RSAConference2016

San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: HT-R03

Linguistic Passphrase Cracking

Peder Sparell
M.Sc. GPEN CHFI
Simovits Consulting
peder.sparell@simovits.com

Mikael Simovits
M.Sc. CISSP
Simovits Consulting (CEO)
mikael@simovits.com

#RSAC

Background

Implementation

Objectives and entropy calculations

Results

Background

- Standards requirements/recommendations on long *and* complex/random passwords (e.g. within ISO27000 series and NIST Electronic Authentication Guideline)
- Increasing use of phrases as passwords
- Publicly available cracking software are not very effective on longer passwords or phrases > 2 words
- Languages are relatively predictable – not random
- Linguistically correct phrases - more effective cracking?
- How can language be modelled to generate/crack such phrases?
- Is it advisable to base a password policy on phrases?

Passwords – search scope/complexity

- Number of possible combinations: $c = bl$
- Sometimes written as: $2^{log_2(c)}$

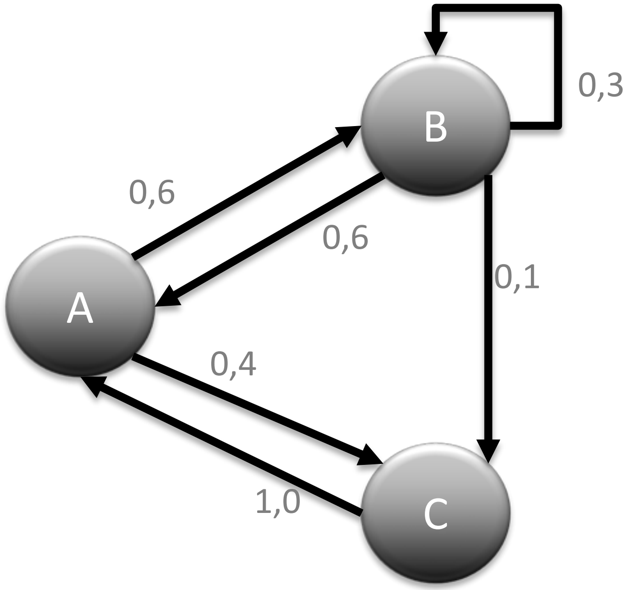
Password length	Lowercase	Lower + uppercase	Alfanumerical	Alfanumerical + special chars
1 (base)	26	52	62	95
6	$2^{28,2}$ (=3,1*10^{8})	$2^{34,2}$ (=2,0*10^{10})	$2^{35,7}$ (=5,7*10^{10})	$2^{39,4}$ (=7,4*10^{11})
8	$2^{37,6}$ (=2,1*10^{11})	$2^{45,6}$ (=5,3*10^{13})	$2^{47,6}$ (=2,2*10^{14})	$2^{52,6}$ (=6,6*10^{15})
10	$2^{47,0}$ (=1,4*10^{14})	$2^{57,0}$ (=1,4*10^{17})	$2^{59,5}$ (=8,4*10^{17})	$2^{65,7}$ (=6,0*10^{19})
14	$2^{65,8}$ (=6,5*10^{19})	$2^{79,8}$ (=1,1*10^{24})	$2^{83,4}$ (=1,2*10^{25})	$2^{92,0}$ (=4,9*10^{27})
16	$2^{75,2}$ (=4,3*10^{22})	$2^{91,2}$ (=2,9*10^{27})	$2^{95,3}$ (=4,8*10^{28})	2^{105} (=4,4*10^{31})
20	$2^{94,0}$ (=2,0*10^{28})	2^{114} (=2,1*10^{34})	2^{119} (=7,0*10^{35})	2^{131} (=3,6*10^{39})

This works delimitations on passphrases

- 2 or more words (usually >=3)
- Assembled to a string without white spaces
- Lower case
- Examples:
  - The king shall rule -> thekingshallrule
  - My brother rocks -> mybrotherrocks

