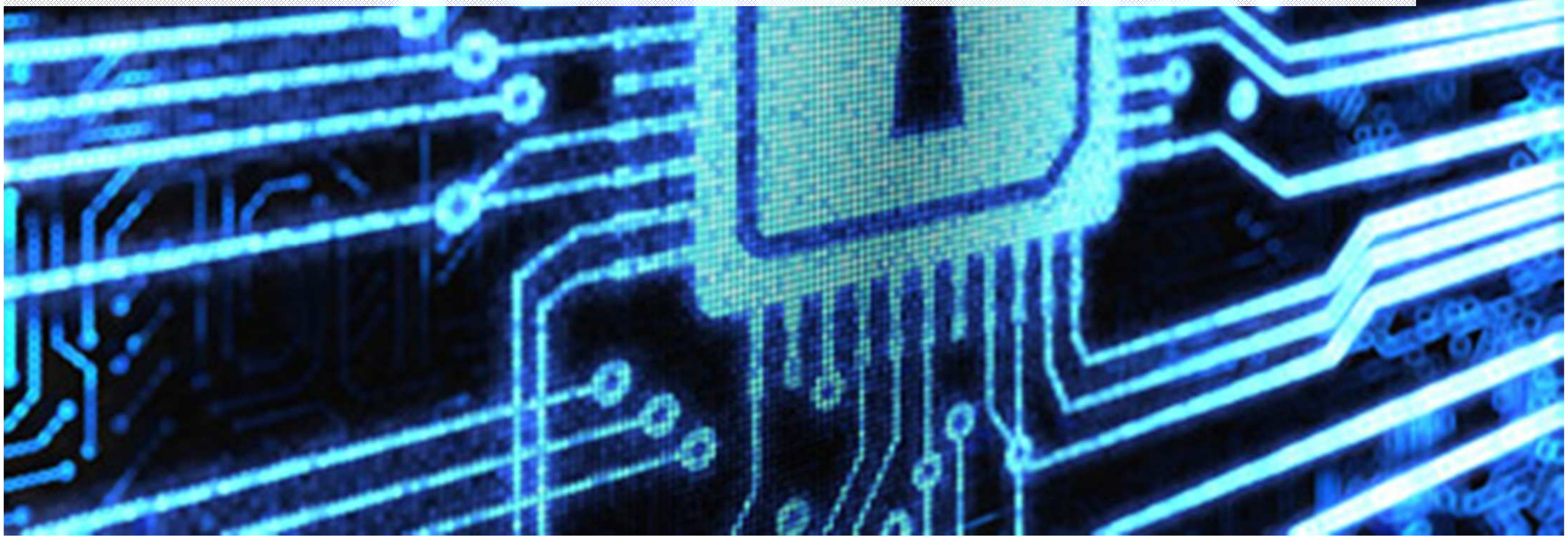
RUHR-UNIVERSITÄT BOCHUM

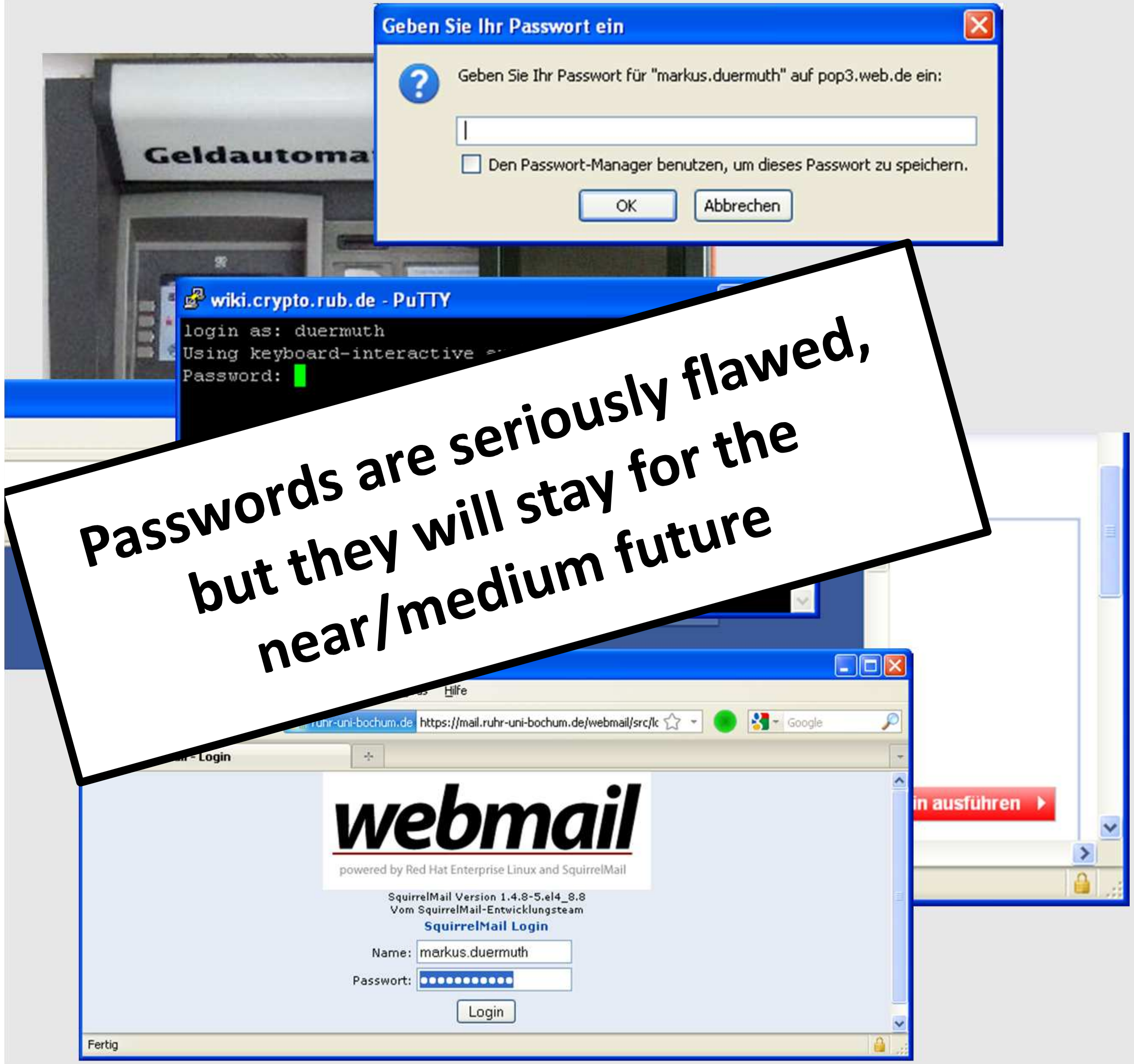
Password Security and Markov Models

Markus Dürmuth
Horst Görtz Institute for IT-Security
Ruhr-University Bochum



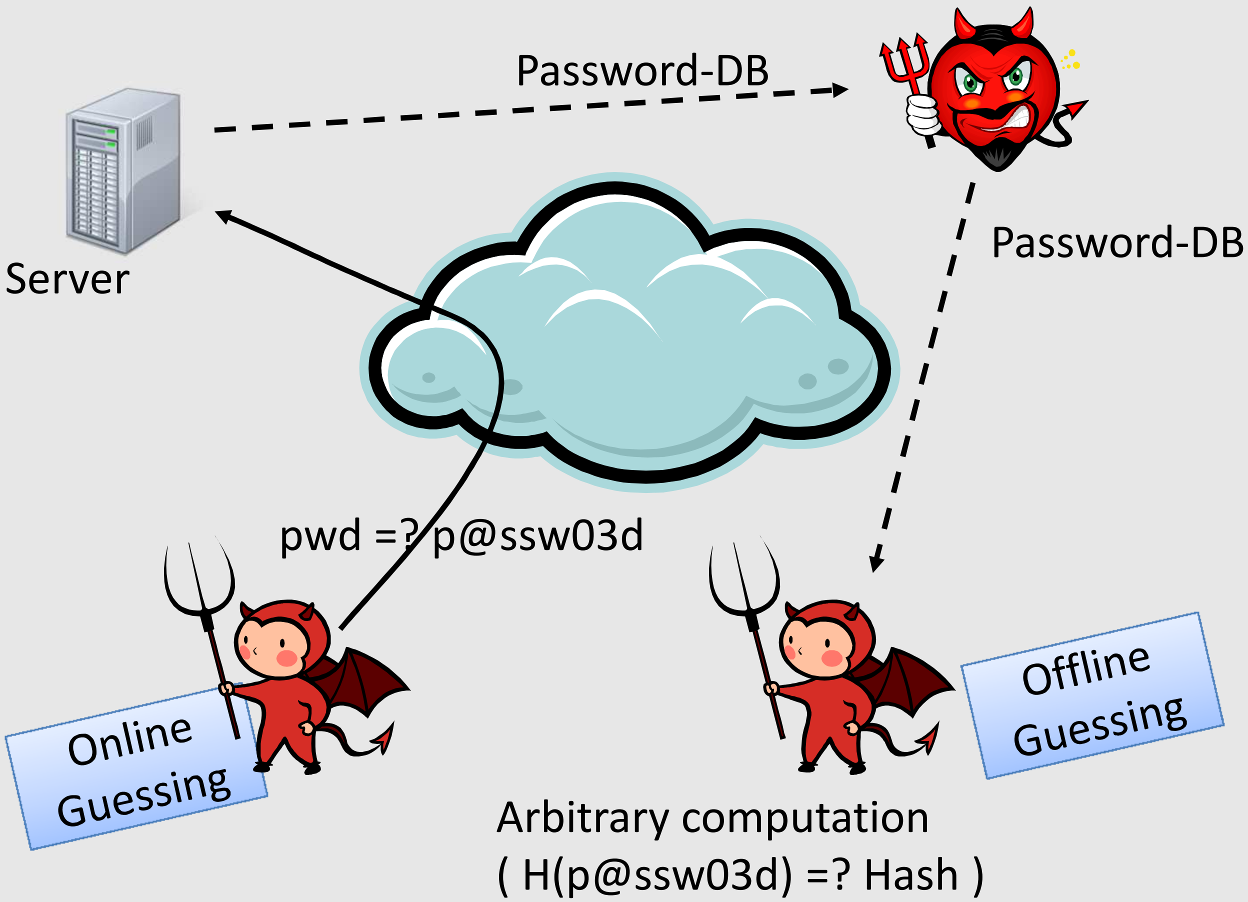
RUB

User authentication with passwords



Passwords are seriously flawed, but they will stay for the near/medium future

Online vs offline guessing



1. **Markov models**
2. **Password Guessing** (with Markov models)
3. **Measuring password strength** (with Markov models)
4. **Personal information and password guessing** (and Markov models)

Markov models 101

- Goal: Guess passwords in order of decreasing likelihood

  => Estimate password probabilities

- Idea: Estimate probabilities from real password data (e.g. RockYou list)…
- …but data is limited

- E.g.: RockYou list:
  - <bobby1998> (substantially) less likely than <bobby1993>?
  - (Probably) NO!

