Length-preserving Bit-stream-based JPEG Encryption

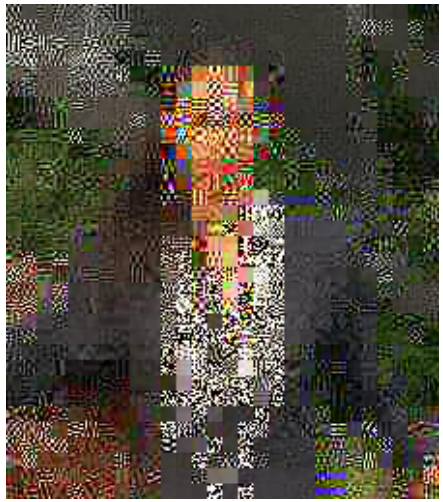
Andreas Unterweger

Department of Computer Sciences
University of Salzburg

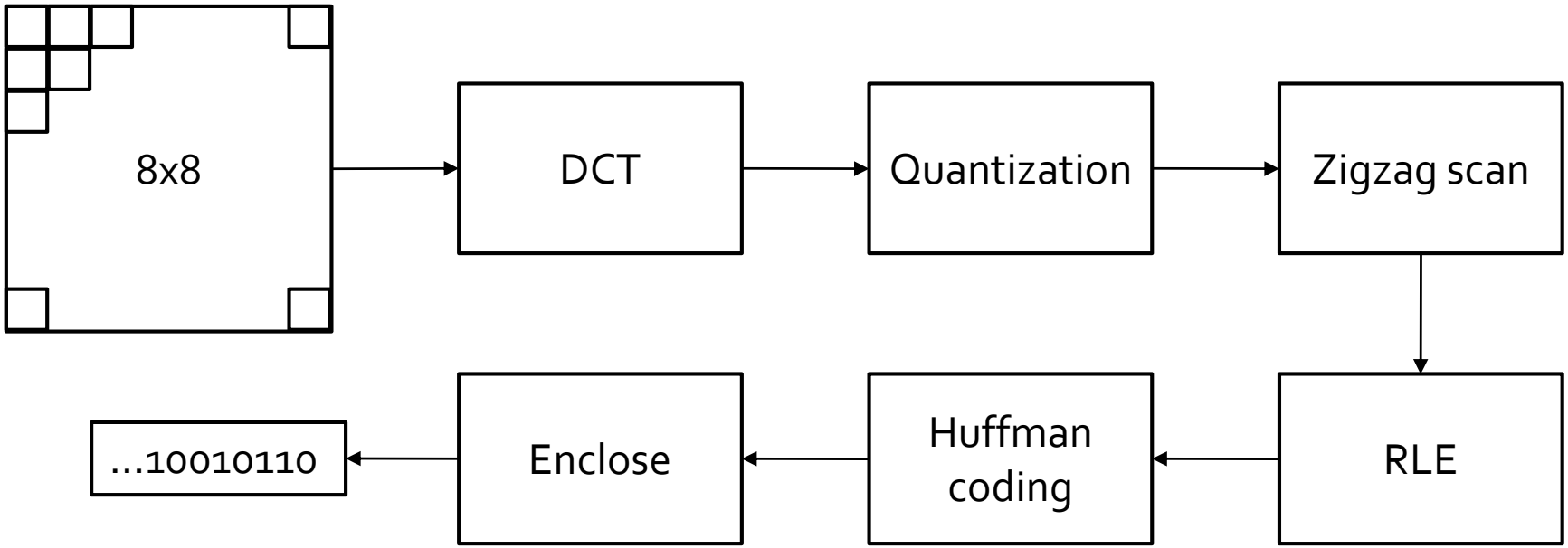
June 8, 2012

Goals

- What?
  - Encrypt a given JPEG file
  - Don't change its length
  - Keep it format-compliant
- What for?
  - Privacy
  - Content control
  - ...



