Building Better Passwords using Probabilistic Techniques

Shiva Houshmand
Sudhir Aggarwal
Florida State University
Department of Computer Science
E-Crime Investigative Technologies Laboratory

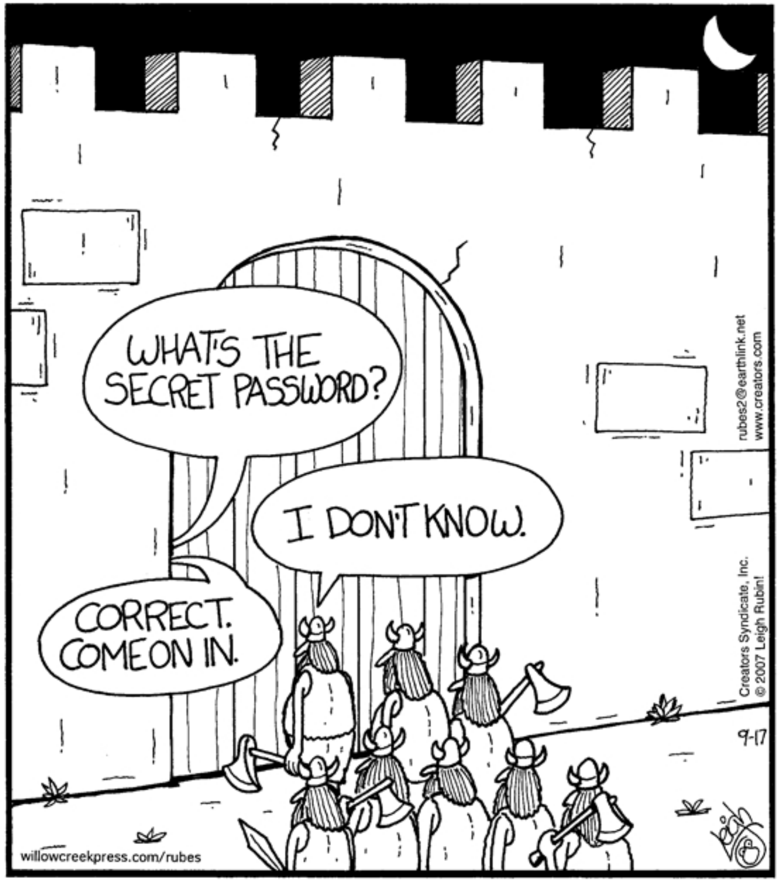
Outline

- Introduction
  - Problems with passwords
- Probabilistic password cracking using grammars
  - Training
  - Cracking
- Our approach – analysis and modification
  - The AMPs system
  - Estimating strength of password
  - Modifying the password
  - Updating AMPs over time
- Entropy measures in updating the system

Introduction

- Passwords are still the most common means of securing computer systems and websites.

- Most users do not have the information to ensure that they are using a “strong” password.



Why great care and consideration should be taken when selecting the proper password

Existing problems with passwords

- Rule-based password creation policies
  - Inconsistent
  - Confusing
  - Frustrating

Website	Length	Digit	Special char
Chase.com	7-32	1	Not allowed
Bank of America	8-20	1	Certain ones allowed
Ets.org	8-16	1	At least one
Banana Republic	5-20	-	Not allowed

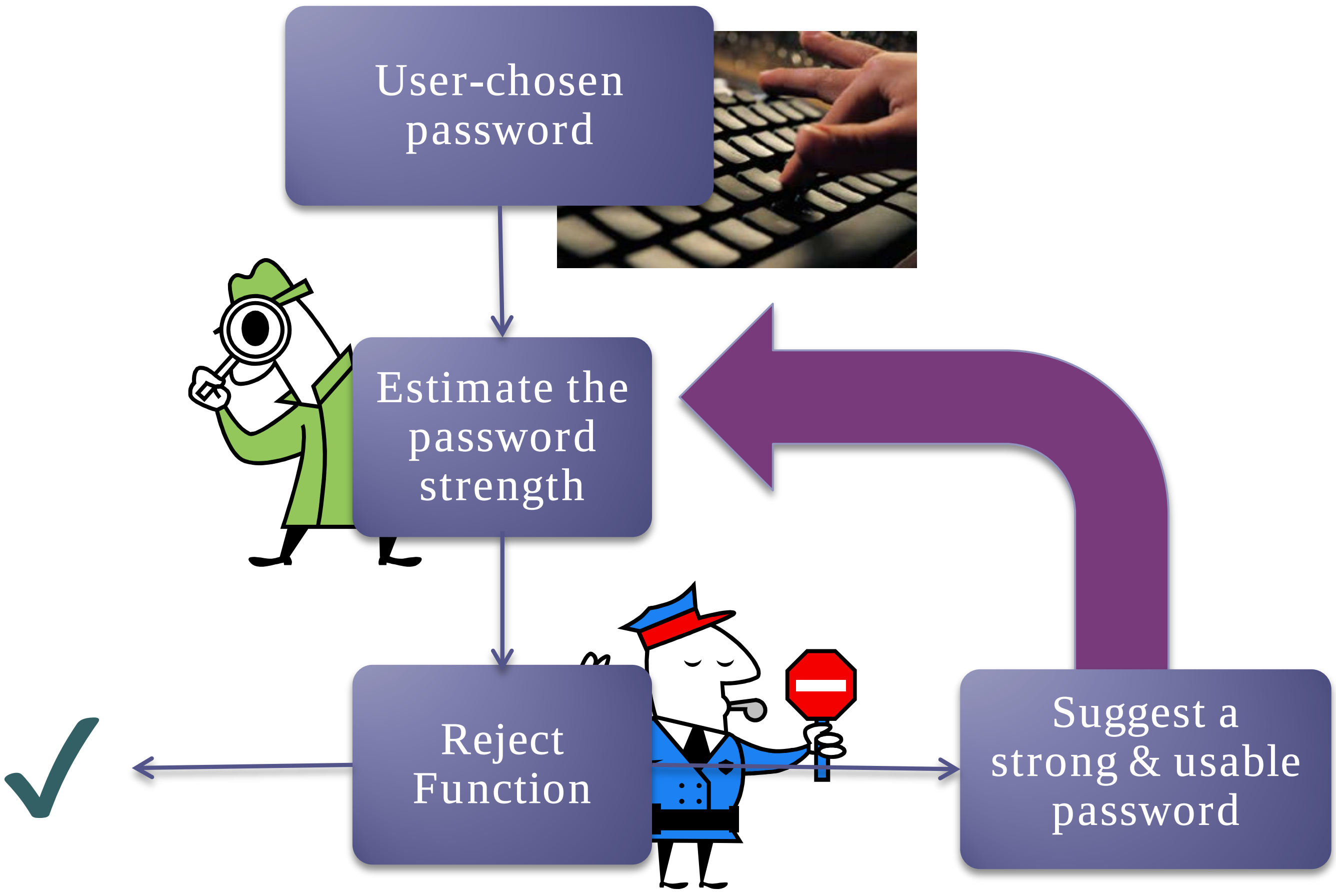
- Password checkers
  - No scientific basis

alice123!

Services	Strength scores	
Apple	Moderate	2/3
Dropbox	Very Weak	1/5
Drupal	Strong	4/4
eBay		-/5
FedEx	Very Weak	1/5
Google	Good	4/5
Intel	Oh No!	1/2
Microsoft (v1)	Strong	3/4
Microsoft (v2)	Weak	1/4
Microsoft (v3)	Medium	2/4
PayPal	Strong	4/4
QQ	Strong	4/4
Skype	Medium	2/3
Twitter	Perfect	6/6
Yahoo!	Very Strong	4/4
12306.cn	Average	2/3

Analyze and Modify Passwords

Abstract



Probabilistic password attack

[Weir, Aggarwal and De Medeiros]

- Infer a probabilistic context-free grammar from datasets
  - Some words are more likely than others
    - Password, monkey, football
  - Some mangling rules are more likely than others
    - Capitalize the first letter, add the digits at the end
  - Assign probability to dictionary words, digits, symbols, mangling rules

Probabilistic password attack

- Training
  - Construct the context-free grammar
    - Parse every password into base structures and count their frequency.
    - Base structures consist of L (alpha sequences), D (digits), S (symbols), M(capitalization)
    - Base structure also includes length information

Password12%

$L_8(M_8)D_2S_1$

M_8 = ULLLLLLL

Probabilistic password attack

Training

Training Set

tiny99
1pass!
this2!
star99
.
.
.
tree99
burn1!
1star!
down11

→ Training →

S→	L_4D_2	0.5
S→	$D_1L_4S_1$	0.25
S→	$L_4D_1S_1$	0.25
D_2→	99	0.7
D_2→	11	0.3
D_1→	1	0.8
D_1→	2	0.2
S_1→	!	1.0
L_4→	alex	0.1
S→* alex2! With probability $0.25 \times 0.1 \times 0.2 \times 1.0 = 0.005$		

Note: Alpha sequence probabilities come from dictionaries and are equal to $1/n_L$, where n_L is the number of words in the dictionary of length L.