- Need a way to generalize the observations

#	password
2	bobby1999
0	bobby1998
3	bobby1997
1	bobby1996
3	bobby1995
5	bobby1994
3	bobby1993
3	bobby1992
2	bobby1991
1	bobby1990

Markov models 101

- Idea 2: Reduce the space that needs to be learned

$$P(\text{ passwd }) =$$
$$P(\text{ p }) \cdot P(\text{ a } | \text{p }) \cdot P(\text{ s } | \text{ pa }) \cdot$$
$$\cdot P(\text{ s } | \text{ pas }) \cdot P(\text{ w } | \text{ pass }) \cdot P(\text{ d } | \text{ passw })$$

- Not really helpful, but…
- …Markov assumption: these conditional probabilities can be approximated by a **short history**, e.g., for 3-grams (history 2):

$$P(\text{ passwd }) =$$
$$P(\text{ pa }) \cdot P(\text{ s } | \text{ pa }) \cdot$$
$$\cdot P(\text{ s } | \text{ as }) \cdot P(\text{ w } | \text{ ss }) \cdot P(\text{ d } | \text{ sw })$$

- …and these 3-grams are easier to learn (!)

Markov models 101

- In general:

$$P(c_1, \dots, c_k) = P(c_i | c_1, \dots, c_n) \prod_{i=n}^{k} P(c_i | c_{i-n+1}, \dots, c_{i-1})$$

- Estimate the conditional probabilities from frequencies

$$P(c_i | c_{i-k+1}, \dots, c_{i-1}) = \frac{count(c_{i-k+1}, \dots, c_{i-1}, c_i)}{count(c_{i-k+1}, \dots, c_i)}$$

- For example

$$P(w|pass) = \frac{count(passw)}{count(pass*)} = \frac{97963}{114218} = 0.86$$

- Better estimation uses “smoothing”
(we are currently implementing/testing this)

Markov models 101

[RockYou password list]

- $P^{3-gram}(\text{bobby1998}) \approx P^{3-gram}(\text{bobby1997}) \approx P^{3-gram}(\text{bobby1996})$

#	3-gram
49994	bob
27698	obb
42105	bby
33025	by1
37238	y19
374503	199
31974	998
34095	997
47124	996
58307	995

1. Markov models

2. Password Guessing (with Markov models)

3. Measuring password strength (with Markov models)

4. Personal information and password guessing (and Markov models)

Anatomy of password guessing

Given

$$h = Hash(\text{password} \;\|\; \text{salt})$$

find <password>

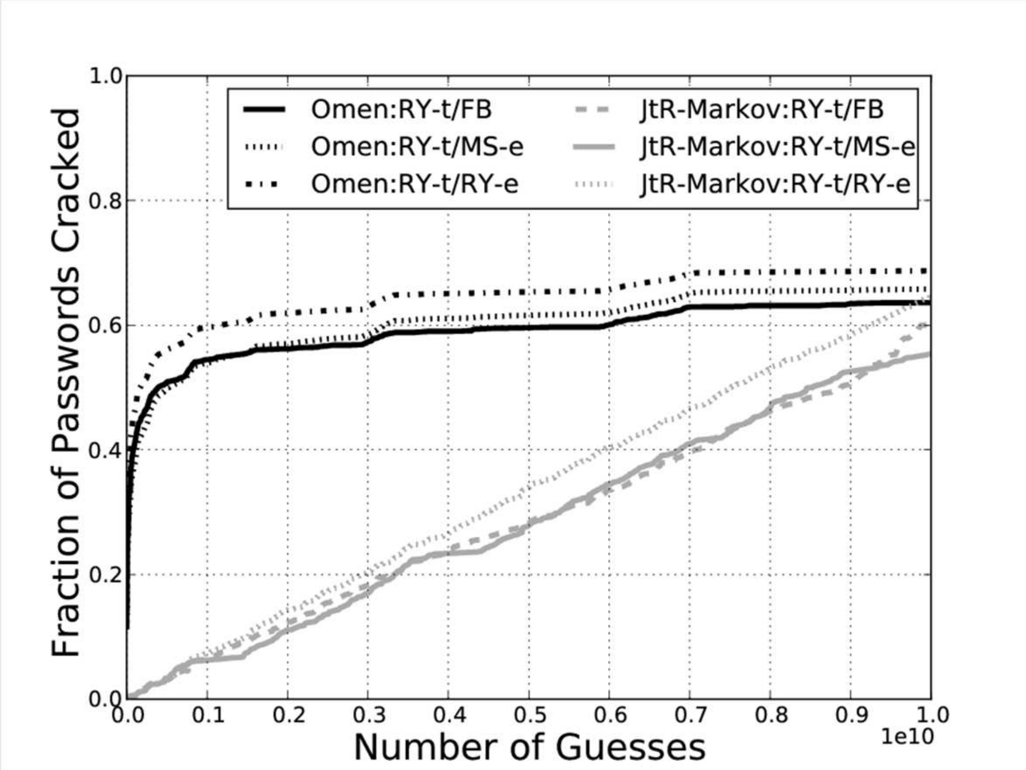
1. Guess password
2. Verify password

Estimate password strength using Markov models

Previous work by Narayanan et al.