Brief summary: JPEG compression

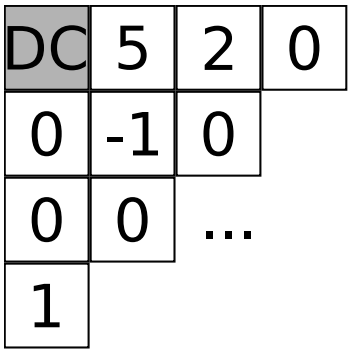


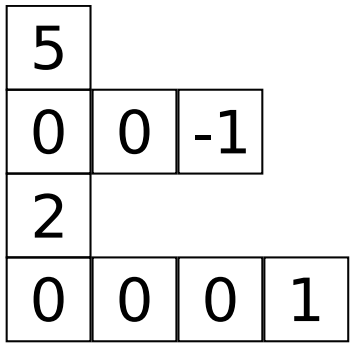
- JPEG terminology
  - Coefficient: 1 DCT-transformed DC or AC value
  - Block: 8x8 DCT coefficients (1 DC, 63 AC)
  - iMCU: Group of at least one luma and one chroma block

Encryption approach overview

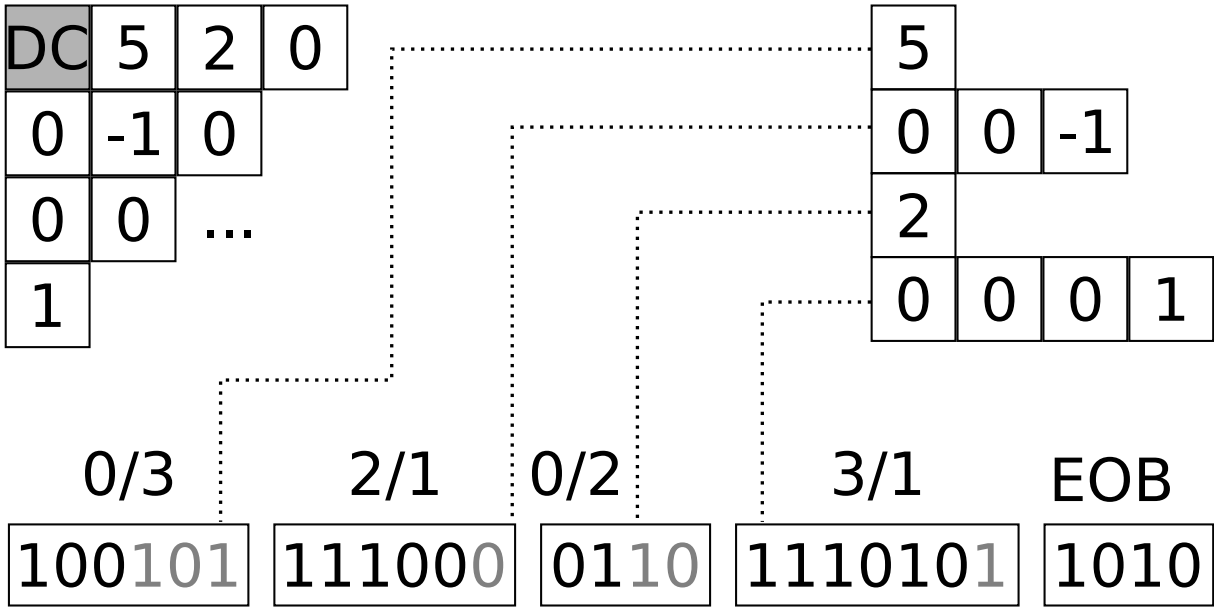
- Approach performs three operations per iMCU
  - Change order of run-length-value pairs within a block
  - Scramble value bits
  - Swap order of compatible blocks within an iMCU
- Practical considerations
  - Number of blocks per iMCU is small, but constant $\rightarrow$ spatially limited, independently encrypted areas within a picture
  - Self-information increases with number of run-length-value pairs and value bits $\rightarrow$ adaptive encryption strength
  - DC coefficients are transmitted separately as differences which are relatively easy to attack $\rightarrow$ only encrypt AC coefficients