Probabilistic password attack

Generating the guesses

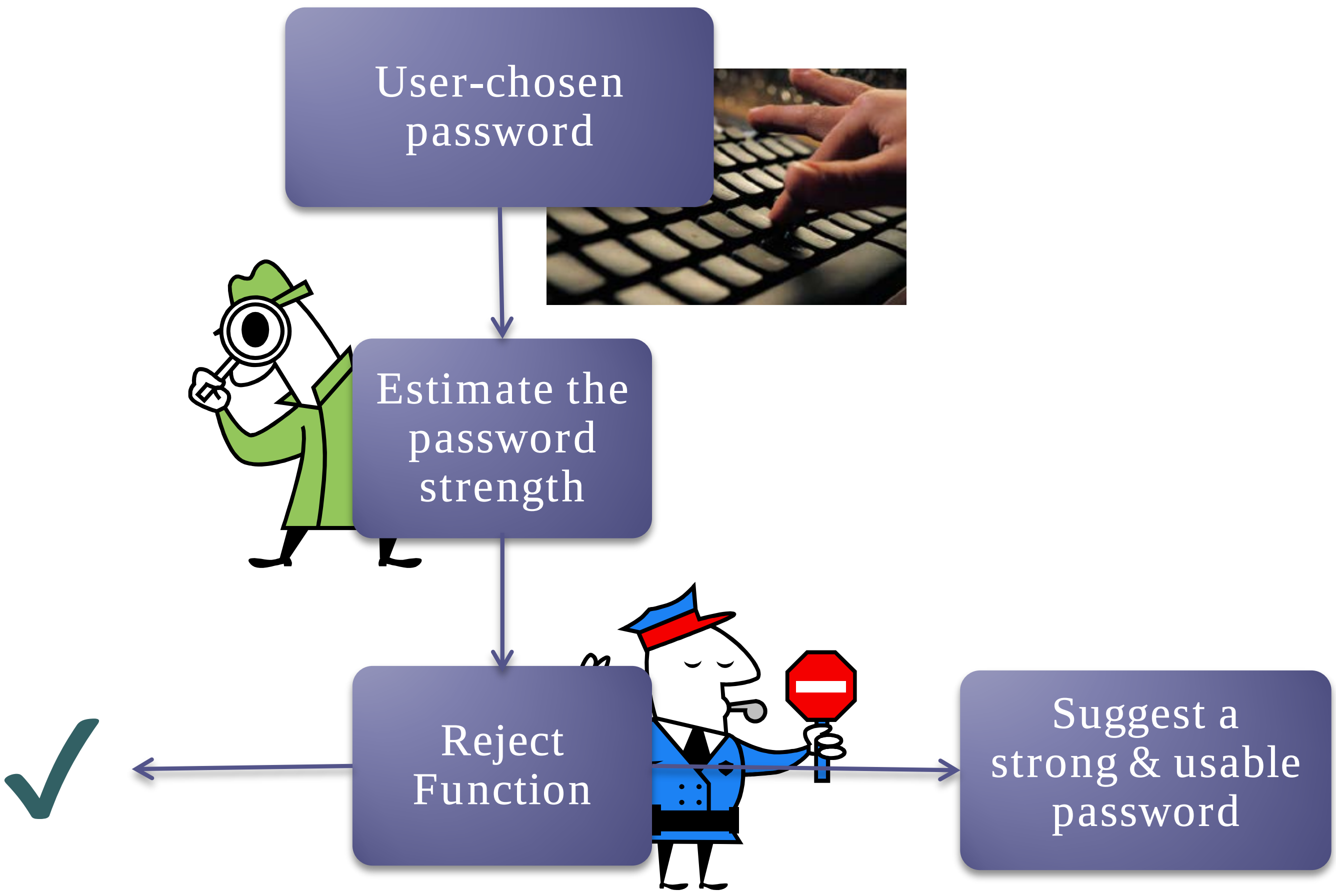
S→	L_4D_2	0.5
S→	$D_1L_4S_1$	0.25
S→	$L_4D_1S_1$	0.25
D_2→	99	0.7
D_2→	11	0.3
D_1→	1	0.8
D_1→	2	0.2
S_1→	!	1.0
L_4→	alex	0.1

Guessing →

alex 99 andy 99 beta 99 …	0.035
1 alex ! 1 andy ! … alex 1 ! andy 1 ! …	0.02
alex 11 andy 11 …	0.015
2 alex ! 2 andy ! … alex 2 ! andy 2 ! ….	0.005

AMP System Overview

Analyzer and Modifier for Passwords

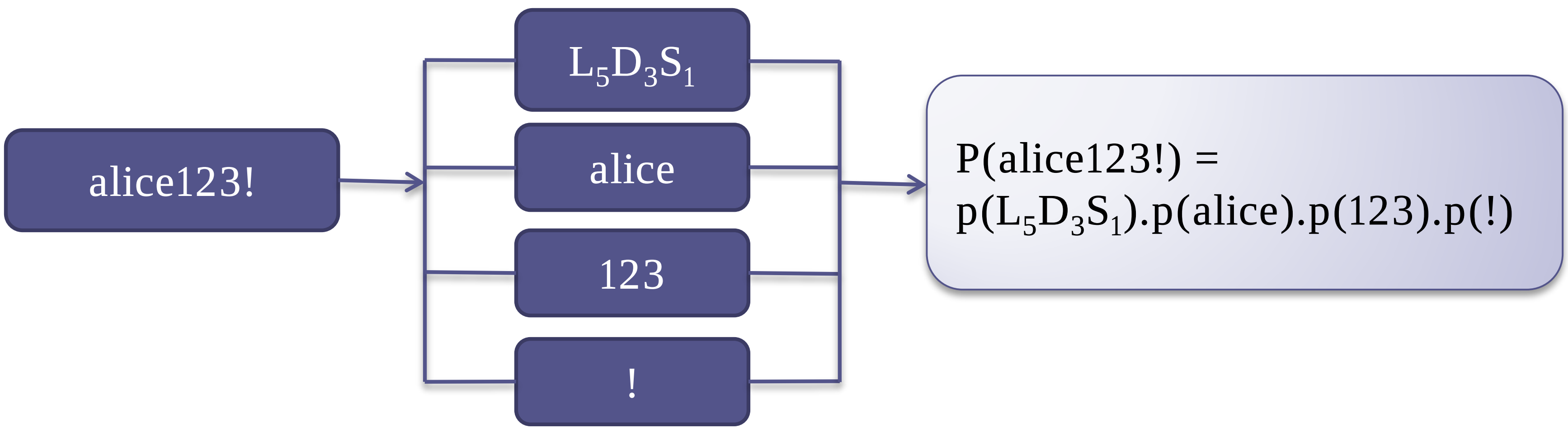


AMP Analyzing

Estimate the password strength

- Train the system on real user passwords and produce the context-free grammar.

- Using the context-free grammar, we calculate the probability of the user-chosen password.