- Enumerate all passwords *x* with $P(x) > \lambda$ (in whatever order)
- Implemented in John the Ripper Jumbo Patch

  - 10B guesses
  - Trained on 30M passwords from RockYou
  - Tested on RockYou, MySpace, Facebook passwords



[A. Narayanan, V. Shmatikov. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. ACM CCS 2005.]

Ordered Markov ENumerator (OMEN)

With Claude Castelluccia , Abdelberi Chaabane, Daniele Perito

- Output passwords in the correct order

Pre-processing:

- Discretize probabilities:
  lvl_abc = floor (log (2 * p_abc))
- This makes the "probabilities" additive:
  $L(c_1, \ldots, c_k) =$
  $\sum_{i=n}^{k} L(c_i | c_{i-n+1}, \ldots, c_{i-1})$

Algorithm (for fixed password length **k**)

For *cur_lvl = 0,-1,-2,…*

  For all $a_1,\ldots,a_{k-1}$ with $a_i \leq 0$ and $\Sigma_i\, a_i = cur\_lvl$:

    For all 2-grams with level a_1

      For all "matching" 3-grams with level a_2

        …

          For all "matching" 3-grams with level a_{k-1}

            Output the password

J

Example

For *cur_lvl = 0,-1,-2,…*

For all $a_1,\ldots,a_{k-1}$ with $a_i \le 0$ and $\Sigma_i\, a_i = cur\_lvl$:

For all 2-grams with level a_1

For all "matching" 3-grams with level a_2

…

For all "matching" 3-grams with level a_{k-1}

Output the password ☺

Bsp: n=k=3

- cur_lvl = 0
  - a = (0,0)
    - aa
      - a | aa -> aaa
      - b | aa -> aab
    - ab
      - a | ab -> aba
- cur_lvl = -1
  - a = (-1,0)
    - ac
      - [none]
  - a = (0,-1)
    - aa
      - c | aa -> aac
    - ab
      - z | ab -> abz
- cur_lvl = -2
  - …

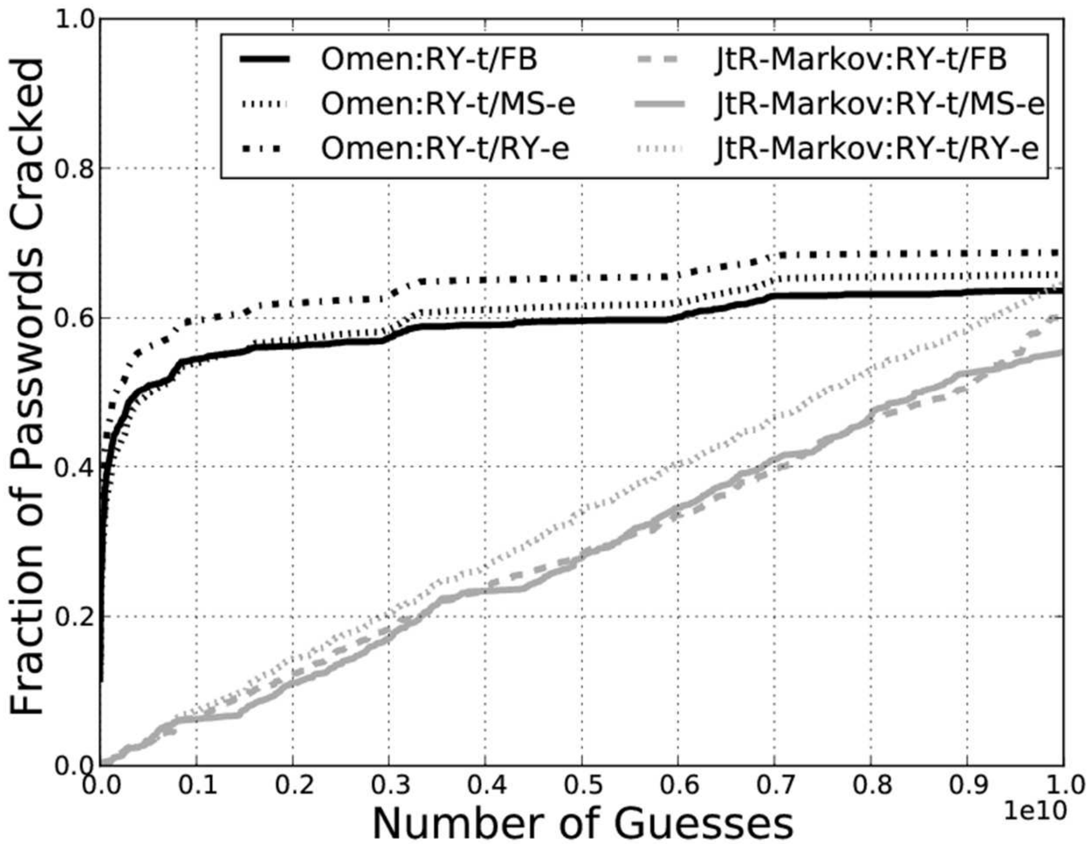
2-grams	lvl
aa	0
ab	0
ac	-1
…	[-9]

