IOWA STATE UNIVERSITY

Detecting Mobile Steganography Using Random Tests

Seth Pierre*, Wenhao Chen*, Li Lin#, Abby Martin #, Yong Guan*, Roy Maxion@, Jennifer Newman #,*

*Iowa State University, Electrical & Computer Engineering

Iowa State University, Mathematics

@Carnegie Mellon University, Computer Science Department

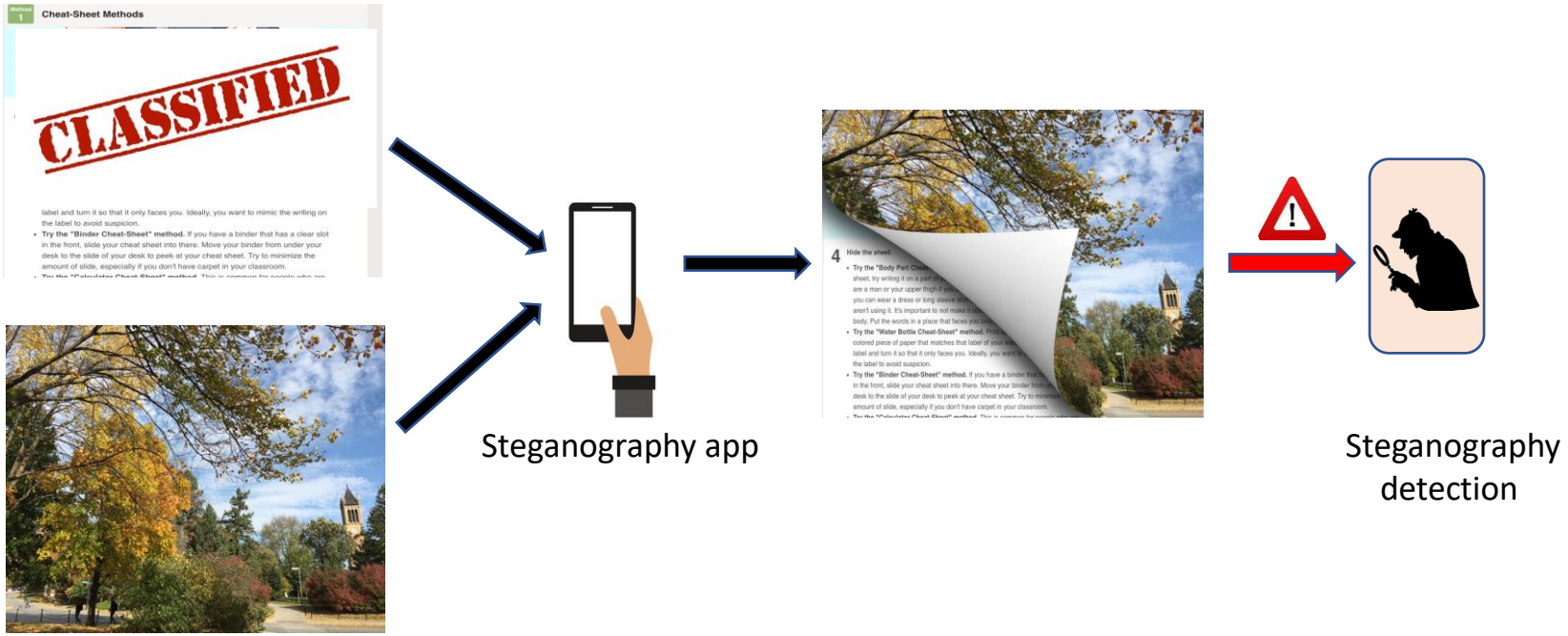
csafe
Center for Statistics and Applications in Forensic Evidence

Secret messages

- Passing messages “hidden in plain sight” has existed for millenniums
- Recall: writing in lemon juice, then using heat to expose
- This is called *steganography*: “covered” + “writing” from the Greek
- Mobile steganography apps are plentiful and easy to use but do not have effective detection tools
- All current tools rely on knowledge of the embedding process for detection of the stego image, and have shortcomings
- We developed a method that performs detection blindly with no prior knowledge of the app, under two conditions on the embedding
- We show for a limited data set that it correctly detects nearly all stego images with this type of embedding and correctly detects nearly all innocent images

What is steganography?