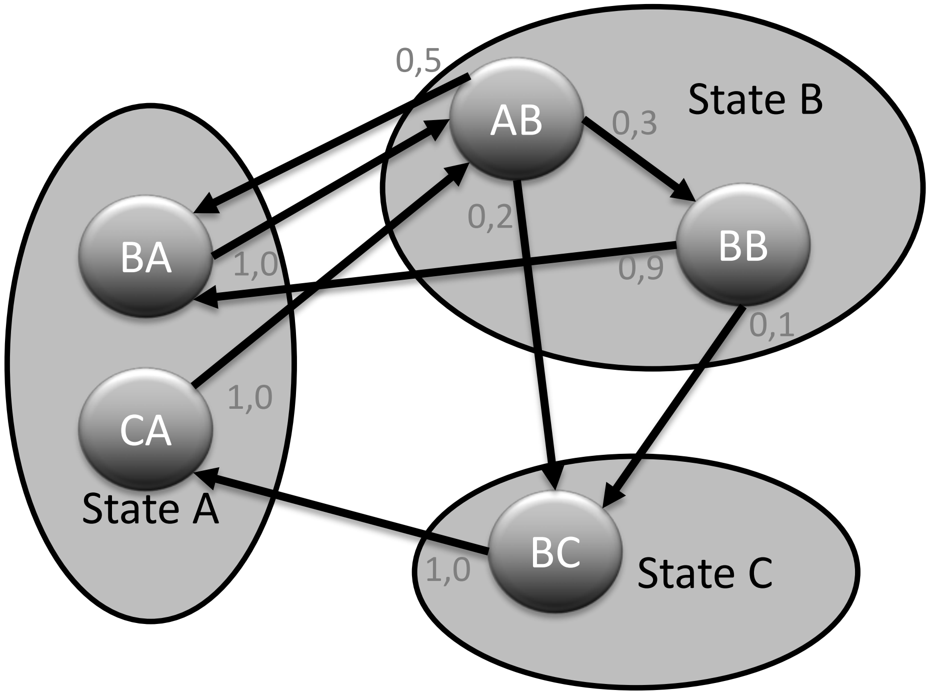
Markov chains

- A Markov process is a process where transitions to other states are determined by probability distribution based on the current state
- A Markov chain is the resulting sequence of states from a Markov process



Markov chains of order *m*

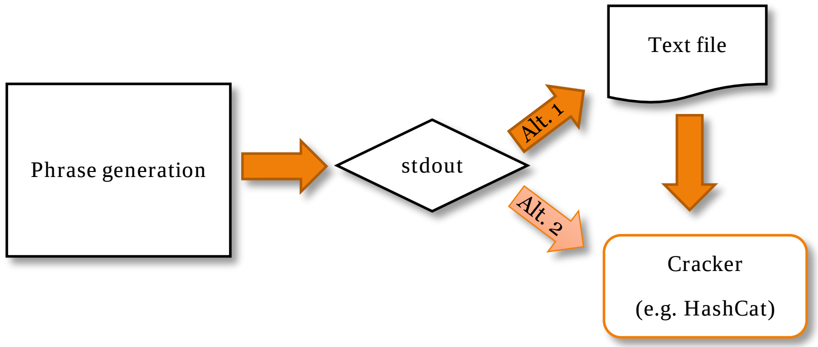
- A Markov chain of order m redefines the states to include a ’history’ of *m* states