AMP

Setting the Threshold

- Threshold: is a probability value thp

Total_Guesses:	69491415	Probability:	3.1716e-10
Total_Guesses:	69744266	Probability:	3.14529e-10
Total_Guesses:	70000775	Probability:	3.12015e-10
Total_Guesses:	70602451	Probability:	3.09261e-10
Total_Guesses:	71121270	Probability:	3.06813e-10
Total_Guesses:	71519812	Probability:	3.04416e-10
Total_Guesses:	71799637	Probability:	3.02051e-10
Total_Guesses:	72097254	Probability:	2.9943e-10
Total_Guesses:	72304253	Probability:	2.97314e-10
Total_Guesses:	72508371	Probability:	2.95322e-10
Total_Guesses:	72969956	Probability:	2.92856e-10
Total_Guesses:	73582269	Probability:	2.90398e-10
Total_Guesses:	74074952	Probability:	2.87881e-10
Total_Guesses:	74277559	Probability:	2.85883e-10
Total_Guesses:	74826737	Probability:	2.83975e-10
Total_Guesses:	75329839	Probability:	2.81662e-10
Total_Guesses:	75667418	Probability:	2.79658e-10
Total_Guesses:	76191974	Probability:	2.77426e-10
Total_Guesses:	76346168	Probability:	2.75369e-10

- Converting to time: $\frac{Total\_number\_of\_guesses}{Calculations\_per\_hour} = Expected\_time(hour)$

Example table for threshold

Total number of guesses g(t)	Probability t	Time (on my laptop for MD5 hash)
1,800,000,000	1.31×10^{-11}	1 hour
14,400,000,000	1.59×10^{-12}	8 h
21,600,000,000	1.20×10^{-12}	12 h
28,800,000,000	6.37×10^{-13}	16 h
43,200,000,000	2.96×10^{-13}	24 h
86,400,000,000	9.94×10^{-14}	48 h
129,600,000,000	6.7×10^{-14}	72 h
172,800,000,000	5.29×10^{-14}	96 h

AMP

Setting the Threshold approaches

1. Using password guesser
   - Accurate
   - Straightforward
   - Takes a long time

2. Using the context-free grammar
   - Gives a lower bound for the number of guesses
   - Faster

AMP-Setting the Threshold

Running password guesser

```
Total_Guesses:  69491415   Probability:  3.1716e-10   base_struct:  000Ue12
Total_Guesses:  69744266   Probability:  3.14529e-10  base_struct:  00Le$$
Total_Guesses:  70000775   Probability:  3.12015e-10  base_struct:  !Le2Le-
Total_Guesses:  70602451   Probability:  3.09261e-10  base_struct:  2Le12#
Total_Guesses:  71121270   Probability:  3.06813e-10  base_struct:  9.3.
Total_Guesses:  71519812   Probability:  3.04416e-10  base_struct:  Le2Ue143
Total_Guesses:  71799637   Probability:  3.02051e-10  base_struct:  93.2
Total_Guesses:  72097254   Probability:  2.9943e-10   base_struct:  Le63Le07
-------------------------------------------------------------------------------
Total_Guesses:  72304253   Probability:  2.97314e-10  base_struct:  0000..
Total_Guesses:  72508371   Probability:  2.95322e-10  base_struct:  Ue5Ue4
Total_Guesses:  72969956   Probability:  2.92856e-10  base_struct:  1Le95Le3
Total_Guesses:  73582269   Probability:  2.90398e-10  base_struct:  93.3
Total_Guesses:  74074952   Probability:  2.87881e-10  base_struct:  12 13
Total_Guesses:  74277559   Probability:  2.85883e-10  base_struct:  27Le2001
Total_Guesses:  74826737   Probability:  2.83975e-10  base_struct:  Le3Ue1Ue7
Total_Guesses:  75329839   Probability:  2.81662e-10  base_struct:  Le58Le8Le
Total_Guesses:  75667418   Probability:  2.79658e-10  base_struct:  .Le2Le0
Total_Guesses:  76191974   Probability:  2.77426e-10  base_struct:  5_007
Total_Guesses:  76346168   Probability:  2.75369e-10  base_struct:  Le@Le!2
Total_Guesses:  76964953   Probability:  2.73163e-10  base_struct:  4Le9Le5
Total_Guesses:  77380282   Probability:  2.71075e-10  base_struct:  1@2-1
Total_Guesses:  77947787   Probability:  2.69186e-10  base_struct:  9Le�
Total_Guesses:  78858297   Probability:  2.67563e-10  base_struct:  1991+
Total_Guesses:  78913486   Probability:  2.65541e-10  base_struct:  1138Le10
```

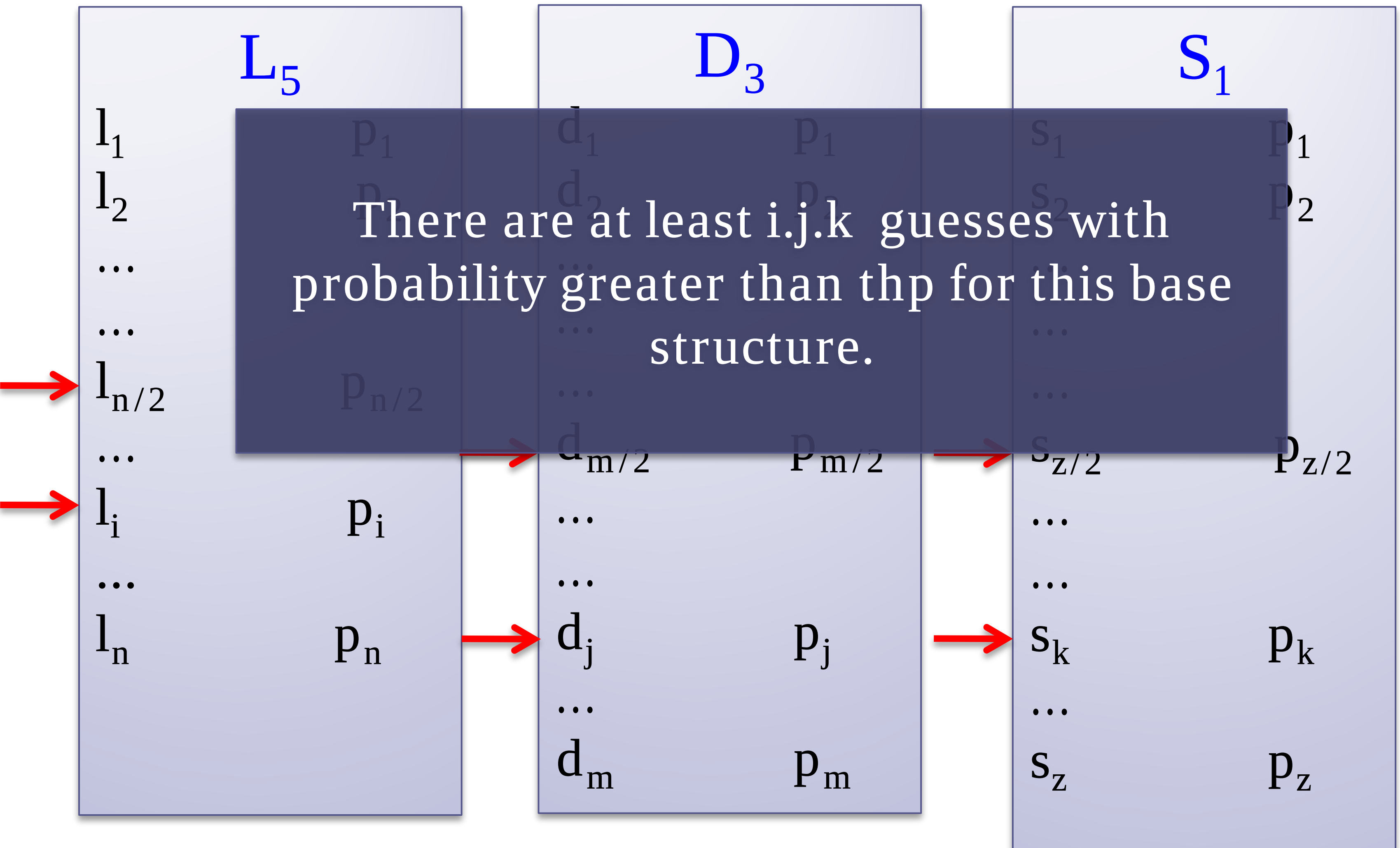
AMP-Setting the Threshold

Using the Grammar

- Estimating the number of guesses before threshold (thp).
- Starting from the first base structure, for example $b_1=L_5D_3S_1$ with probability p_1, we need to find the elements in each component so that the product of their probabilities is > thp.

AMP

Set the threshold – Using the context free grammar



There are at least i.j.k guesses with probability greater than thp for this base structure.

MODIFYING A WEAK PASSWORD

Modifying a weak password

- There are certain numbers or words that are easy to remember for each individual.
- Edit distance: The minimum number of operations used to transform a string to another one.
- We only change within edit distance of 1.

