Practical Privacy-Preserving Video Surveillance

Andreas Unterweger

Department of Computer Sciences
University of Salzburg
&
School of Information Technology and Systems Management
Salzburg University of Applied Sciences

June 25, 2013

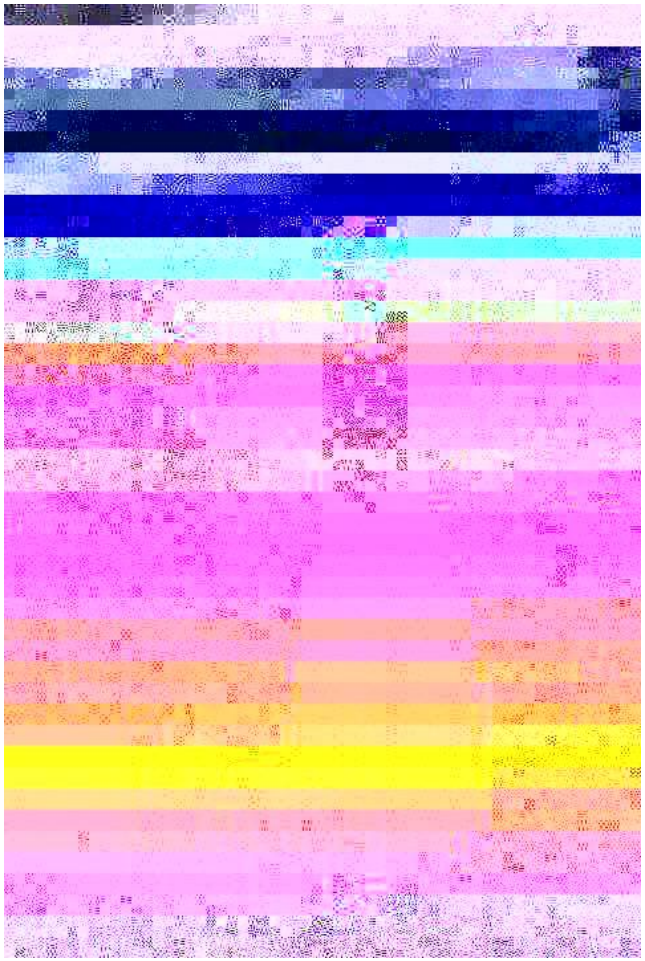
Context: Project overview

- Scenario
  - Surveillance cameras capture people → privacy is an issue
  - Solution: Encryption for privacy preservation
  - Quasi all existing approaches focus on in-compression encryption
  - Issue: In-camera compression process cannot be intercepted
- Project PrivSurv (Privacy for Surveillance)
  - Development of post-compression encryption approaches for
    - Picture formats: JPEG, JPEG 2000 and JPEG XR
    - Video formats: H.264 and SVC
  - Institutions involved:
    - University of Salzburg
    - Salzburg University of Applied Sciences
    - Commend International