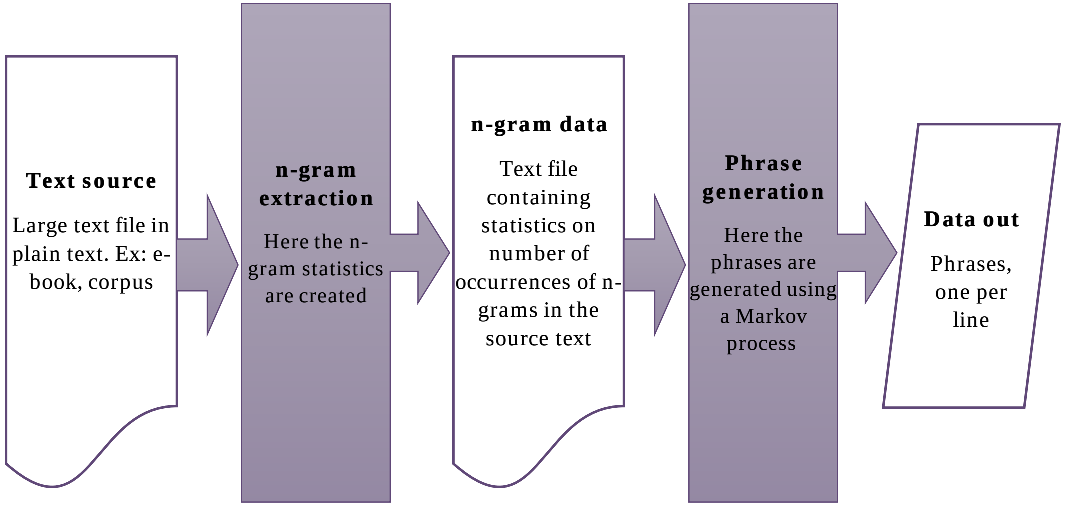
n-grams

- n-grams are used for language modelling, using Markov chains of order *n-1*
- n-grams are sequences containing *n* elements
- The elements could be characters **or words**
- Statistics for the probability distribution is generated from counting number of occurrences of n-grams in large texts
- Example: Statistics of 3-grams shows, for example, if character ’e’ is more likely than character ’x’ to follow a current state of ’th’

Implementation - Overview



Implementation - Phrase generation


Text source
Large text file in plain text. Ex: e-book, corpus
n-gram extraction
Here the n-gram statistics are created
n-gram data
Text file containing statistics on number of occurrences of n-grams in the source text
Phrase generation
Here the phrases are generated using a Markov process
Data out
Phrases, one per line

Phase 1: n-gram extraction

- At start-up, a text file to analyse is chosen, as well as the desired order and level of n-grams
- Uses regular expressions
- Punctuation marks (.,!?:) are replaced by a single dot (.) to represent sentence breaks
- All characters are changed to lower case
- Result: n-grams are saved to a text file together with their number of occurrences
- Examples (char/word):

```
…
arw 3
ary 137
as. 24
as_ 2382
asa 48
asb 7
asc 42
asd 7
ase 207
…
```

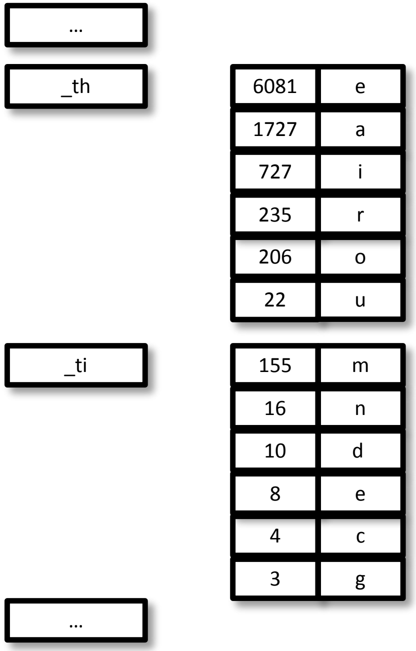
```
…
the shelbyville runner 1
the sheldon penny 1
the shelf . 10
the shelf adaptors 1
the shelf and 6
the shelf are 1
the shelf at 3
…
```

Simovits Consulting

RSAConference2016

Phase 2: Phrase generation

- Interval of desired phrase lengths can be set
- Number of desired words can be set
- The desired n-gram file is read and loaded to memory as in the example in the picture
- Starting states (starting with '.') are loaded into a separate list
- A threshold value decides if n-gram should be loaded to the list or ignored
- From every starting state the phrases are built up char by char (or word by word), by recursively going through the possible state transitions and adding the new chars (or words) to the current phrase until it is long enough