- Hiding a message in an innocent-looking object like an image so that the visual appearance is not distorted
- Steg detection – detection of hidden message (hard to do)



Motivation

- Current steg detection tools do not perform very effectively
  - Stego Suite (Wetstone) – uses hash tables of previously detected images
  - DC3 StegDetect – out of date
- Other tools being investigated appearing in our work at AAFS 2021
  - StegExpose
- Many tools depend on knowledge of the app itself or machine learning tools
- We propose a method to detect stego images that were created following a certain embedding method that does not require knowledge of the app itself or machine learning

If the message was embedded this way …

1. ***Spatial domain*** image like PNG, BMP, etc.
   - No jpeg or compressed images
2. Message is a ***text message*** that is embedded in a lexicographical scan order into the LSB plane
3. Message is ***not encrypted***

Then we detect it this way …

1. *Extract bits* from the LSB plane in certain *lexicographical scan or*ders
2. *Apply tests of randomness* to the bit sequence
3. If most tests indicate the bit sequence is NOT random, then the image is flagged as suspect (a possible stego image)

Why does this work?

Plaintext (message in English)

HERE ARE THE ANSWERS…

ASCII (message in ascii code)

HERE ARE THE ANSWERS…

01001000010001010101001001000101001000000100000101010 0…

- If bit sequence is not encrypted, the bit sequence is NOT random

character	ASCII
H	01001000
E	01000101
R	01010010
E	01000101
	00100000
A	01000001
R	01010010
E	01000101
	00100000
T	01010100
H	01001000
E	01000101
	00100000
A	01000001
…	…

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits ↓ b_4 b_3 b_2 b_1 ↓ Row → Column	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Scan orders

- We extract sequences using several different scan orders
- Then apply randomness tests to each sequence
- If a bit sequence is not random, it is very likely a text message in ASCII
- Only the scan orders that produces a bit sequence that is flagged as not random is needed to indicate the image is stego
- Then, the ASCII code can be applied to that bit sequence to decode

- NIST has a suite of tests that give a probability of a bit sequence being random

NIST Test Suite - tests for random bit sequences

- NIST'S *Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*
- Used to measure the randomness of a bit sequence generated by pseudorandom number generators
- We use 5 of the many tests available in the NIST code
- Given a sequence of bits, each test produces a *p* value between 0 and 1
- Higher value for *p* indicates more likely the bit sequence is random
- So, if *p* is below a small threshold it means the bit sequence is not random
- We chose $p = 0.05$

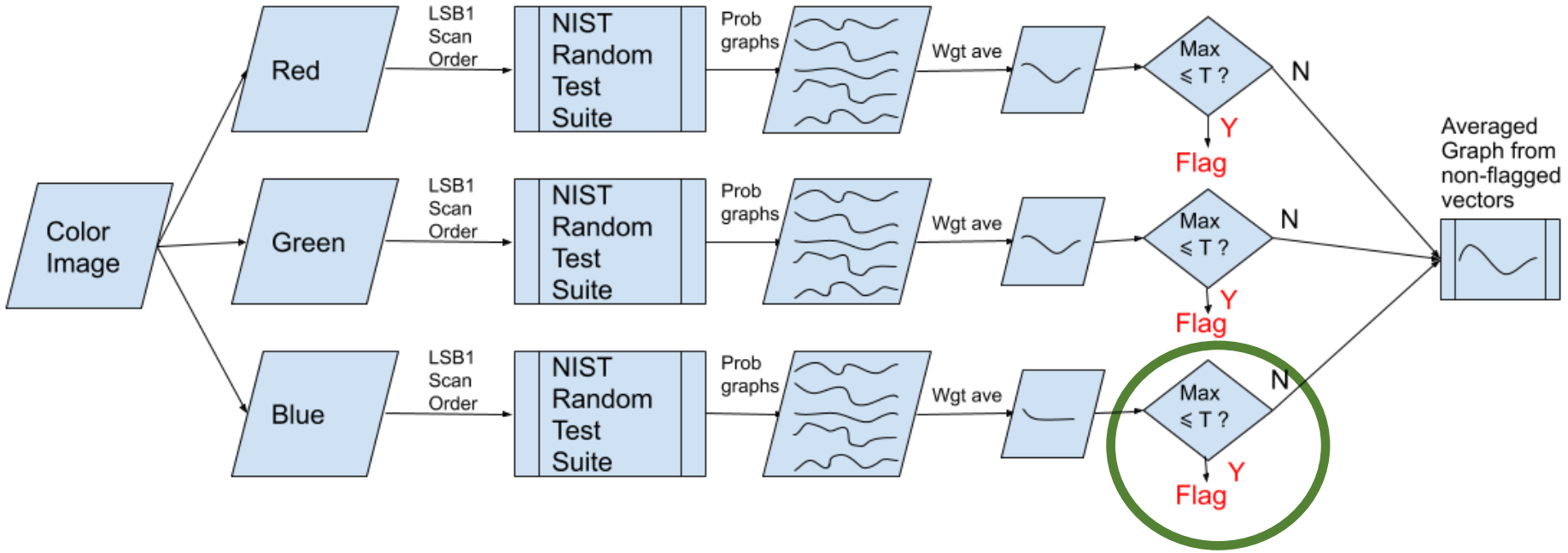
Why use NIST's Test Suite?