Overview

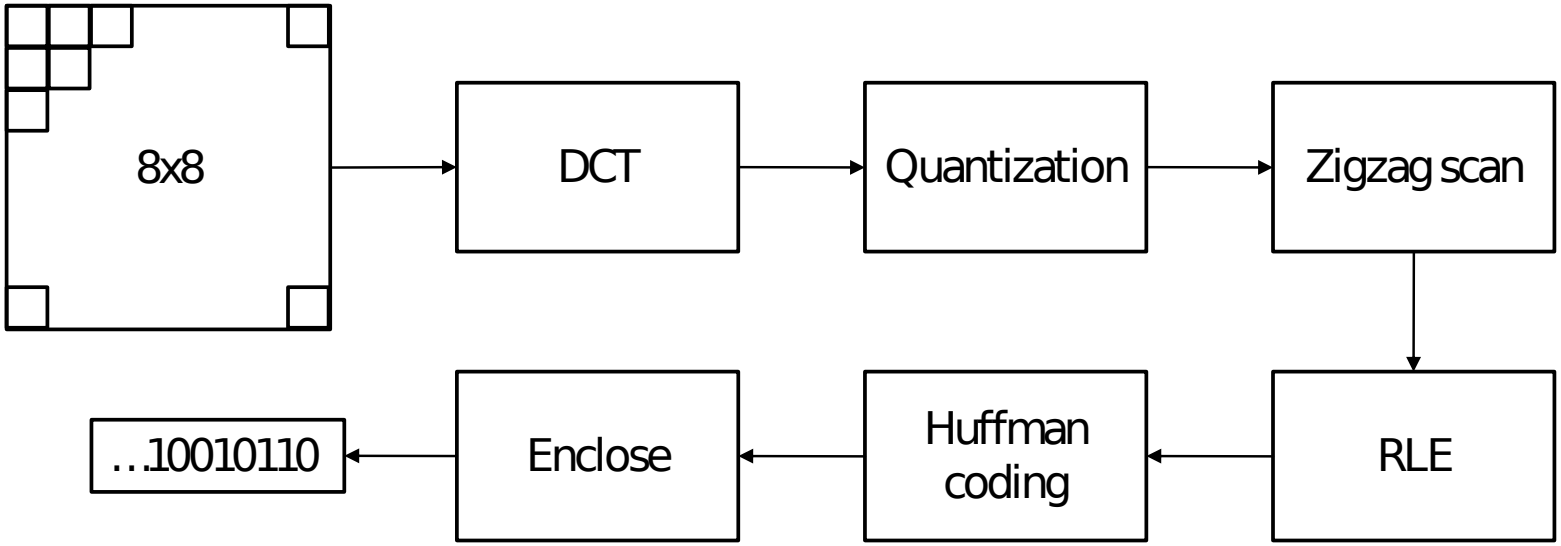
- Typical environment
  - Cameras deliver (Motion) JPEG
  - No possibility to encode in-camera
  - Need for fast and cheap post-processing
- Proposed approach and implementation
  - Encrypts (Motion) JPEG files/streams
  - Processes VGA@25fps in real-time
  - Keeps file(s) format-compliant
  - Preserves file size

Encryption example



JPEG overview

- Picture is split into blocks of size $8 \cdot 8$ (luma samples)
- Each block is processed separately



- Corresponding luma and chroma blocks form an iMCU
- JPEG files consist of multiple compressed iMCUs and headers

Encryption approach overview

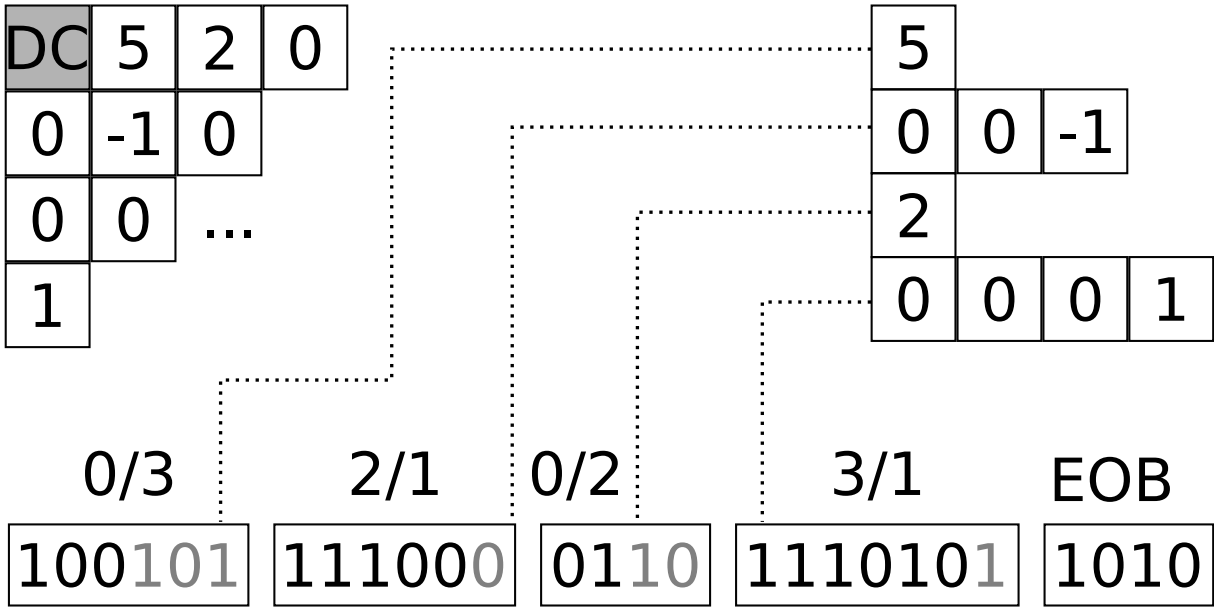
- Three operations per iMCU
  - Change order of run-length-value pairs (within each block)
  - Scramble value bits
  - Swap order of compatible blocks
- Practical considerations
  - PRNG required for reordering and scrambling $\rightarrow$ AES in OFB mode
  - iMCUs are spatially limited $\rightarrow$ support for selective encryption
  - DC coefficients are relatively easy to attack $\rightarrow$ encrypt them with a different key and algorithm