Entropy of the English language

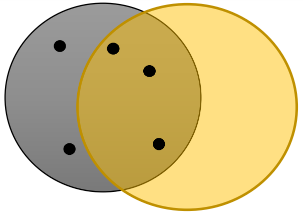
- Different estimation suggestions:
  - 1,75 bits/char
  - 1,6 bits/char
- NIST (National Institute of Standards & Technology) suggests:
  - First char: 4 bits
  - Char 2-8: 2 bits
  - Char 9-20: 1,5 bits
  - Char 21 and subsequent: 1 bit
  - Example:
    A 16 character phrase of the English language has (1*4)+(7*2)+(8*1,5) = 30 bits of entropy

Comparison entropy calculations

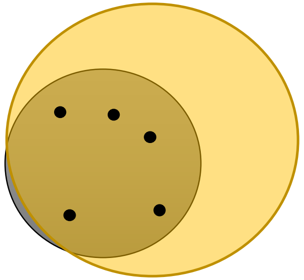
For every generated list of phrases we define:

- **Target entropy**: H_t is calculated according to NIST:s interpretation of the entropy of the English language
- **Potential entropy**: $H_p = log_2(X)$
  - X is the number of phrases in the list
- **Estimated entropy**:
  - Efficiency: $e = \frac{cracked\ hashes}{sought\ hashes}$
  - $H_{est} = log_2\left(\frac{X}{e}\right) = log_2(X) - log_2(e)$
- The objective is that the “potential” and especially the “estimated” entropy values are as close as possible to the “target” entropy.

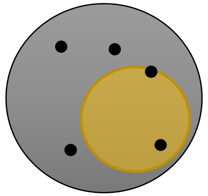
Some case Illustrations



Enough number of phrases in list.
About half of them are not valid phrases.
Potential entropy is close to Target entropy, Estimated entropy is higher.



Large number of phrases in list.
Covers the whole language, but many non valid phrases.
Potential and Estimated entropy high.



Small number of phrases in list.
All of them are valid phrases, but they do not cover the language.
Potential entropy lower than Target entropy, Estimated entropy higher.

RSAConference2016

#RSAC

Background

Implementation

Objectives and entropy calculations

Results

Testing variables

- Test sample basis contained of 66 hashes from passwords of length 10-20
- n-gram statistics were generated from 3 different source texts
  - English news sites (3 million sentences)
  - English Wikipedia (1 million sentences)
  - General web sites/blogs etc. (1 million sentences)
- The input data values used for each list can be derived from the selected file names. Example: ’**L20W6T5N3WNews**’
  - length of the phrases in the list/file is 20 (L20)
  - phrases consists of exactly 6 words (W6)
  - the threshold value is set to 5 (T5)
  - n-grams of order 3 has been used (N3)
  - n-grams on the word level has been used (C=char, W=words)
  - text source is the one from news sites (Wiki, Web or News)

Results details – Phrase length 10

Phrase List	Time generation	Time HashCat	Cracked hashes	Possible outcomes (English)	Phrases Generated	Target entropy	Potential entropy	Estimated entropy
L10T100N5CNews	4.2 h	46 s	7/15	2^{21} (2.1 mil.)	$2^{29.6}$ (825 mil.)	21	29.6	30.7
L10T0N3WNews	1.5 h	1 s	2/15		$2^{23.9}$ (15.2 mil.)		23.9	26.8
L10T1N3WNews	16 min	0 s	2/15		$2^{21.3}$ (2.6 mil.)		21.3	24.2

Results details – Phrase length 14

Phrase List	Time generation	Time HashCat	Cracked hashes	Possible outcomes (English)	Phrases Generated	Target entropy	Potential Entropy	Estimated entropy
L14T3000N5CNews	1.7 h	6 s	0/15	2^{27} (134 mil.)	$2^{26.4}$ (90 mil.)	27	26.4	-
L14T1N8CWiki	23.0 h	4 min 32s	2/15		$2^{30.8}$ (1865 mil.)		30.8	33.7
L14T1N3WNews	20.2 h	31 s	2/15		$2^{28.9}$ (505 mil.)		28.9	31.8
L14W-5T0N3WNews	24.0 h	45 s	2/15		$2^{28.2}$ (312 mil.)		28.2	31.1