Modifying a weak password

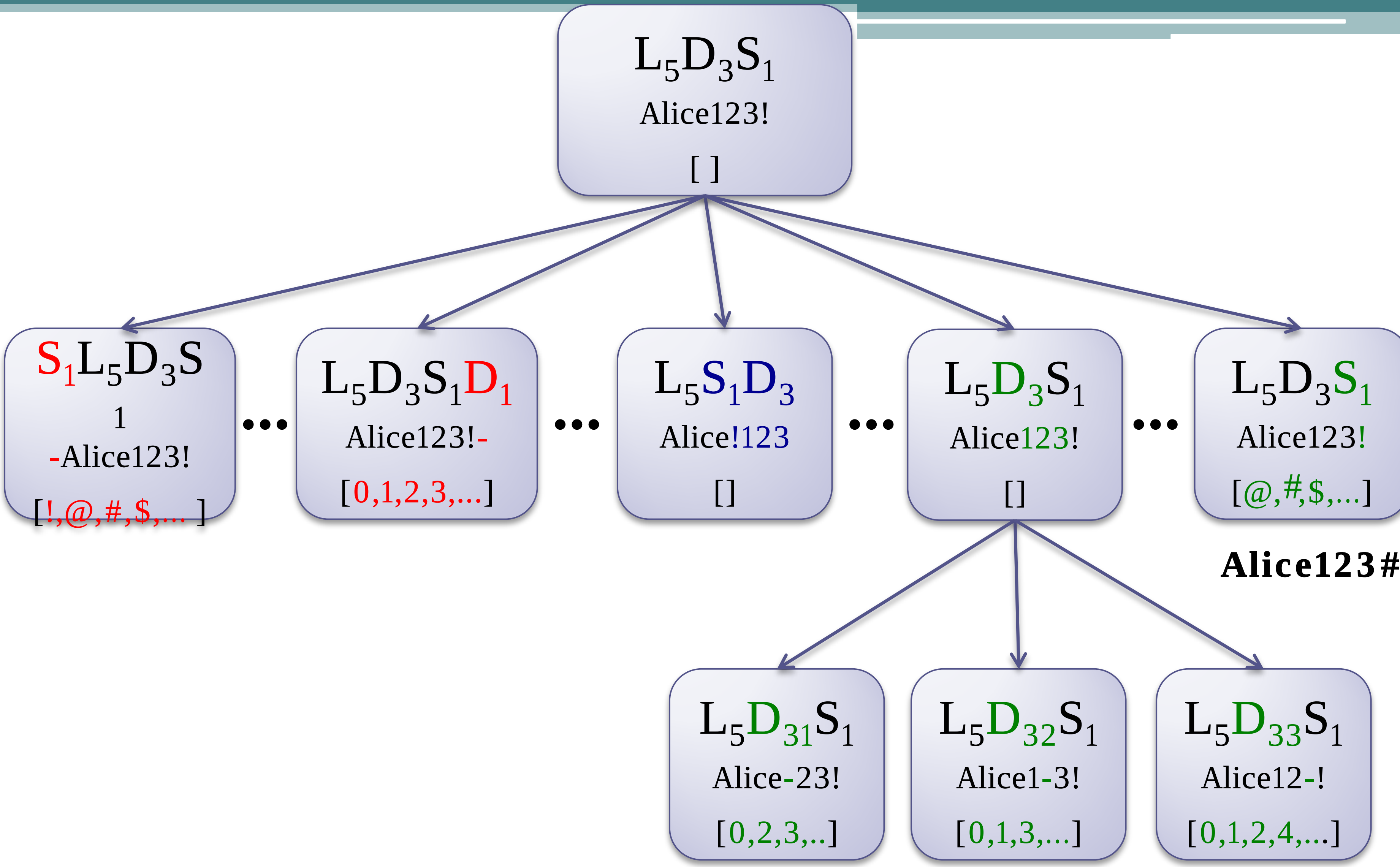
distance function

- Operations on the base structure — $L_5D_3S_1$
  - Insertion — L5S_1D3S1
  - Deletion — L_5D_3~~S_1~~
  - Transposition — $D_3L_5S_1$

- Operations on the component — D_3: 123
  - Insertion — 1263
  - Deletion — ~~1~~23
  - Substitution — 129

  L_5 : alice

  - Case (only for alpha part) — aLice


L5D3S1
Alice123!
[ ]
S1L5D3S1
-Alice123!
[!,@,#,$,...]
L5D3S1D1
Alice123!-
[0,1,2,3,...]
L5S1D3
Alice!123
[]
L5D3S1
Alice123!
[]
L5D3S1
Alice123!
[@,#,$,...]
Alice123#
L5D31S1
Alice-23!
[0,2,3,..]
L5D32S1
Alice1-3!
[0,1,3,...]
L5D33S1
Alice12-!
[0,1,2,4,...]

Modifying a weak password

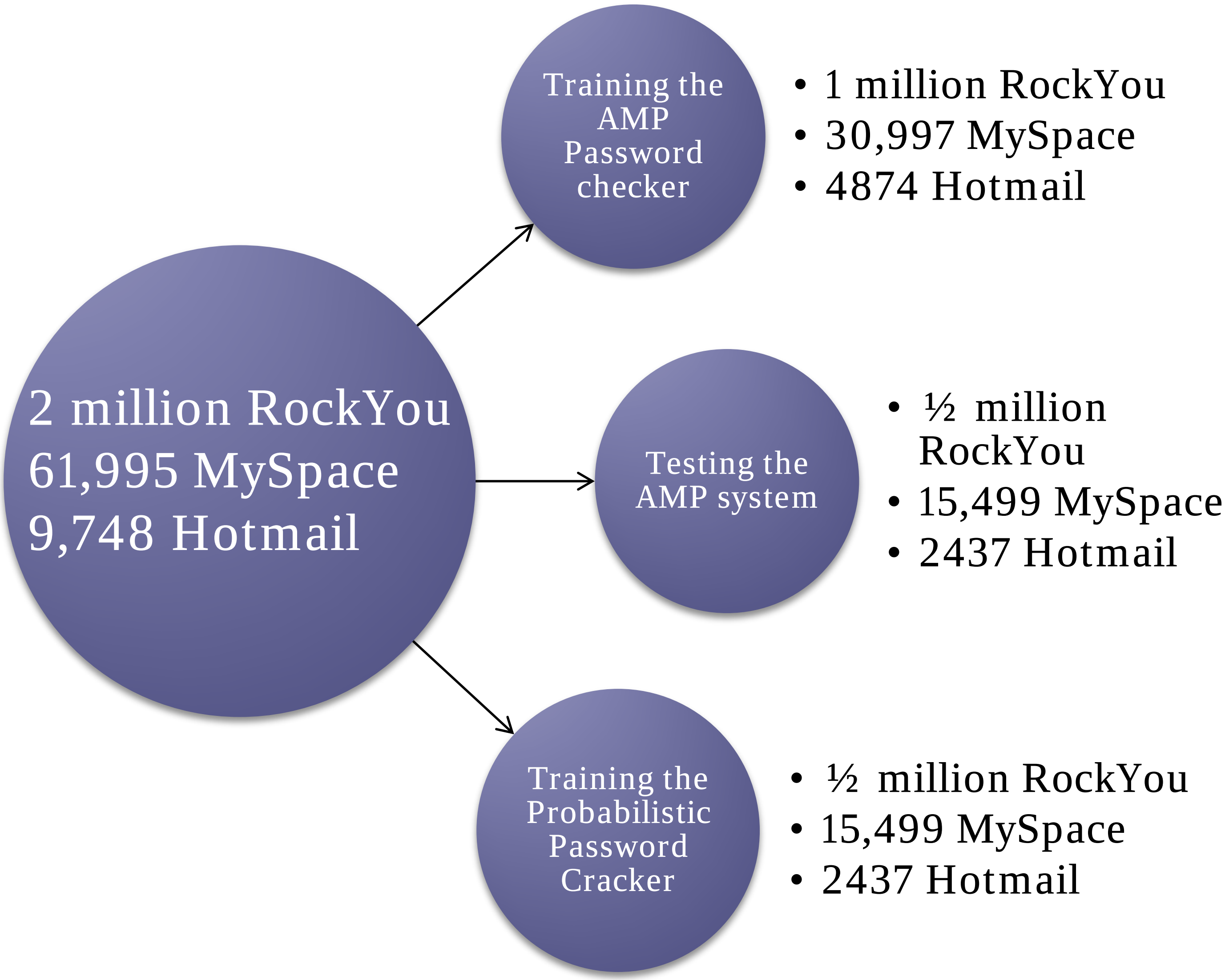
Example

Input password to AMP	Output of modifier
trans2	%trans2
colton00	8colton00
789pine	789pinE
mitch8202	mitch=8202
cal1fero	cal8fero
KILLER456	KILlER456
violin22	violin^22
ATENAS0511	0511AETENAS
*zalena6	*3zalena6
KYTTY023	KYTTY023r