Approach part 1 – Run-length-value pair order change

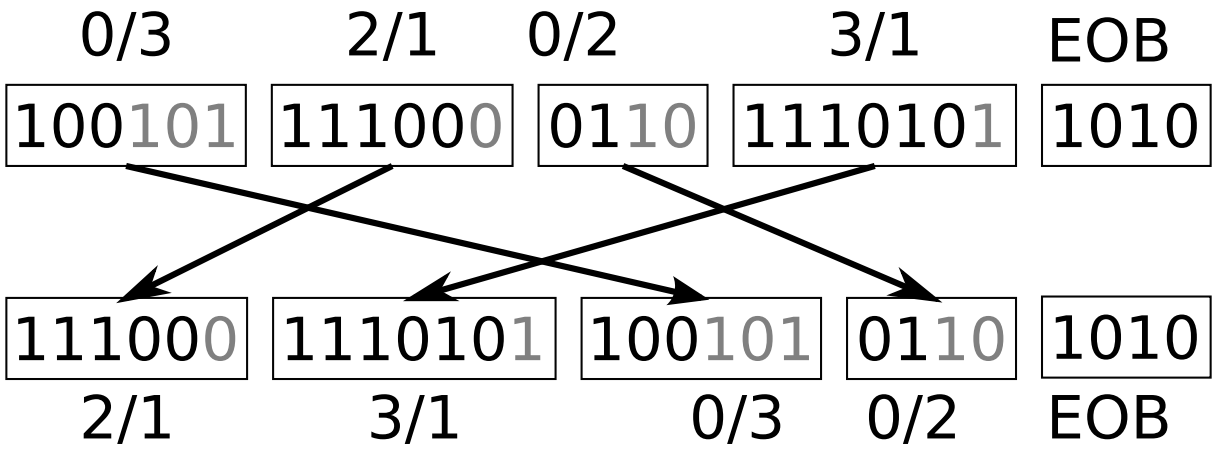
DC	5	2	0
0	-1	0	
0	0	...	
1			

5			
0	0	-1	
2			
0	0	0	1

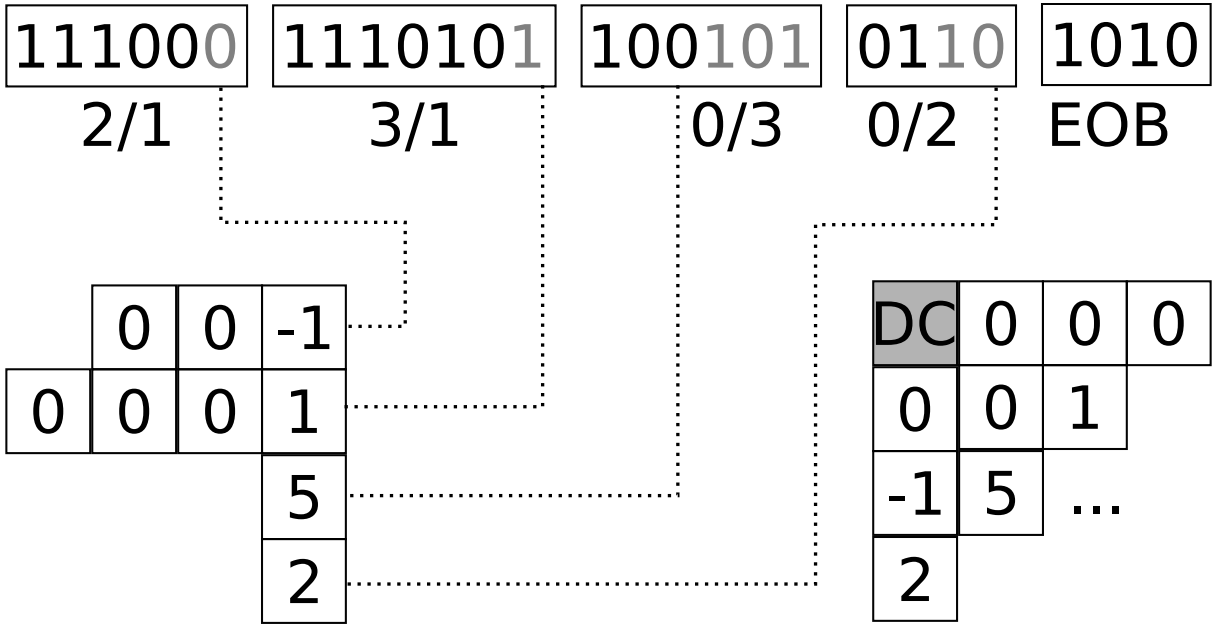
Approach part 1 – Run-length-value pair order change