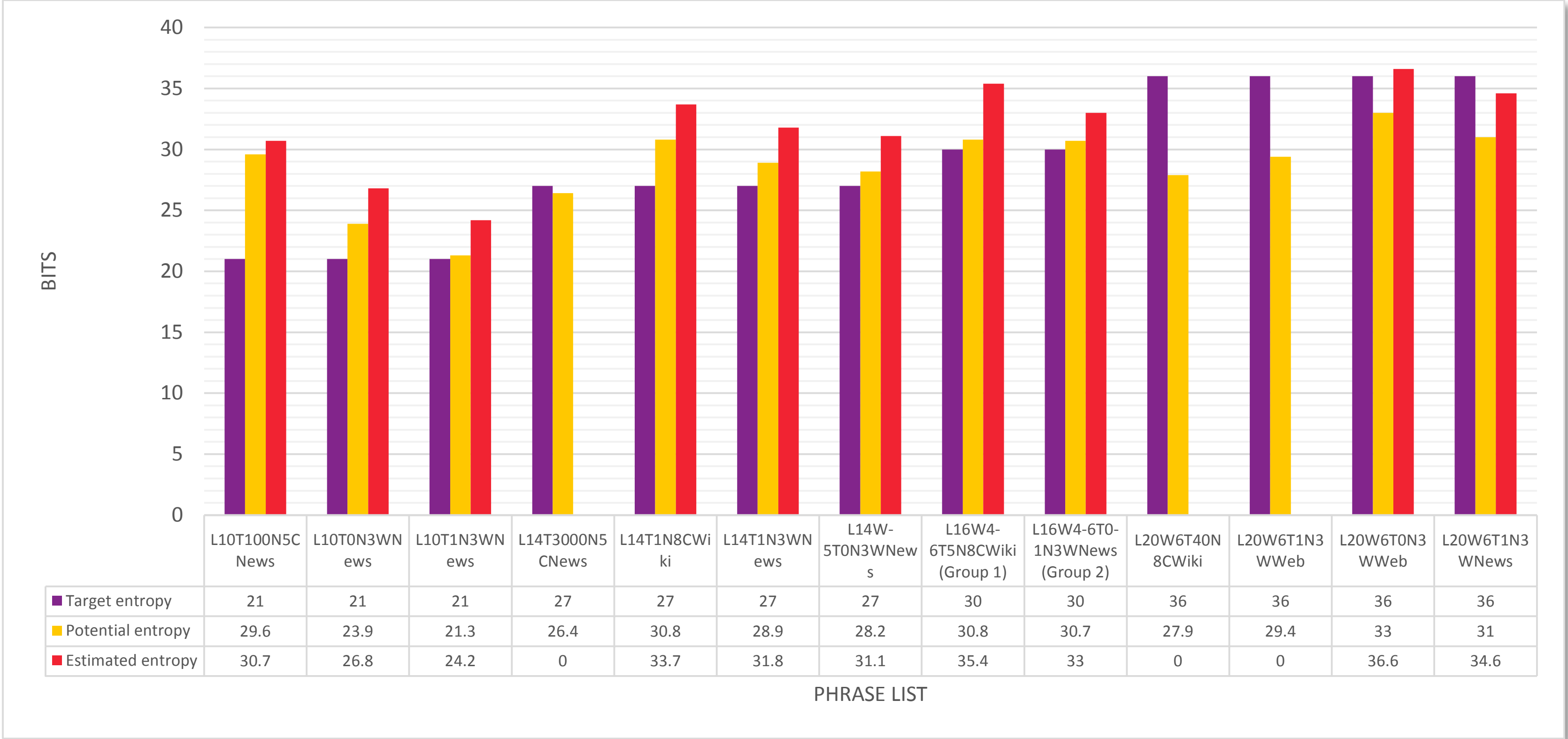
Results details – Phrase length 16

Phrase List	Time generation	Time HashCat	Cracked hashes	Possible outcomes (English)	Phrases Generated	Target entropy	Potential Entropy	Estimated entropy
L16W4T5N8CWiki	7.3 h	31 s	1		$2^{28.8}$ (479 mil.)			
L16W5-6T5N8CWiki	34.7 h	3 min 37 s	0		$2^{30.3}$ (1 355 mil.)			
Total, group 1	**42 h**	**4 min 8 s**	**1/24**	**$<2^{30}$ (<1 100 mil.)**	**$2^{30.8}$ (1 834 mil.)**	**<30**	**30.8**	**35.4**
L16W4T0N3WNews	9.2 h	6 s	4		$2^{26.0}$ (67.5 mil.)			
L16W5T0N3WNews	79.5 h	10min 16 s	1		$2^{29.7}$ (887 mil.)			
L16W6T1N3WNews	58.7 h	1 min 32 s	0		$2^{29.7}$ (851 mil.)			
Total, group 2	**147.4 h**	**11min 54 s**	**5/24**	**$<2^{30}$ (<1 100 mil.)**	**$2^{30.7}$ (1 806 mil.)**	**<30**	**30.7**	**33.0**
L16W5T0N3WWeb	16.6 h	13 s	1		$2^{27.6}$ (199 mil.)			

Results details – Phrase length 20

Phrase List	Time generation	Time HashCat	Cracked hashes	Possible outcomes (English)	Phrases Generated	Target entropy	Potential Entropy	Estimated entropy
L20W6T40N8CWiki	12.8 h	18 s	0/12	$<2^{36}$ (<69 000 mil.)	$2^{27.9}$ (244 mil.)	<36	27.9	-
L20W6T1N3WWeb	52.2 h	2 min 14 s	0/12		$2^{29.4}$ (727 mil.)		29.4	-
L20W6T0N3WWeb	960 h	29 min 9 s	1/12		$2^{33.0}$ (8 500 mil.)		33.0	36.6
L20W6T1N3WNews	550.5 h	39 min 3 s	1/12		$2^{31.0}$ (2 131 mil.)		31.0	34.6