- The input to the NIST test is a sequence of bits or bitstream
- NIST's test suite is generally accepted as a good measure of the randomness of a sequence of bits
- This method does not use any knowledge of embedding algorithm, app, etc. to perform the test
- This method can be used to test any image for stego even apps that are not known yet to be stego apps
- This method is not a targeted detection for a specific app, only for an unencrypted text message that is embedded in a certain order in the pixels

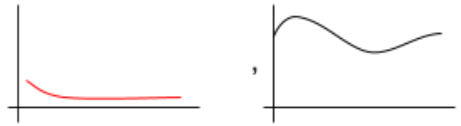
Our algorithm

- Scan each LSB plane of each color channel to produce a bit sequence
  - Use 2 scan directions: Horizontal and vertical
- For one sequence of bits, put successively larger chunks - from 1%, 2%, …, 100% - into each test
- Each test produces a vector with 100 values, and each value is the output *p value*
- Combine all 5 vectors from the 5 tests into one vector (weighted average)
- Check the maximum of the weighted average vector: if $\max \leq 0.05$, then NOT RANDOM, otherwise random
- If NOT RANDOM, flag that color channel and scan order
- OUTPUT: all flagged *p*-graphs, random graph from remaining vectors

Statistical Steg Test Algorithm Workflow



Output = {All Flagged graphs, averaged graph} =



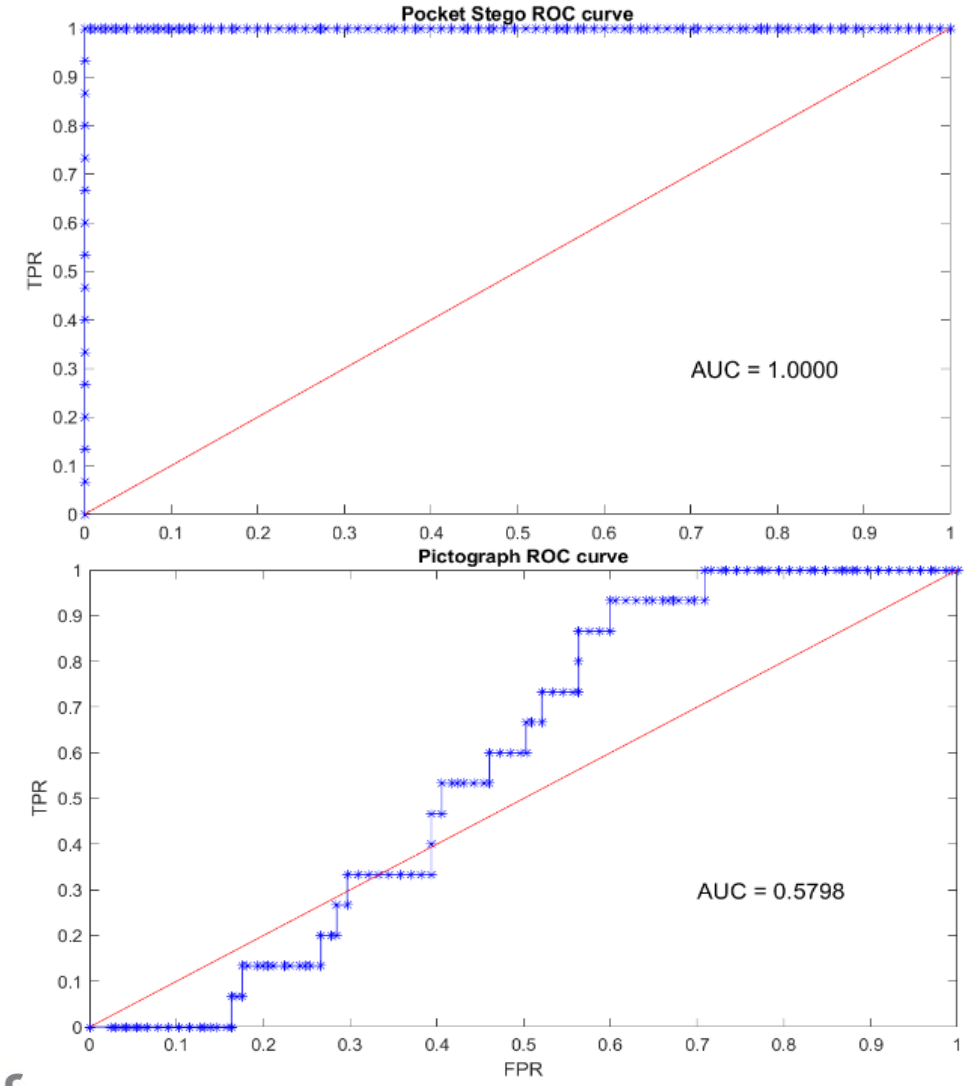
- Output provides flagged (non-random) and remaining (random)

Data

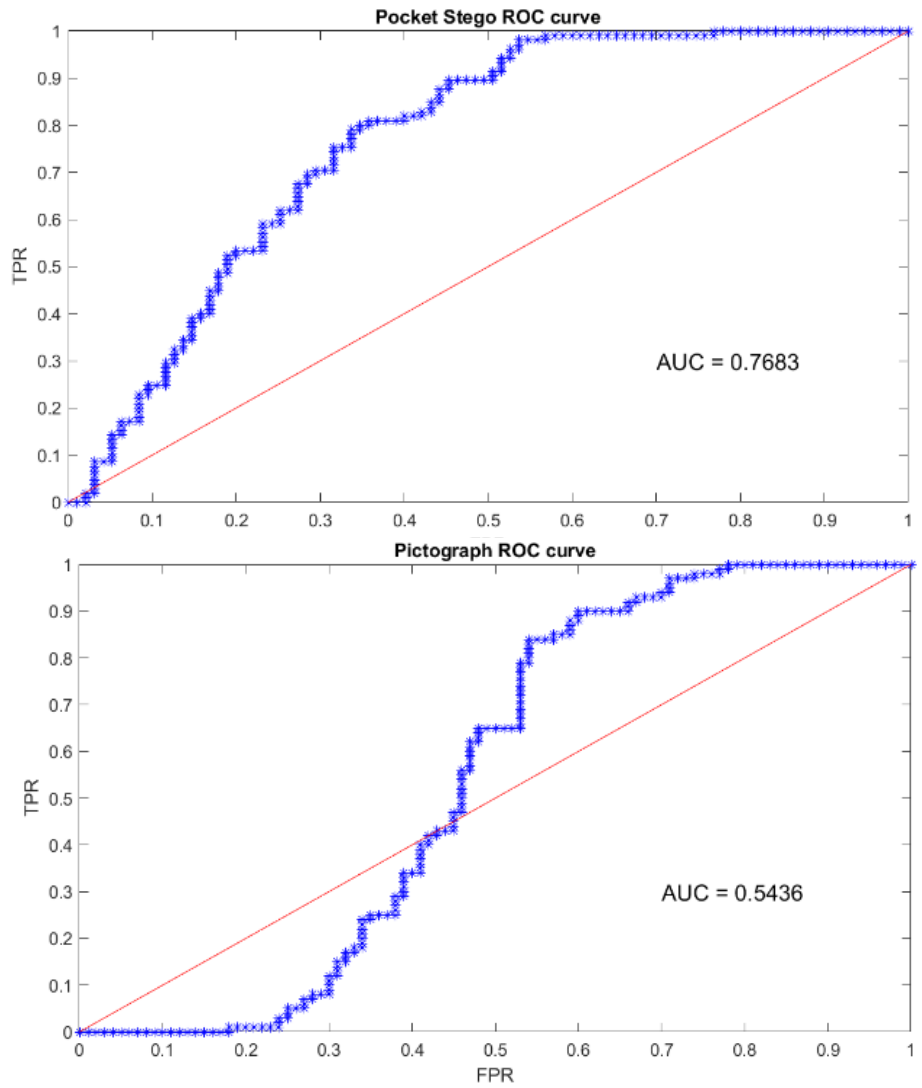
- Applied to 120 images that were created using 4 mobile apps from data in the forensic digital image database StegoAppDB from CSAFE
- In principle, would work on any stego image with similar properties
- Can be applied to any suspicious image