3-grams	lvl
a | aa	0
b | aa	0
c | aa	-1
…	[-9]
z | aa	-5
a | ab	0
b | ab	-3
…	[-9]
z | ab	-1
a | ac	-4
…	[-9]

Results (comparing with JtR Markov mode)

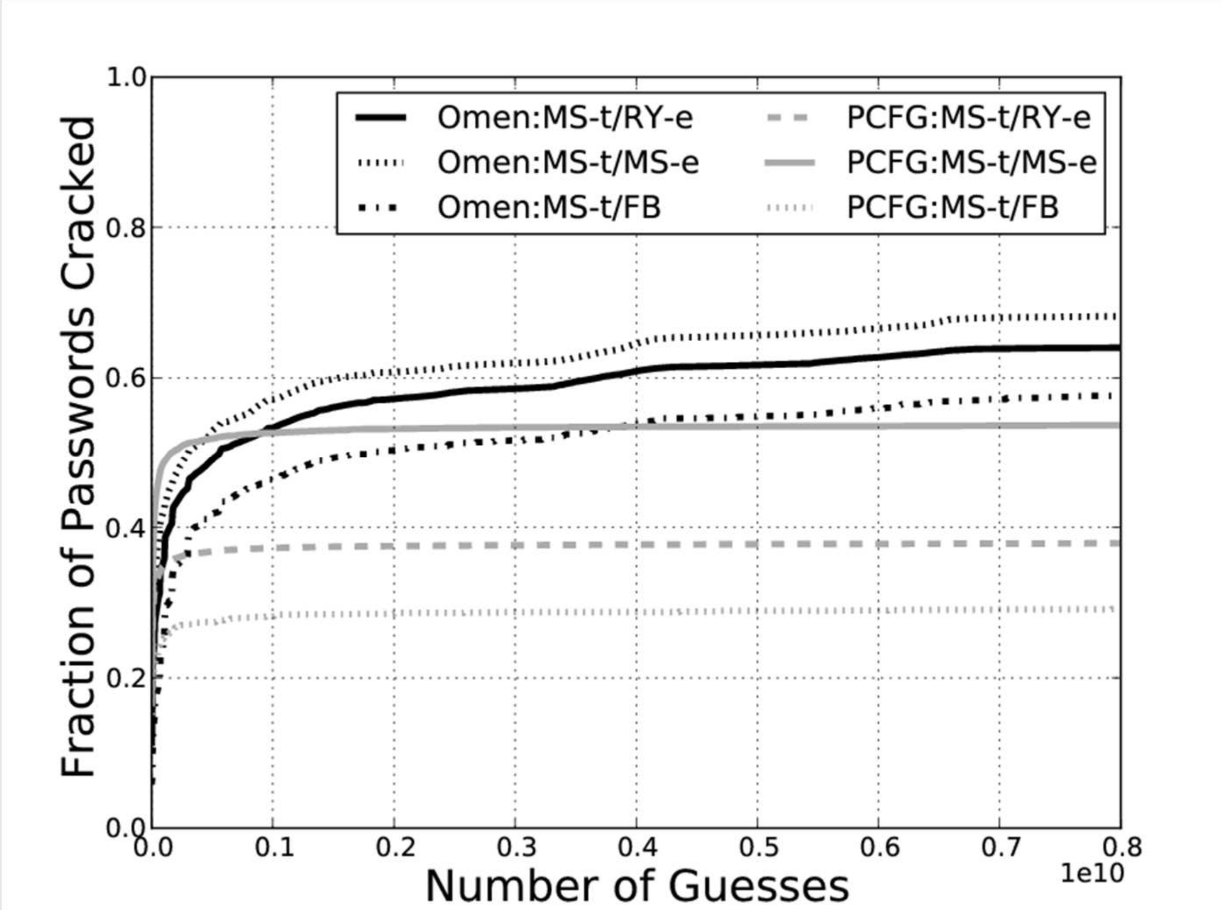
- JtR Markov based on 2-grams
- OMEN based on 3-grams

- At the end-point, performance should be identical (for the same n-gram model)



Results (comparing with PCFG)

- Probabilistic Context Free Grammars (Matt Weir et al.)
- Currently the best-known password guesser
- Learns “structure” of passwords (similar to mangling rues), and uses these to guess passwords



1. Markov models

2. Password Guessing (with Markov models)

3. **Measuring password strength (with Markov models)**

4. Personal information and password guessing (and Markov models)

Attempt: Password rules

"At least 8 characters, one upper-case, lower-case, and special character"

"Must contain at least one number and one special character"

"At least 6 characters and one special character"

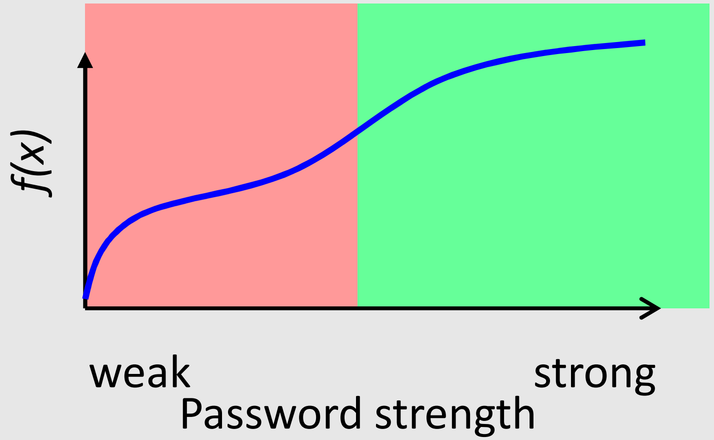
- "At least one number"
  Many passwords follow the form
  <word> + "1"