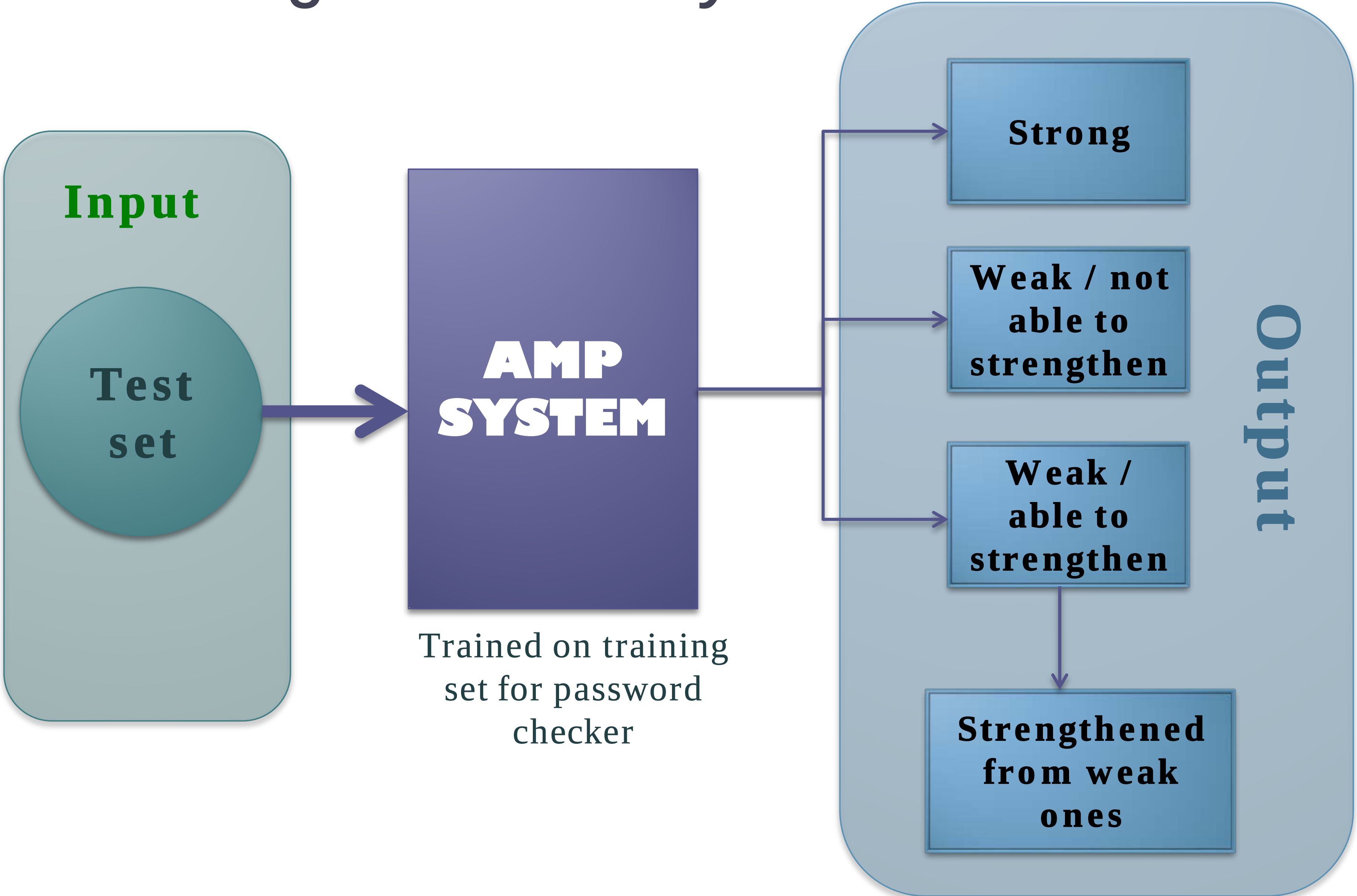
Testing

Testing the AMP System

Experiment Setup



Testing the AMP System



Some results

Cracked by John the Ripper – 1 day threshold

	Originally Strong passwords	Originally Weak passwords (Not able to make stronger)	Originally Weak passwords (Able to make stronger)	Strengthened passwords Modified from weak ones
Hotmail $\frac{cracked}{total}$ Percentage	$\frac{2}{325}$ (0.61%)	$\frac{49}{53}$ (92.45%)	$\frac{988}{2059}$ (47.98%)	$\frac{2}{2059}$ (0.097%)
MySpace $\frac{cracked}{total}$ Percentage	$\frac{23}{1484}$ (1.55%)	$\frac{104}{149}$ (69.80%)	$\frac{5,343}{13,866}$ (38.53%)	$\frac{71}{13,866}$ (0.51%)
RockYou $\frac{cracked}{total}$ Percentage	$\frac{281}{32,794}$ (0.86%)	$\frac{22,248}{24,745}$ (89.90%)	$\frac{235,302}{442,461}$ (53.18%)	$\frac{1,186}{442,461}$ (0.27%)

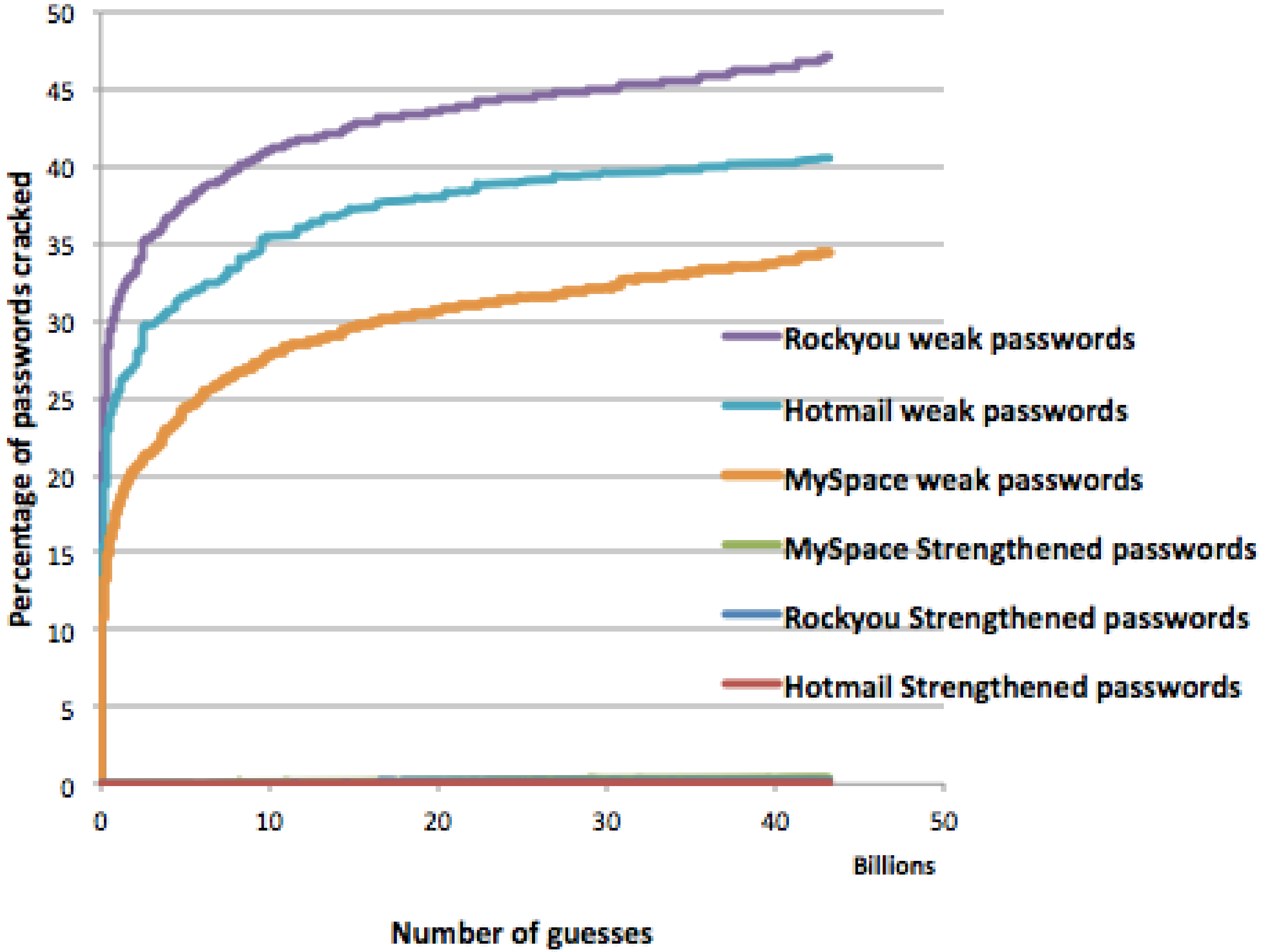
Some results

Cracked by Probabilistic Password Cracker – 1 day threshold

	Originally Strong passwords	Originally Weak passwords (Not able to make stronger)	Originally Weak passwords (Able to make stronger)	Strengthened passwords Modified from weak ones
Hotmail $\frac{\text{cracked}}{\text{total}}$ Percentage	$\frac{1}{325}$ (0.3%)	$\frac{53}{53}$ (100%)	$\frac{1069}{2059}$ (51.91%)	$\frac{113}{2059}$ (5.48%)
MySpace $\frac{cracked}{total}$ Percentage	$\frac{27}{1484}$ (1.81%)	$\frac{135}{149}$ (90.60%)	$\frac{8,341}{13,866}$ (60.15%)	$\frac{698}{13,866}$ (5.03%)
RockYou $\frac{cracked}{total}$ Percentage	$\frac{467}{32,794}$ (1.42%)	$\frac{24,378}{24,745}$ (98.51%)	$\frac{259,027}{442,461}$ (58.54%)	$\frac{18,134}{442,461}$ (4.1%)