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

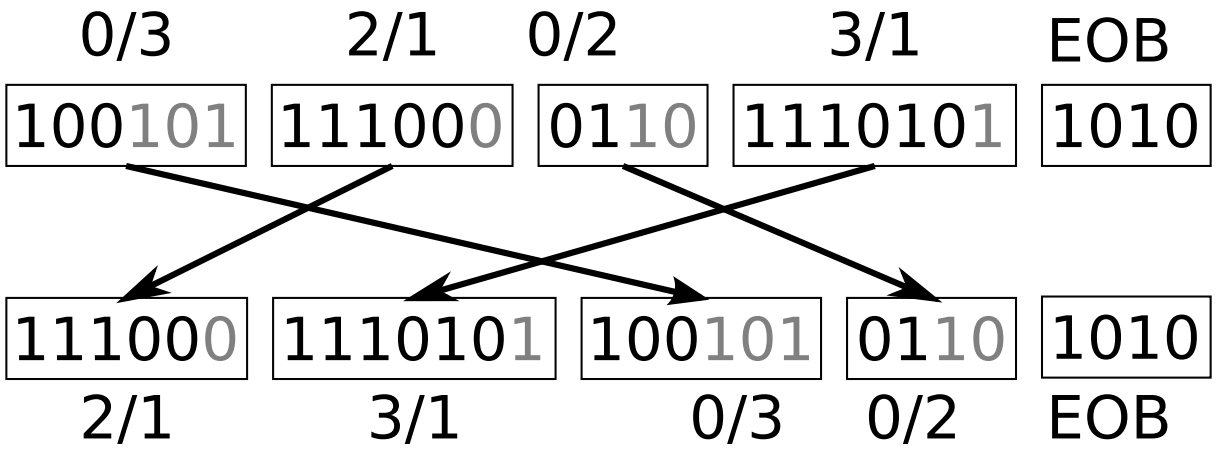




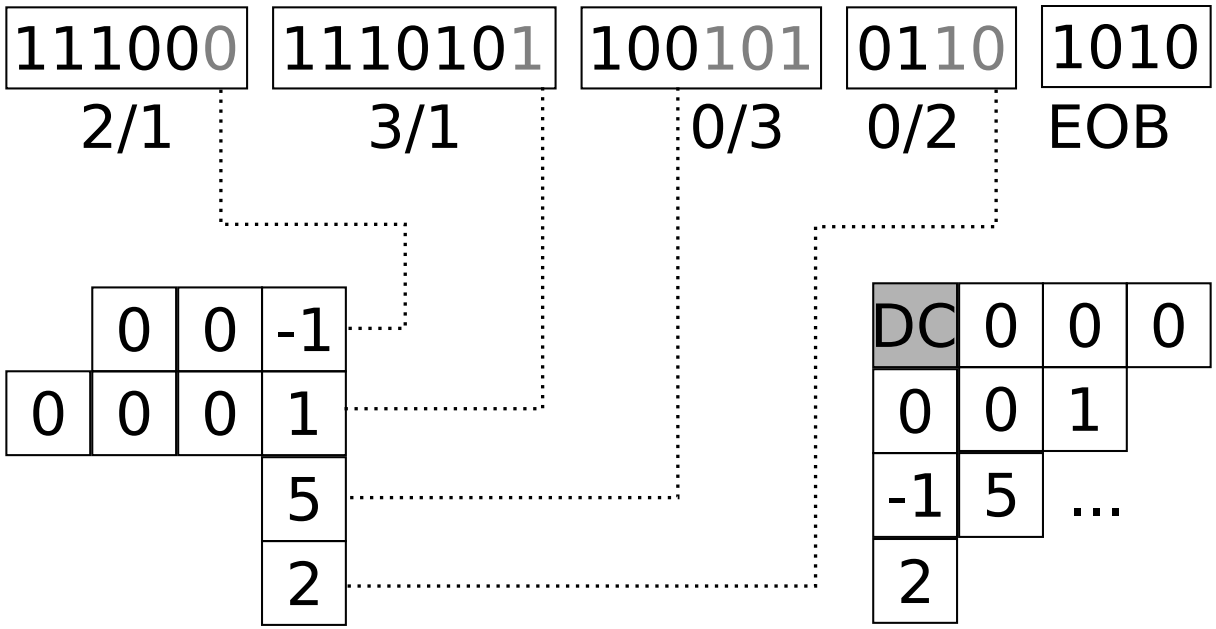
Approach part 1 – Run-length-value pair order change