- MS password checker:
  - Random [a-z]{11} password 'ynazwuaewfv' is scored weak
  - 'P@ssw0rd' is scored strong

RockYou	w/ Policy
123456	abc123
12345	princess1
123456789	blink182
password	angel1
iloveyou	123abc
princess	iloveyou2
1234567	babygirl1
rockyou	iloveyou1
12345678	jesus1
abc123	monkey1

"Optimal" password checkers

With Claude Castelluccia and Daniele Perito [NDSS 2012]



$f(x) = 1/\,P(x)$

§ $P(x)$ varies with sites, over time, …

Solution:
Compute f(x) from the current password database

RockYou	MySpace	PhpBB	Singles.org
123456	password1	123456	123456
12345	abc123	password	jesus
123456789	password	phpbb	password
password	iloveyou1	qwerty	love
iloveyou	iloveyou2	12345	12345678
princess	fuckyou1	letmein	christ
1234567	myspace1	12345678	jesus1
rockyou	soccer1	1234	princess
12345678	iloveyou	test	blessed
abc123	iloveyou!	123	sunshine

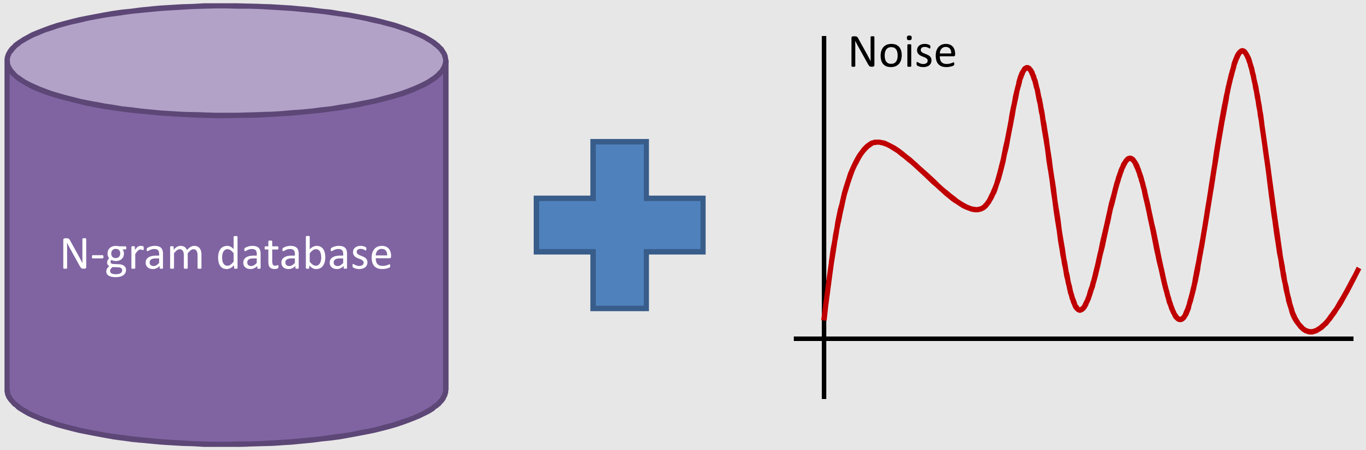
Security considerations

- DB may reveal passwords
- Strong passwords are especially vulnerable
- Take password:
  - **k$Hgw8*lp@**
  - Each n-gram likely to appear only once
  - Chain them together to reconstruct the password