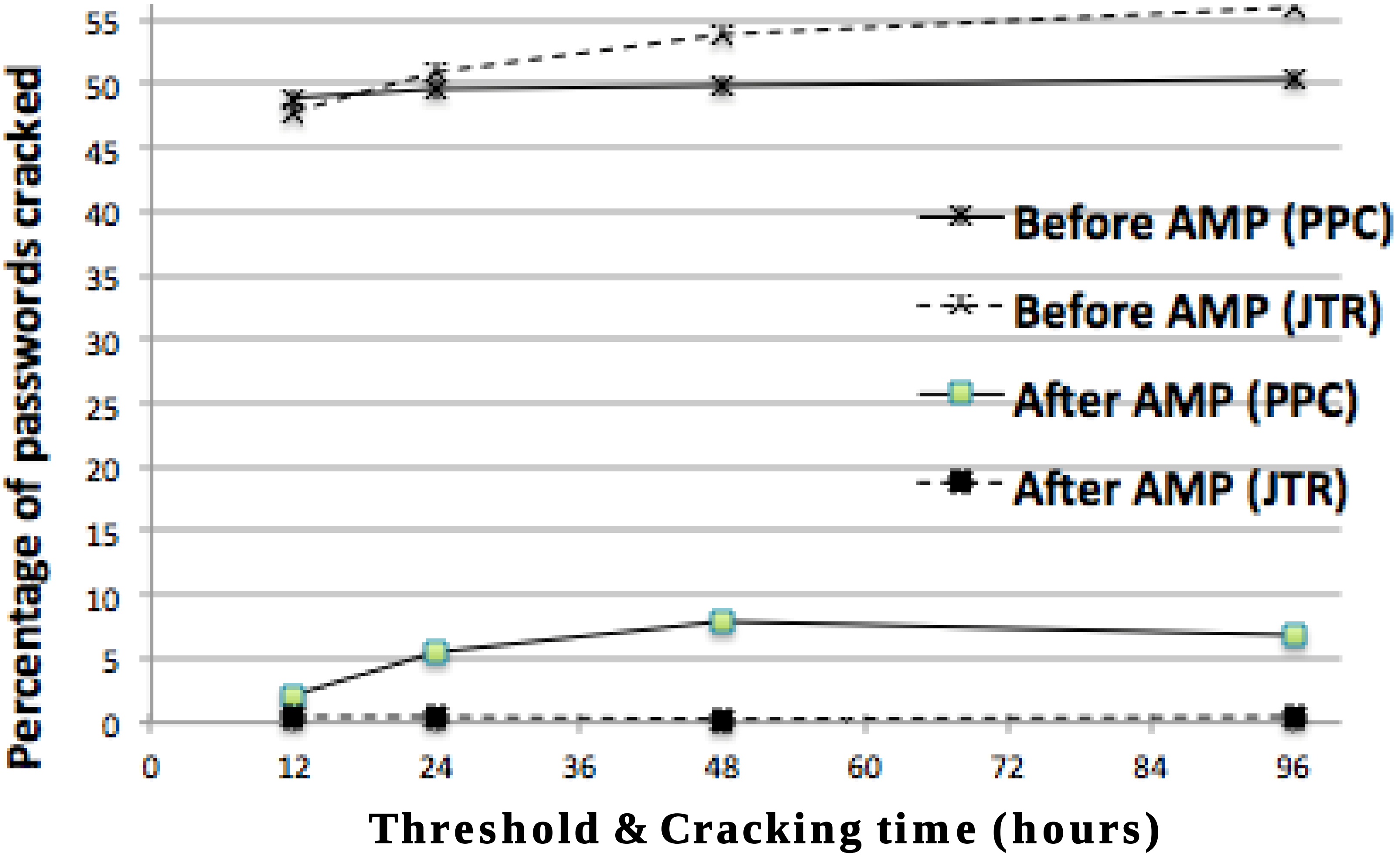
Some results

Weak and Strengthened passwords cracked by John the Ripper



Some results

Beyond 1 day Threshold



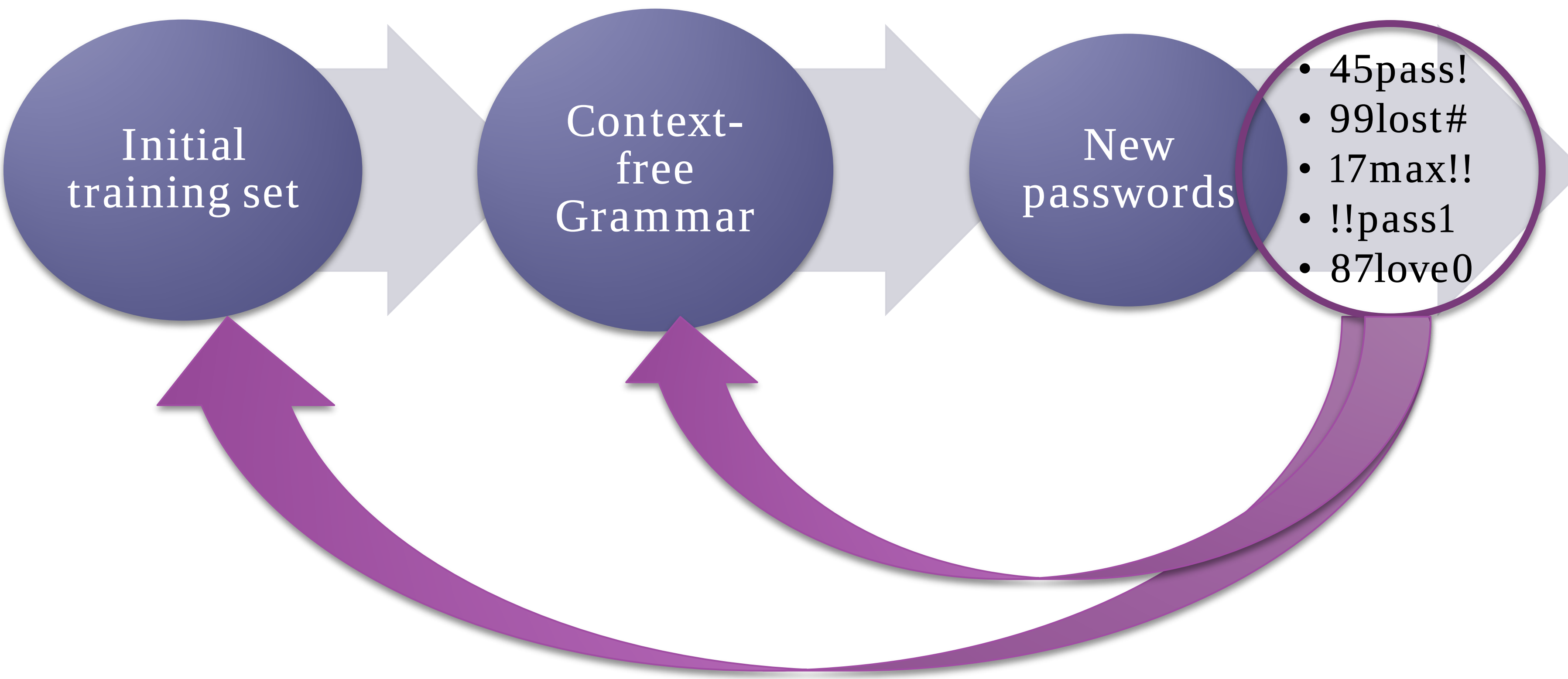
Dynamically Updating

Update the training set

- As we keep using AMP, we suggest more passwords with lower probabilities as strong passwords.
- As people use our suggested passwords more, the probability distribution of passwords changes.
- An attacker might be able to crack passwords using the recent set of real user passwords.

