jameel Ahmad, Shujun Li, Ahmad-Reza Sadeghi, and Thomas Schneider, "CTL: A Platform-Independent Crypto Tools Library Based on Dataflow Programming Paradigm," in *Proceedings of 16th International Conference Financial Cryptography and Data Security (FC 2012)*. 2012, vol. 7397 of *Lecture Notes in Computer Science*, pp. 299–313, Springer, An extended edition is available at `http://eprint.iacr.org/2011/697`.

[10] Junaid Jameel Ahmad, Shujun Li, Ihab Amer, and Marco Mattavelli, "Building multimedia security applications in the MPEG Reconfigurable Video Coding (RVC) framework," in *Proceedings of 2011 ACM SIGMM Multimedia and Security Workshop (MM&Sec 2011)*. 2011, pp. 121–130, ACM.

[11] Damien de Saint Jorre, Jérôme Gorin, Jean-François Nezan, Mickaël Raulet, Nicolas Siret, Matthieu Wipliez, and Hervé Yviquel, "Report on performance of generated code (C, LLVM, and VHDL) from RVC descriptions," ISO/IEC JTC1/SC29/WG11, MPEG2011/m20074, 96th MPEG Meeting, Geneva, Switzerland, March 2011.

[12] Junaid Jameel Ahmad, *Secure Computing with the MPEG RVC Framework*, Ph.D. thesis, University of Konstanz, Germany, December 2012, Available at `http://kops.ub.uni-konstanz.de/handle/urn:nbn:de:bsz:352-221317`.

[13] Maneli Noorkami and Russell M. Mersereau, "Compressed-domain video watermarking for H.264," in *Proc. 2005 IEEE International Conference on Image Processing (ICIP 2005)*. 2005, vol. 2, pp. 890–893, IEEE.

[14] Andreas Westfeld, "F5–a steganographic algorithm: High capacity despite better steganalysis," in *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings*. 2001, vol. 2137 of *Lecture Notes in Computer Science*, pp. 289–302, Spring.

[15] "Open RVC-CAL Compiler (ORCC)," `http://sourceforge.net/projects/orcc`.

[16] "JM: H.264/AVC reference software," Evaluated software version: 18.4, `http://iphome.hhi.de/suehring/tml`, August 2012.

[17] "Independent JPEG Group," Evaluated software version: 8d, `http://www.ijg.org/files/`, January 2012.

[18] "About Timers (Windows)," `http://msdn.microsoft.com/en-us/library/windows/desktop/ms644900(v=vs.85).aspx`.

[19] Simone Casale Brunet, Abdallah Elguindy, Endri Bezati, Richard Thavot, Ghislain Roquier, Marco Mattavelli, and Jörn W. Janneck, "Methods to Explore Design Space for MPEG RVC Codec Specifications," *Signal Processing: Image Communication, special issue on Reconfigurable Media Computing*, to appear 2013.

[20] ISO/IEC, "Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines," ISO/IEC 10918-1 (JPEG), 1994.