Phrase list efficiency

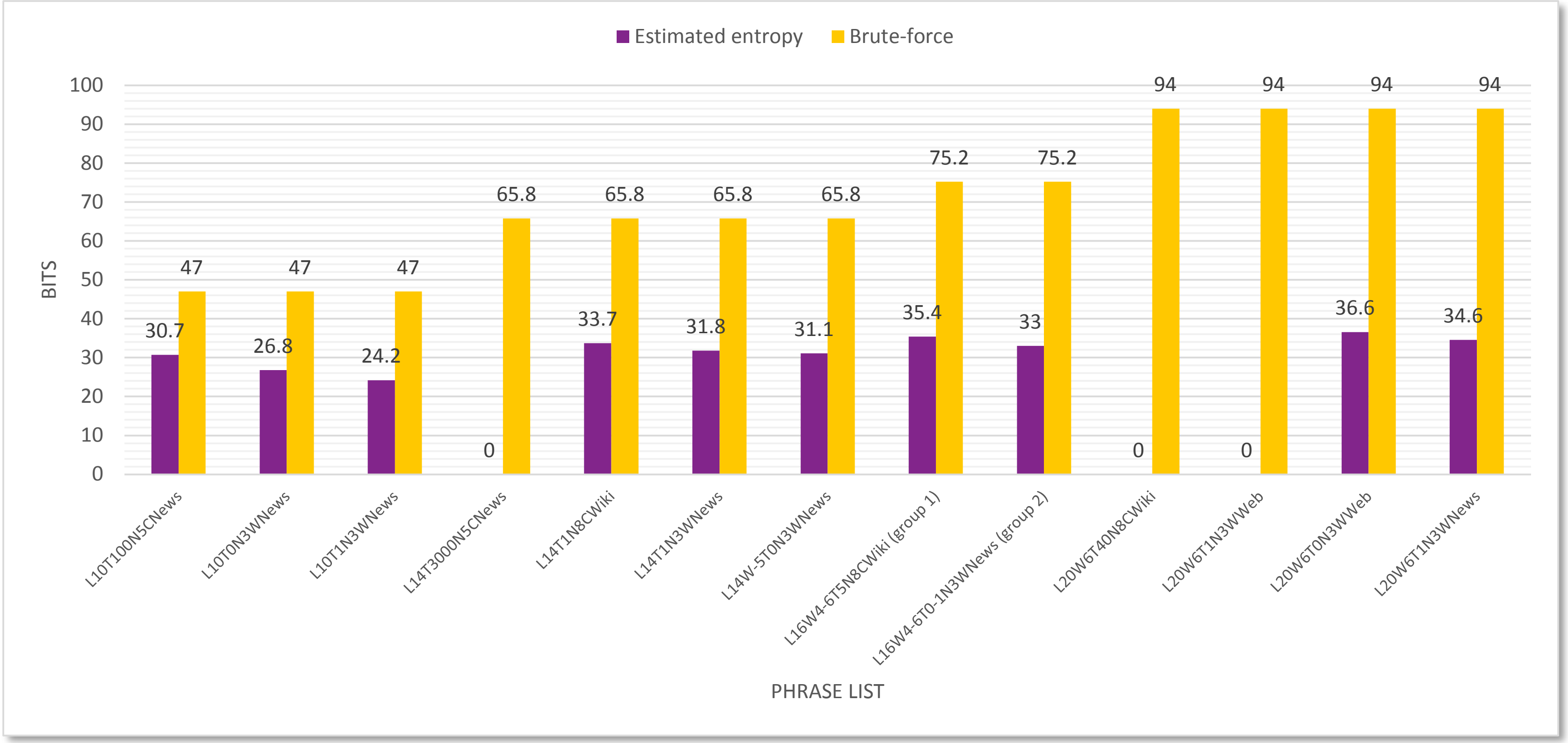


	L10T100N5CNews	L10T0N3WNews	L10T1N3WNews	L14T3000N5CNews	L14T1N8CWiki	L14T1N3WNews	L14W-5T0N3WNews	L16W4-6T5N8CWiki (Group 1)	L16W4-6T0-1N3WNews (Group 2)	L20W6T40N8CWiki	L20W6T1N3WWeb	L20W6T0N3WWeb	L20W6T1N3WNews
🟪 Target entropy	21	21	21	27	27	27	27	30	30	36	36	36	36
🟨 Potential entropy	29.6	23.9	21.3	26.4	30.8	28.9	28.2	30.8	30.7	27.9	29.4	33	31
🟥 Estimated entropy	30.7	26.8	24.2	0	33.7	31.8	31.1	35.4	33	0	0	36.6	34.6

Simovits Consulting

RSAConference2016

Brute force comparison



Simovits Consulting

RSAConference2016

LinkedIn cracking

- Leaked hashes from LinkedIn 2012
- Using the previously generated phrase lists
- This was not the main objective of this work, so lists are not optimized
- Estimates of the target scope (number of crackable passwords) of each length, based on passwords statistics on already cracked passwords
(www.adeptus-mechanicus.com)
- The target scope below still includes many words beyond the original target scope of this work (non -linguistically correct phrases)

Phrase length	Cracked hashes	Target scope *(estimated total no. of lowercase passwords)*
10	18 269 (11%)	168 000
14	1 882 (10%)	18 300
16	612 (10%)	6 100
20	6 (9%)	68

Simovits Consulting

Conclusions

- Good results compared to brute-force
- Efficient – and can be further improved
- Lack of alternative publicly available methods
- Cracks about 10% of really long passwords
- Time-memory trade-off
- If a password policy is based on phrases, it should at least also require
  - Phrases longer than 20 characters
  - Characters from all character sets (alphanumeric + special characters)

Future work & more info

Future work

- Optimization of performance of the implementation
- Include upper case, numbers and special characters
- Etc.

More info

www.simovits.com

peder.sparell@simovits.com

“Apply” slide

- After this session we hope that you will know that passwords as a mean of authentication is more or less obsolete.
- It is possible to implement linguistic password cracking for long phrases and achieve viable results in a short time.
- You know where to begin if you want to carry this work further:
  - Source code available on Github: https://github.com/Sparell/Phraser

Thanks

peder.sparell@simovits.com

mikael@simovits.com

Simovits Consulting