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



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



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

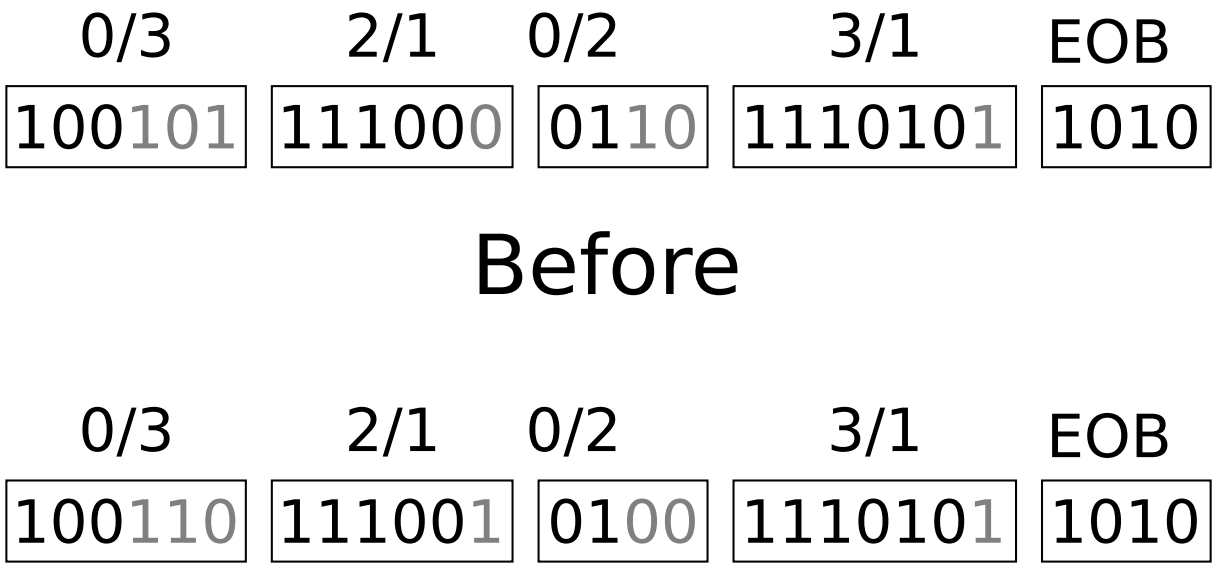
DC	5	2	0
0	-1	0	
0	0	...	
1			

Before

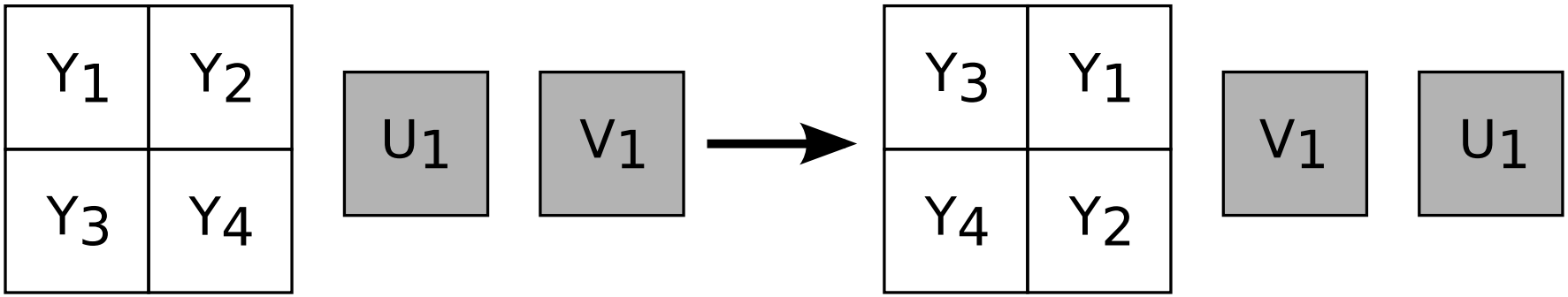
DC	0	0	0
0	0	1	
-1	5	...	
2			

After

Approach part 2 – Value bit scrambling



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
- Inter-chroma-block swapping is possible (in this example)