Results

NIST ROC curve

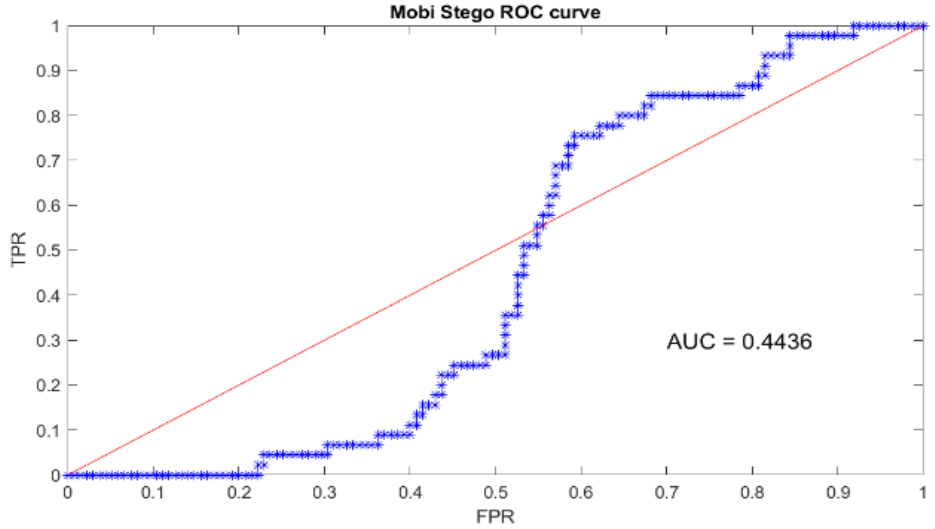


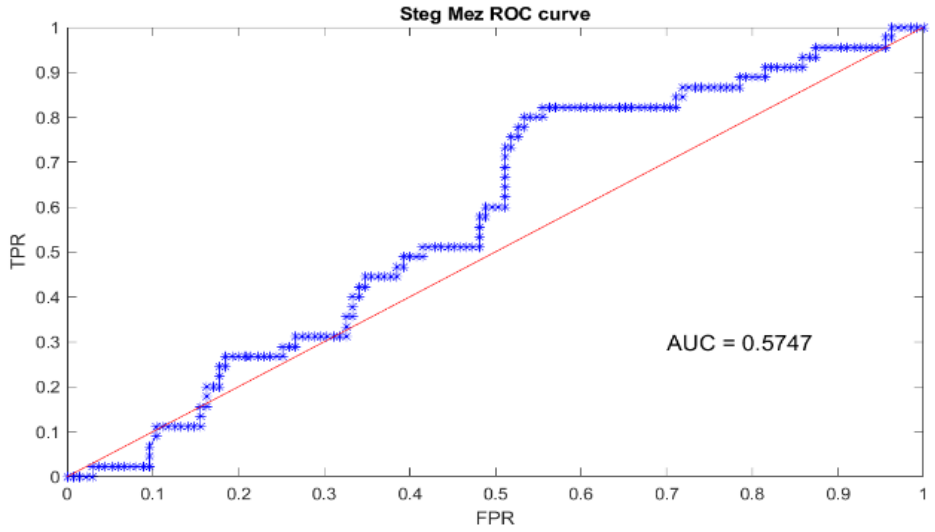
StegExpose ROC curve



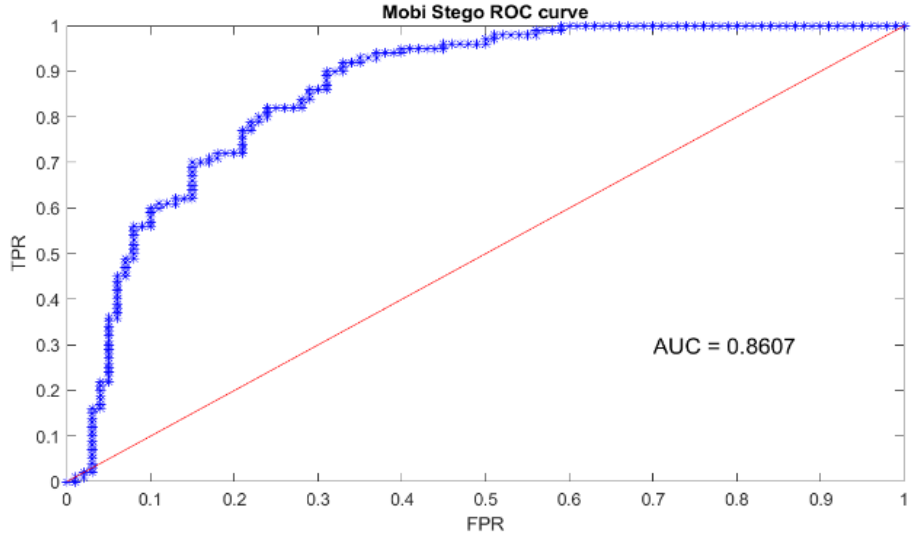
Results

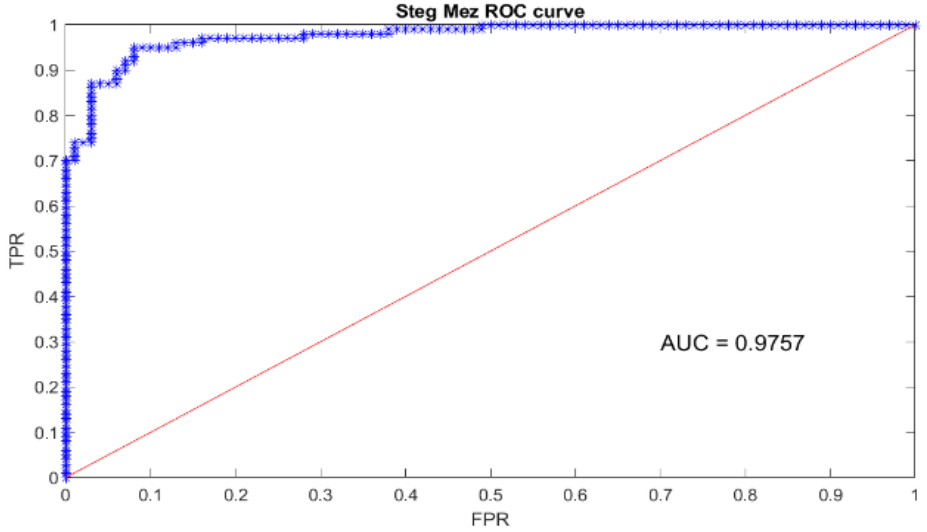
NIST ROC curve





StegExpose ROC curve





Conclusions and future work

- We have shown that a simple test for randomness applied to bit sequences, has a high detect rate of unencrypted text messages hidden in images that follow the lexicographical order
- We expect to apply this method to many more images
- We expect to implement different scan orders
- We expect to include this as a tool in a CSAFE Steg Detection Tool in the future

Acknowledgements

- This work was partially funded by the Center for Statistics and Applications in Forensic Evidence (CSAFE) through Cooperative Agreement #70NANB15H176 between NIST and Iowa State University, which includes activities carried out at Carnegie Mellon University, University of California Irvine, and University of Virginia.