- n-gram database

gw8*l	1
…	
k$Hgw	1
...	
…	
Hgw8*	1
…	
w8*lp	1
…	
$Hgw8	1

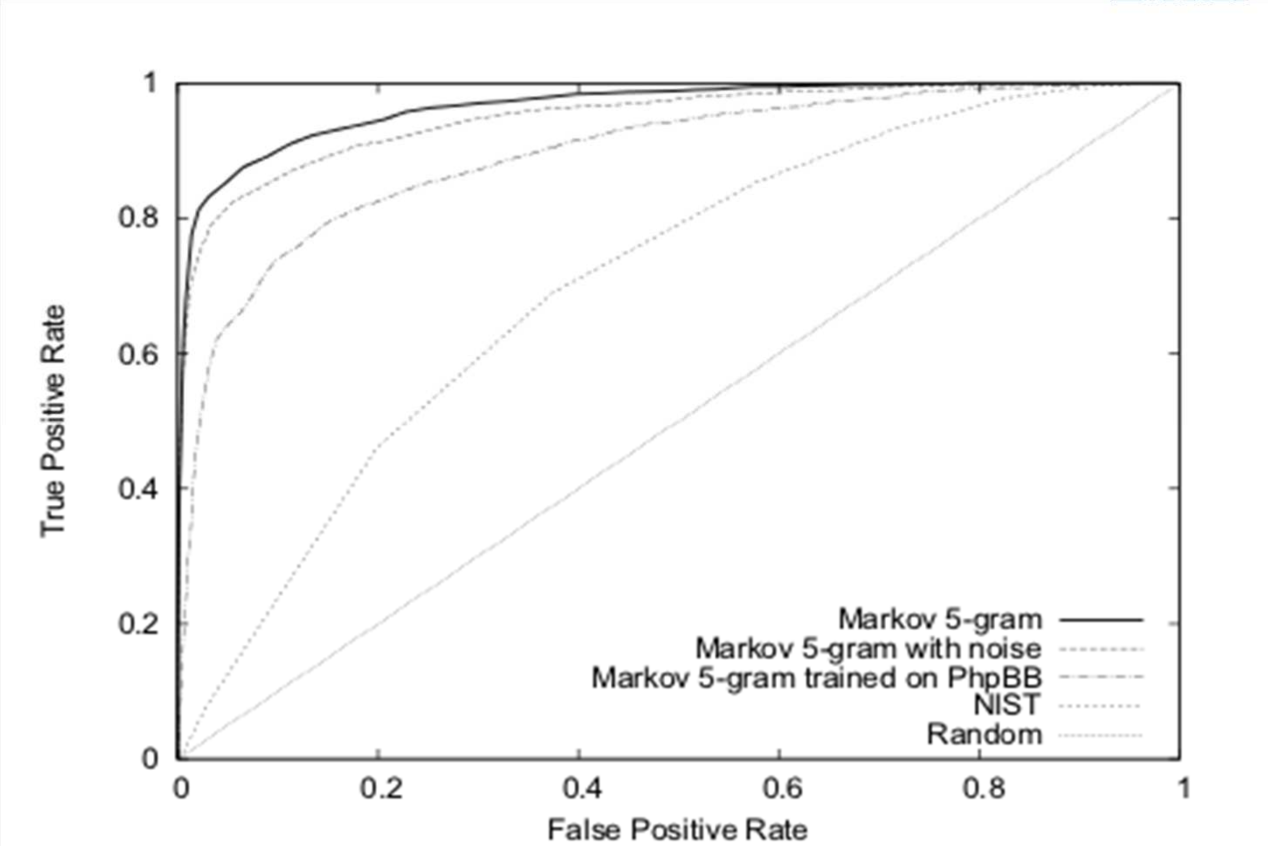
Solution



§ Two questions:

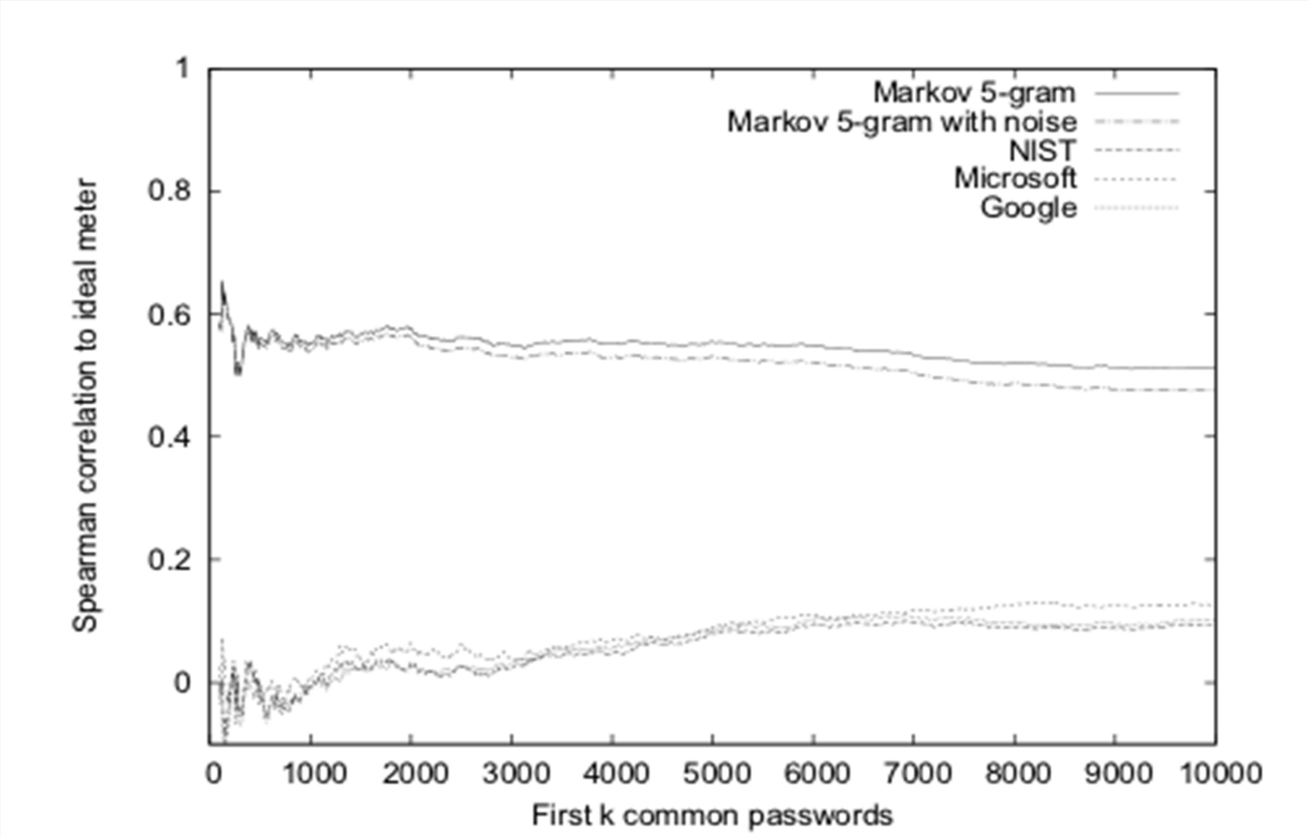
- Security
  - Adding a carefully chosen amount of noise prevents leaking ‘too many’ bits (proof in the paper)
- Accuracy
  - How much does the strength estimation degrade when noise is added?