Approach part 1 – Run-length-value pair order change



Approach part 1 – Run-length-value pair order change



Approach part 1 – Run-length-value pair order change

DC	5	2	0
0	-1	0	
0	0	...	
1			

Before

DC	0	0	0
0	0	1	
-1	5	...	
2			

After

Approach part 2 – Value bit scrambling

AC coefficients (values in run-length-value pairs):

0/3	2/1	0/2	3/1	EOB
100101	111000	0110	1110101	1010

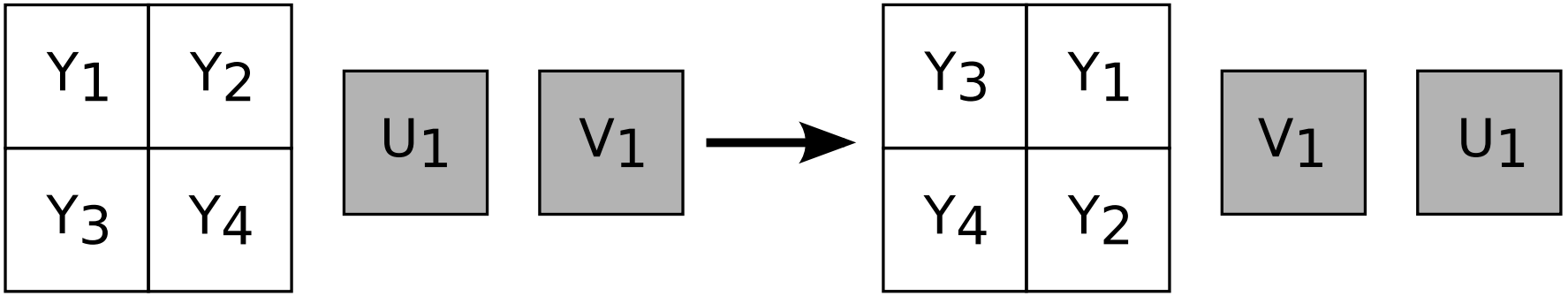
Before

0/3	2/1	0/2	3/1	EOB
100110	111001	0100	1110101	1010

After

DC differences are scrambled in a similar way

Approach part 3 – Intra-iMCU block swapping

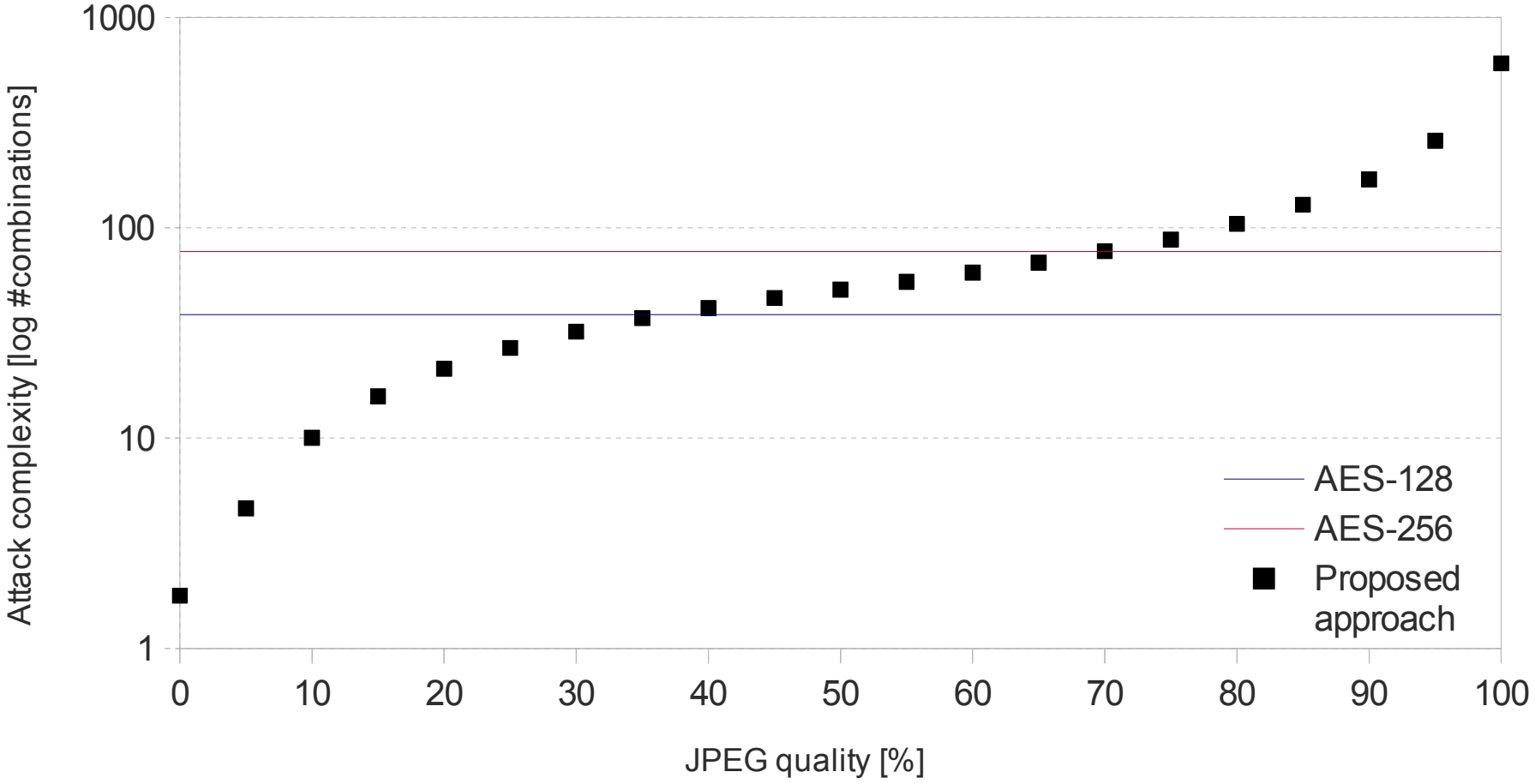


- Example: iMCU with 6 blocks (4 luma, 2 chroma)
- Luma and chroma blocks use different Huffman code words
- Block swapping between luma and chroma is not possible
- Inter-luma-block swapping is possible

Security assessment I

- Analysis of attack complexity (re-reordering and descrambling)
- DC differences are not considered since they are easy to attack
- Tests with LIVE image data base and JPEG reference software
- Attack complexity increases with JPEG quality
- Typical picture: 75% JPEG quality; complexity $\approx 10^{87}$ ways/iMCU
- For comparison: AES-256 key space $\approx 10^{77}$
- Better bruteforce-search AES key than try to decode one iMCU
- Detailed derivation in paper(s)