AMP

Update the training set



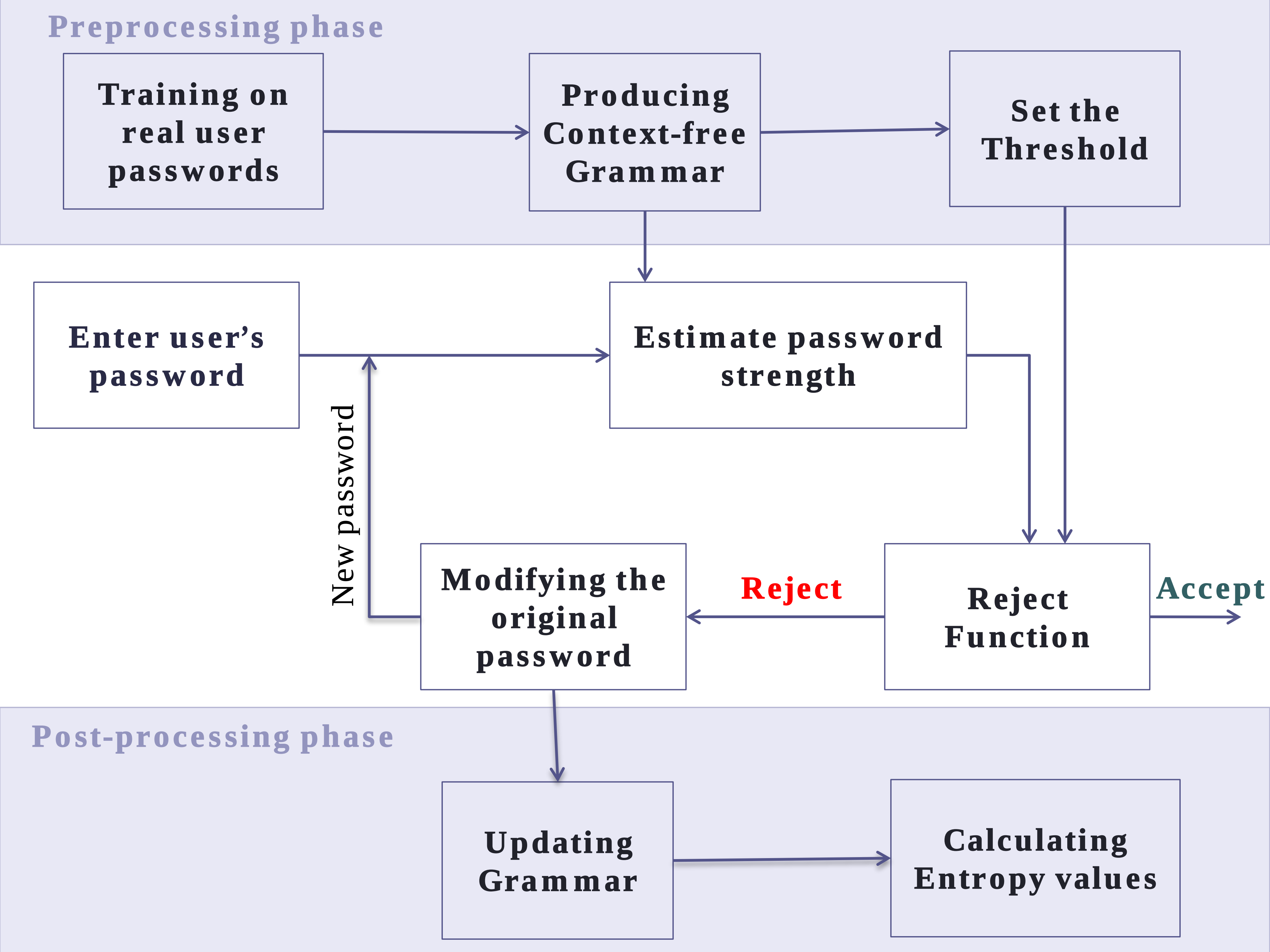
AMP

Update the Context-free Grammar

Base structures		
b_1	$\frac{n_1}{N}$	$\frac{n_1}{N+1}$
b_2	$\frac{n_2}{N}$	$\frac{n_2}{N+1}$
b_3	$\frac{n_3}{N}$	$\frac{n_3}{N+1}$
.		
.		
$b_i=S_2D_2L_4$	$\frac{n_i}{N}$	$\frac{n_i+1}{N+1}$
.		
.		
b_m	$\frac{n_m}{N}$	$\frac{n_m}{N+1}$

S_2		
s_1	$\frac{n_1}{N}$	$\frac{n_1}{N+1}$
s_2	$\frac{n_2}{N}$	$\frac{n_2}{N+1}$
s_3	$\frac{n_3}{N}$	$\frac{n_3}{N+1}$
.		
.		
s_j=!!	$\frac{n_j}{N}$	$\frac{n_j+1}{N+1}$
.		
.		
s_m	$\frac{n_m}{N}$	$\frac{n_m}{N+1}$

Base structures			S_2			D_2		
b_1	$\frac{n_{b1}}{N_b}$	$\frac{n_{b1}}{N_b+1}$	s_1	$\frac{n_{s1}}{N_s}$	$\frac{n_{s1}}{N_s+1}$	d_1	$\frac{n_{d1}}{N_d}$	$\frac{n_{d1}}{N_d+1}$
b_2	$\frac{n_{b2}}{N_b}$	$\frac{n_{b2}}{N_b+1}$	s_2	$\frac{n_{s2}}{N_s}$	$\frac{n_{s2}}{N_s+1}$	d_2	$\frac{n_{d2}}{N_d}$	$\frac{n_{d2}}{N_d+1}$
b_3	$\frac{n_{b3}}{N_b}$	$\frac{n_{b3}}{N_b+1}$	s_3	$\frac{n_{s3}}{N_s}$	$\frac{n_{s3}}{N_s+1}$	d_3	$\frac{n_{d3}}{N_d}$	$\frac{n_{d3}}{N_d+1}$
.			.			.		
.			.			.		
$b_i=S_2D_2L_4$	$\frac{n_{bi}}{N_b}$	$\frac{n_{bi}+1}{N_b+1}$	s_j=!!	$\frac{n_{sj}}{N_s}$	$\frac{n_{sj}+1}{N_s+1}$	d_l=78	$\frac{n_{dl}}{N_d}$	$\frac{n_{dl}+1}{N_d+1}$
.			.			.		
.			.			.		
b_m	$\frac{n_{bm}}{N_b}$	$\frac{n_{bm}}{N_b+1}$	s_k	$\frac{n_{sk}}{N_s}$	$\frac{n_{sk}}{N_s+1}$	d_t	$\frac{n_{dt}}{N_d}$	$\frac{n_{dt}}{N_d+1}$



Preprocessing phase
Training on real user passwords
Producing Context-free Grammar
Set the Threshold
Enter user's password
Estimate password strength
New password
Modifying the original password
Reject
Reject Function
Accept
Post-processing phase
Updating Grammar
Calculating Entropy values

Metrics for password strength

Metrics for password strength

- Guessing Entropy G(X):

  average number of tries for finding

  the password

$$p_1 \geq p_2 \geq \ldots \geq p_n$$

$$G(X) = \sum_{i=1}^{n} i.p_i$$

- Shannon Entropy:

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

Where P(X=x) is the probability that the variable X has the value x.