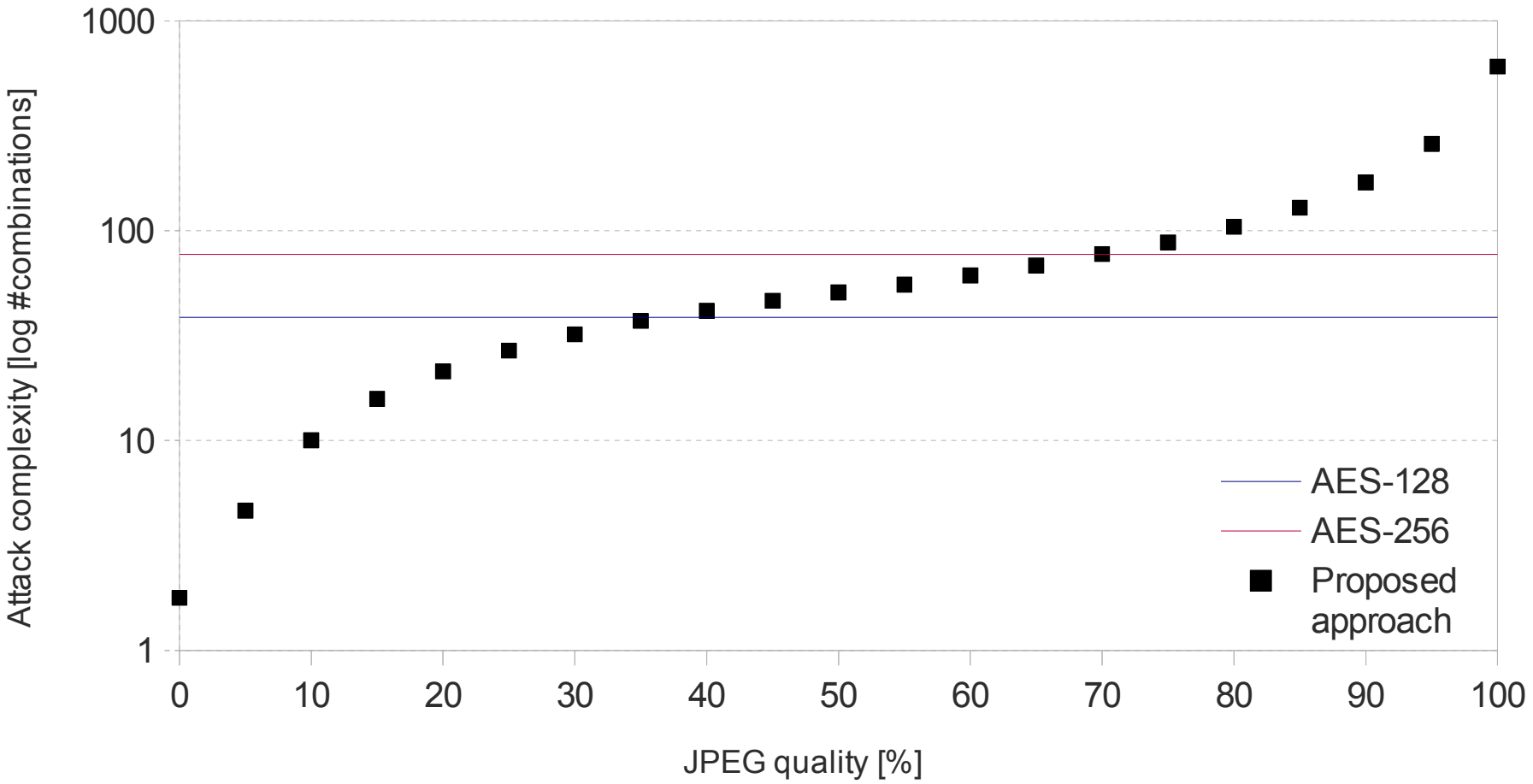
Actual encryption

- Initialize AES implementation in OFB mode (key-dependent)
- Use n "encrypted" bits as n-bit PRNG: $arand(n)$
- Permutation of n code-word-value-pairs or n blocks: swap each element at index i with element at index $arand(\lceil log_2(n) \rceil)$
- Scrambling of n value bits: XOR bits with $arand(n)$

Security assessment I

- Analysis of attack complexity (re-reordering and descrambling)
- Number of run-length-value pairs and value bits influence complexity
- 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

Security assessment II – Typical attack complexity



Future work

- Encrypt DC coefficient differences too
- Swap blocks between different iMCUs (increases attack complexity)
- Encrypt only a part of the picture (RoI)
  - Easy to do on an iMCU basis
  - Hard to signal in a length-preserving way
  - Pseudo solution for signalling: iMCU bitmap in a JPEG comment at the beginning/end of the file (not length-preserving)

Conclusion

- Encryption approach for JPEG files
  - Length-preserving
  - Format-compliant
  - Adaptive
  - Secure
- Bit-stream-based application $\rightarrow$ no recompression necessary (fast)
- Extensions like RoI encryption possible $\rightarrow$ video surveillance

Thank you for your attention!

Questions?