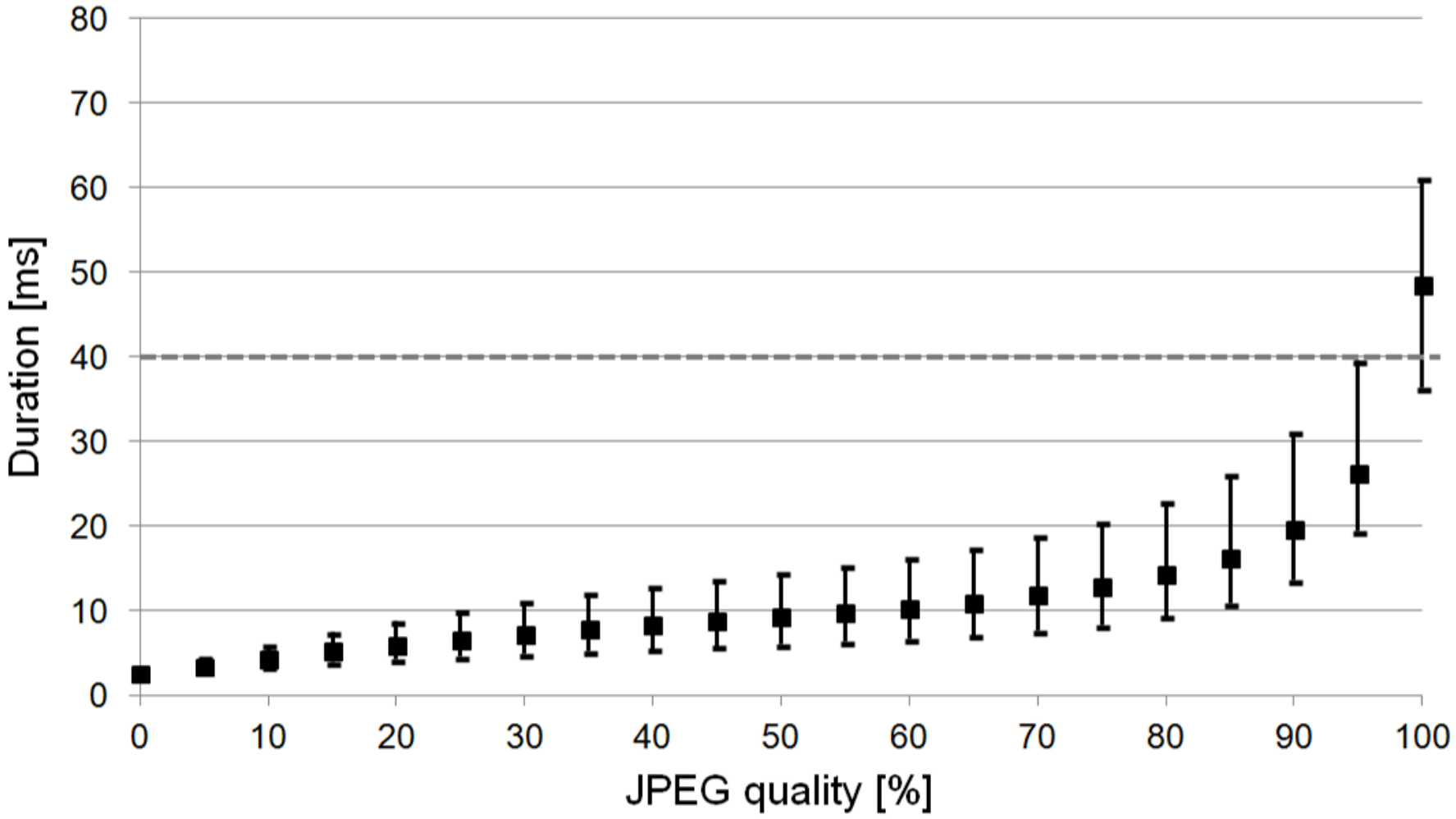
Security assessment II – Typical attack complexity