- Massey proved the following relationship for discrete distributions:

$$G(X) \geq (\frac{1}{4}) 2^{H(X)} + 1$$

Metric for password strength

- Massey proved the following relationship for discrete distributions:

$$G(X) \geq \left(\frac{1}{4}\right)2^{H(X)} + 1$$

Calculation of Entropy based on Context-free grammars for a password distribution

$S \rightarrow L_2D_3$

D_2L_2

S_1D_2

$p(S \rightarrow L_2D_3) = p(B = L_2D_3)$

$L_2D_3 \rightarrow$ up999

it999

$p(L_2D_3 \rightarrow up999)$

$= p(up) \times p(999)$

10it

$S_1D_2 \rightarrow$ #11

$11

#10

$10

Calculation of Entropy

based on context-free grammar for a password distribution

$$H(B,R) = H(B) + H(R \mid B)$$

$$= H(B) + \sum_{b_i} p(b_i) H(R \mid B = b_i)$$

S → **L_2D_3**
D_2L_2
S_1D_2

$$H(B,R) = H(B) + H(R \mid B)$$

$$= H(B) + \sum_{b_i} p(b_i) H(R \mid B = b_i)$$

$$= -\sum_{b_i} p(b_i) \log p(b_i) + \sum_{b_i} p(b_i) H(R \mid B = b_i)$$

$$= -\sum_{b_i} p(b_i) \log p(b_i) + \left[p(b_1) H(L_2D_3) + p(b_2) H(D_2L_2) + p(b_3) H(S_1D_2) \right]$$

Calculation of Entropy

based on context-free grammar for a password distribution

$$H(B,R) = H(B) + \left[p(b_1)H(L_2D_3) + p(b_2)H(D_2L_2) + p(b_3)H(S_1D_2) \right]$$

S → $\mathbf{L_2D_3}$
$\mathbf{D_2L_2}$
$\mathbf{S_1D_2}$

$$H(L_2D_3) = -\sum_{l_2}\sum_{m_2}\sum_{d_3} p(l_2,m_2,d_3)\log p(l_2,m_2,d_3)$$

$$= -\sum_{l_2}\sum_{m_2}\sum_{d_3} p(l_2)p(m_2)p(d_3)\log\left(p(l_2)p(m_2)p(d_3)\right)$$

$$= -\sum_{l_2}\sum_{m_2}\sum_{d_3} p(l_2)p(m_2)p(d_3)\left[\log p(l_2) + \log p(m_2) + \log p(d_3)\right]$$

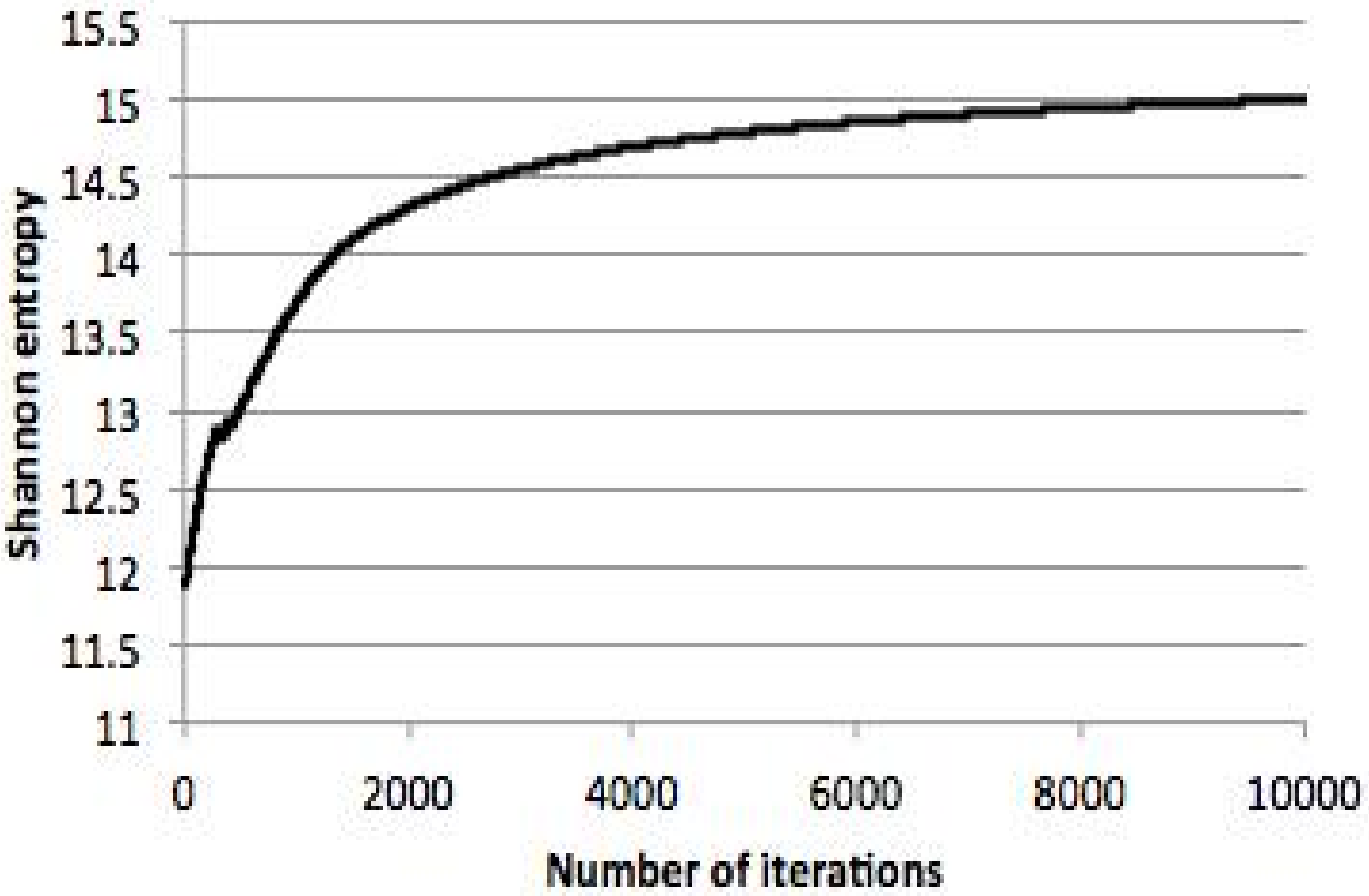
$$= -\sum_{l_2} p(l_2)\log p(l_2) + -\sum_{m_2} p(m_2)\log p(m_2) + -\sum_{d_3} p(d_3)\log p(d_3)$$

$$= H(L_2) + H(M_2) + H(D_3)$$

Increasing Shannon Entropy

- User enters their chosen password
- If it is not strong enough, it will be rejected
- We suggest a new password with probability less than 1/n, n being the total number of passwords in the distribution.
- We update the probabilities by adding the new password to the training set.

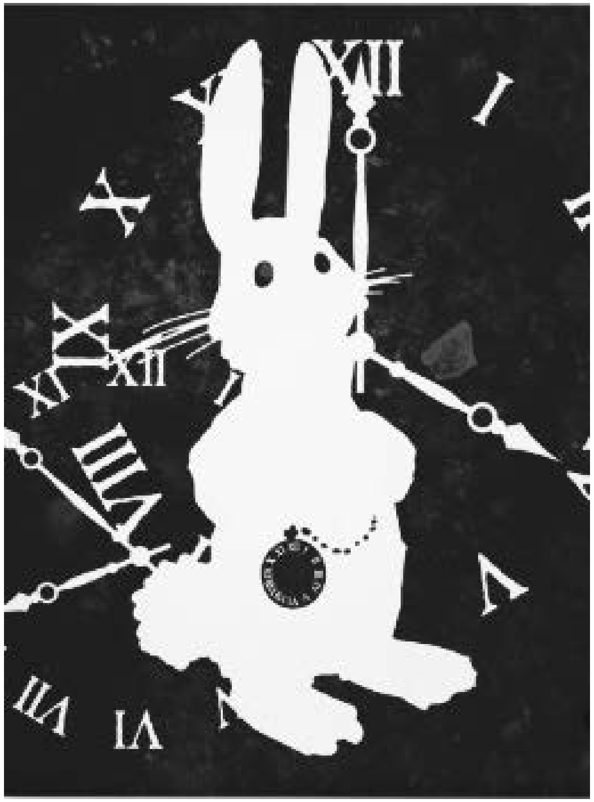
Increasing Shannon entropy



Conclusion

- We developed a technique to measure password strength based on the distribution.
- We developed a model and built a system to help users have strong passwords which are resistant to real attacks.
- We developed dynamic modification techniques to maintain the security of our system and also showed that our updating algorithm drives the grammar to higher Shannon entropy.
- We developed a way to calculate realistic entropy values for password distributions.

Questions/Comments?



- E-Mail Address
  - sh09r@my.fsu.edu
  - sudhir@cs.fsu.edu

- Shiva Houshmand, Sudhir Aggarwal, "Building Better Passwords using probabilistic techniques," ACSAC'12.
- M. Weir, Sudhir Aggarwal, Breno de Medeiros, Bill Glodek, "Password Cracking Using Probabilistic Context Free Grammars," Proceedings of the 30th IEEE Symposium on Security and Privacy, May 2009, pp. 391-405.
- M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10), October 4-8, 2010, pp. 163-175.