Experiment #1



- Find strong and weak password in the RockYou dataset
  - Threshold probability $p = 2^{-20}$
  - Build ground truth (weak, strong labels) from empirical frequencies
- See how well we can classify strong and weak password
  - Measure precision and recall

Experiment #2



§ Spearman correlation (i.e., correlation of rank)

- Measure the frequency of the passwords in the RockYou password set
- See how well password checkers follow this ground truth order

1. **Markov models**
2. **Password Guessing** **(with Markov models)**
3. **Measuring password strength** **(with Markov models)**
4. **Personal information and password guessing** **(and Markov models)**

Personal information and password guessing

- JtR uses the username (+ mangling rules) to form password guesses
- Using what mangling rules? What about further information?

- We tested data scraped from public Facebook profiles
  - first/ last name, username, friends, education, work, contacts, location, birthday, siblings

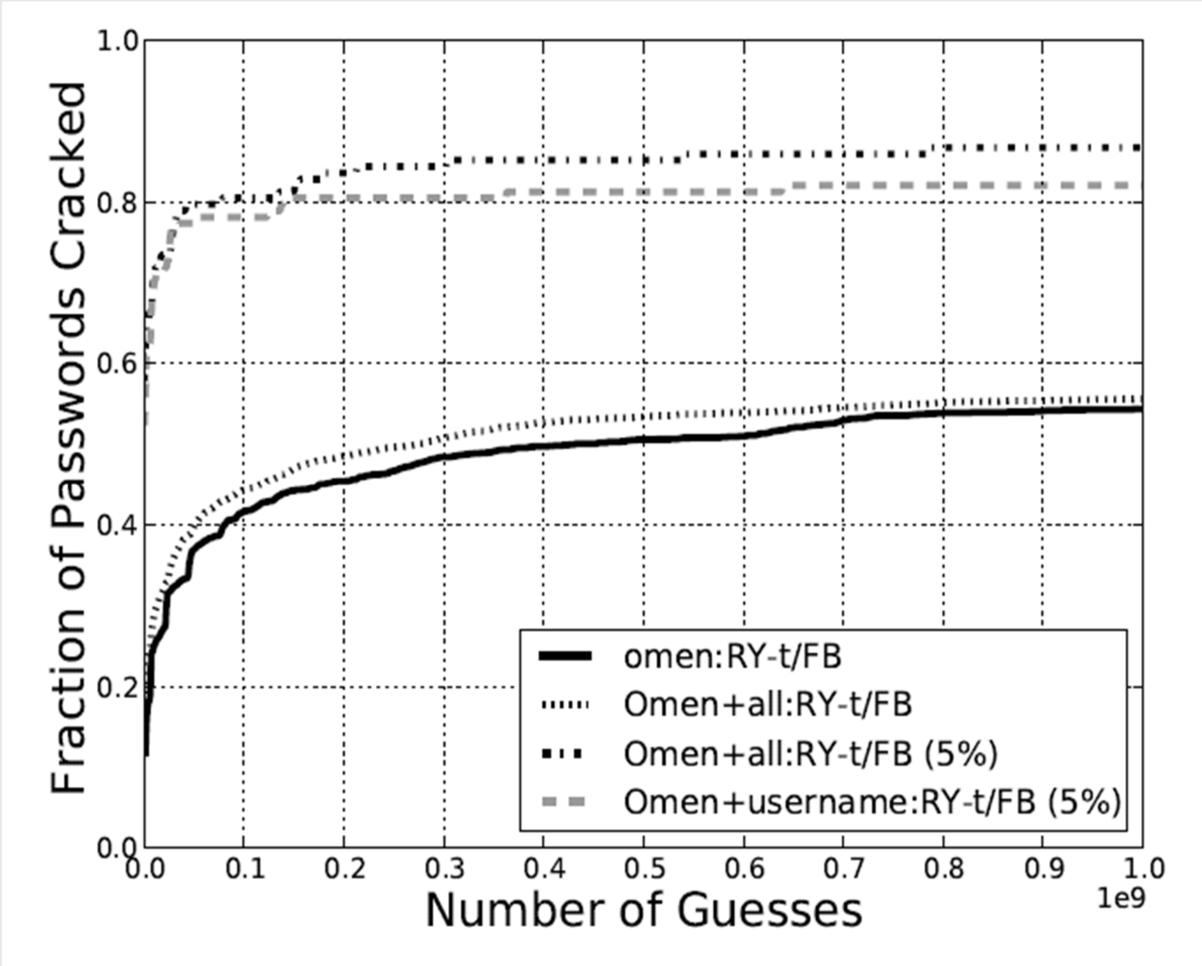
- Can be integrated very nicely with Markov models
  - Boost those n-grams that appear in the personal information

- Useful were siblings, locations, email/username, first name, and birthday
  - Determined automatically which information is useful

- About 5% had a strong overlap between username and passwords

Results

List of Facebook passwords + email address



Conclusion

- Markov models very useful…
- …however, consider “local structure” only

⇒ “combination” of PCFG and Markov models?

- Can be stored securely…
- …if you do it right

- Personal information integrates very nicely with Markov models…
- …but less helpful than we thought…
- …at least the info we tried