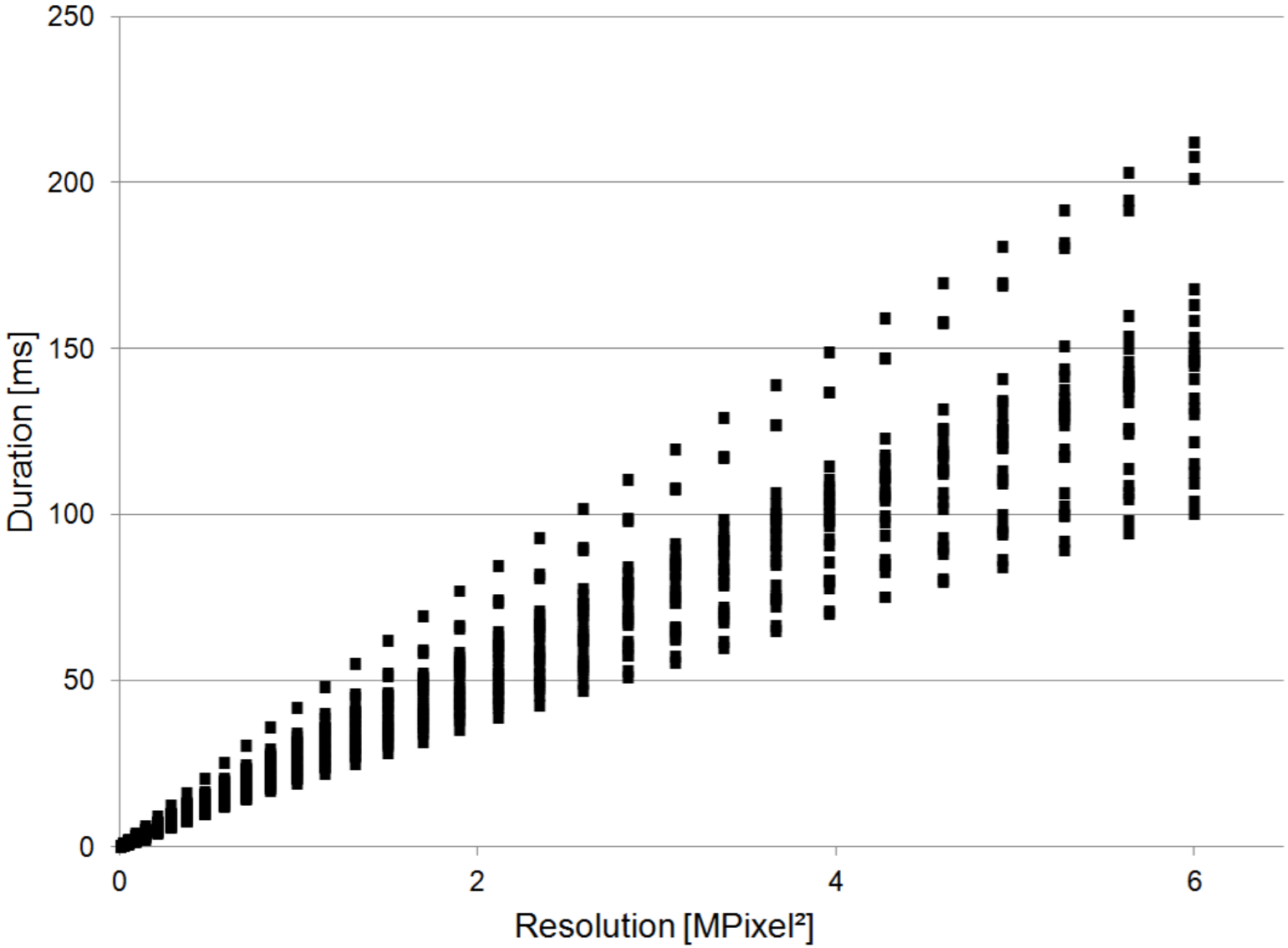
Implementation (Stefan Auer & Alexander Bliem)

- C library based on NanoJPEG (portable)
- iMCU/block identification and Huffman codeword separation
- On-the-fly encryption using the proposed approach
- Done: PC version (Windows, Linux)
- In the works: Raspberry Pi version
- Current results: File-by-file processing on a notebook
  - Intel Core 2 Duo T9600 @ 2.8GHz
  - Microsoft Windows 7 32-bit (safe mode, no services)

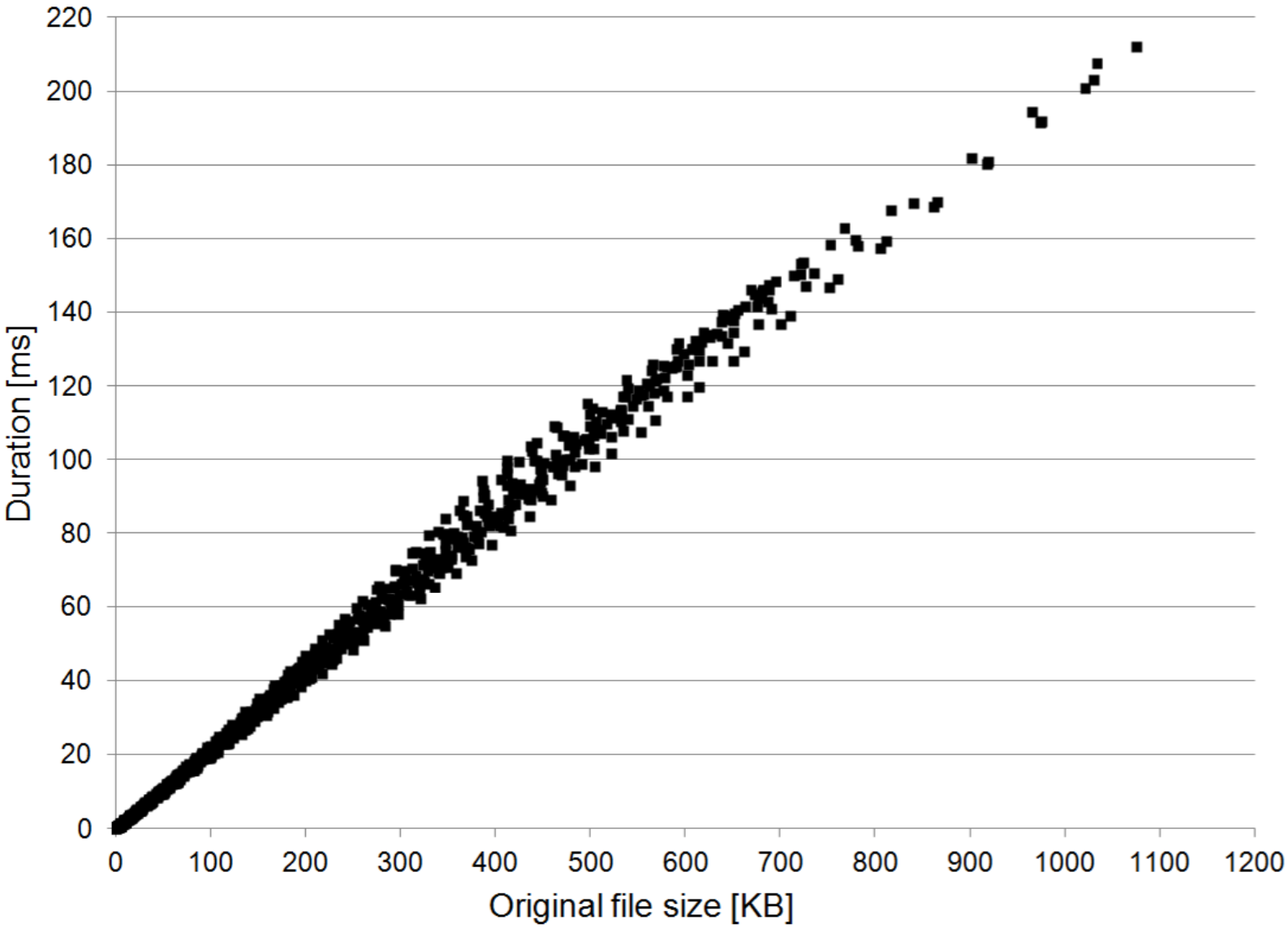
Run-time performance I: Offline (with cache warming)



Run-time performance II: Offline (with cache warming)



Run-time performance III: Offline (with cache warming)



Run-time performance IV: Online (hard real-time)

Approx. 100.000 pictures VGA@25fps (40 ms max. processing time):

JPEG quality	**Max. duration [ms]**	**Avg. duration [ms]**
95%	13.86	12.11 (std.dev.: 0.87)
100%	29.80	26.34 (std.dev.: 1.16)

→ Hard-real time encryption of 100% quality VGA-sized JPEG possible

Future work

- Swap blocks between different iMCUs (increases attack complexity)
- Encrypt only a part of the picture (RoI)
  - Easy to do on an iMCU basis (iMCUs are independent)
  - Hard to signal in a length-preserving way → allowing (small) length changes or distortions makes things easier (→ RoI signalling paper)

Conclusion

- Encryption approach for JPEG files and Motion JPEG streams
  - Format-compliant
  - Length-preserving
  - Secure
- Implementation for PCs and embedded system(s)
  - Capable of real-time (VGA@25fps) encryption
  - Encryption duration depends on file size/resolution
  - Portable
- Bit-stream-based application $\rightarrow$ no recompression necessary
- Extensions like RoI encryption possible $\rightarrow$ e.g. face-only encryption

References

- Proposed encryption approach and implementation:
  - Andreas Unterweger and Andreas Uhl, "Length-preserving Bit-stream-based JPEG Encryption", In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop*, pp. 85-89, ACM, Sep. 2012.
  - Stefan Auer, Alexander Bliem, Dominik Engel, Andreas Uhl, Andreas Unterweger, "Bitstream-based JPEG Encryption in Real-time". In Chang-Tsun Li (Ed.) International Journal of Digital Crime and Forensics, vol. 5(3), *accepted*.
- RoI signalling in JPEG with and without length changes:
  - Dominik Engel, Andreas Uhl and Andreas Unterweger, "Region of Interest Signalling For Encrypted JPEG Images", In *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*, pp. 165-174, ACM, Jun. 2013.

Thank you for your attention!

Questions?