IBRUT: Automating Intelligent Guessing Techniques. Pattern Analysis, Extraction, Reuse.

"全自動智慧型密碼猜測技術，字串分析、萃取、再用“

Fyodor fygrave@tigerteam.net

***Security** today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots*

- Altered quite by John Dryden

Agenda

- “Bigger and better idiots” problem
- Traditional methods
- Analysis
- GA theory quickly
- Problem model design
- Implementation
- Demo
- Questions throughout and at the end

Problem Concepts:

What we know about Password based Authentication schemes

- Weak random seeds (rarely are random)
- Often predicable environment, which affects the password which user selects

Why?

- **“Human factor” based network security:** passwords, hidden folders, hidden interfaces – ***always there***
- **Exploiting *‘weakest link’’*:** vulnerabilities come and go: “human” mistakes remain
- **“bruteforce” methodology:** a need for some brain

Current Methodologies

- **Blind bruteforce** (static dictionary based)
- **“Intelligent” guessing:** some mechanisms available – specific dictionaries, checks for no passwords, same as username passwords

Issues

- **Extremely unproductive**
- **Often – ineffective (time constraints)**

How?

- We need to learn how typical users pick up their passwords
- Need to learn how policies affect it
- Detection of specific dictionary files could be automatic
- Analyze password selection patterns and reuse the knowledge

If…

.. sample material is available…

How a typical user head works..

What we can do

Use available environment and background information to make intelligent guess

… but we are not psychologists, so lets spot the basics

And let machine try to estimate the rest…

….using AI methods

AI: Why Genetic Algorithms

- Adaptive algorithms
- Our problem is not well understood
- Difficult to describe mathematically
- Large search space

Where It Will Work?

- **Compromise internal networks of large organizations (same group of users across multiple systems, same security policies, training/background)**
- **Large group of similar background/cultural level users (I.E. Dial-up users in Kyrgyzstan :p)**

GA theory shortly * (not mine!!)

The theory part is from „Genetic Algorithms and Evolution Strategies” presentation.

- **Presented by:**
- **Julie Leung**
- **Keith Kern**
- **Jeremy Dawson**

Genetic Algorithms

- **Closely follows a biological approach to problem solving**
- **A simulated population of randomly selected individuals is generated then allowed to evolve**

Encoding the Problem

- **Express the problem in terms of a bit string**

x = (1001010101011100)

where the first 8 bits of the string represent the X-coordinate and the second 8 bits represent the Y-coordinate

Basic Genetic Algorithm

- **Step 1. Generate a random population of *n* chromosomes**
- **Step 2. Assign a fitness to each individual**
- **Step 3. Repeat until *n* children have been produced**
  - **Choose 2 parents based on fitness proportional selection**
  - **Apply genetic operators to copies of the parents**
  - **Produce new chromosomes**

Fitness Function

- For each individual in the population, evaluate its relative fitness
- For a problem with *m* parameters, the fitness can be plotted in an *m*+1 dimensional space

Sample Search Space

- **A randomly generated population of individuals will be randomly distributed throughout the search sp**

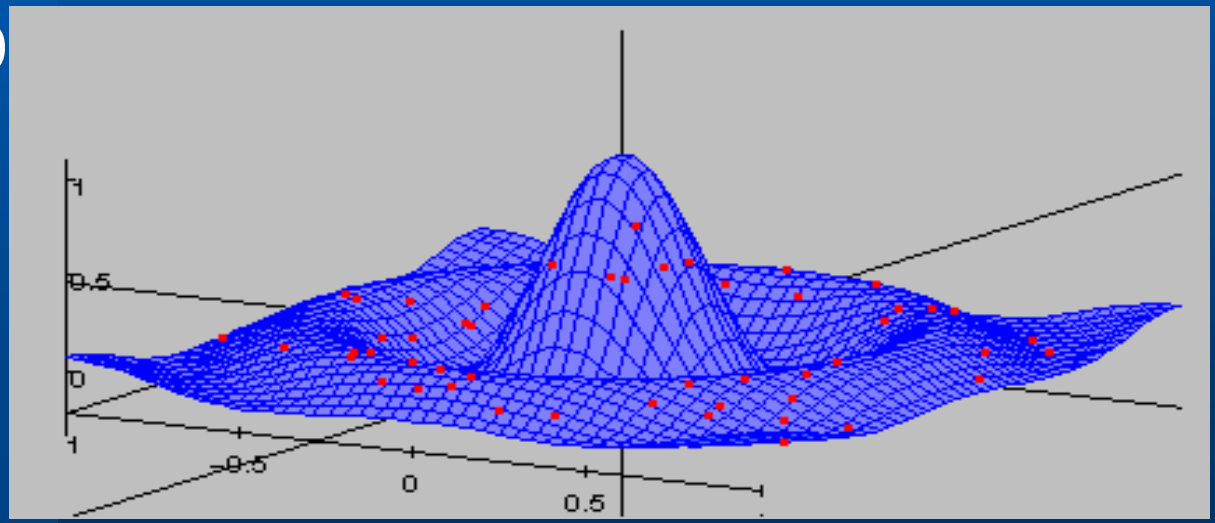


Image from http://www2.informatik.uni-erlangen.de/~jacob/Evolvica/Java/MultiModalSearch/rats.017/Surface.gif

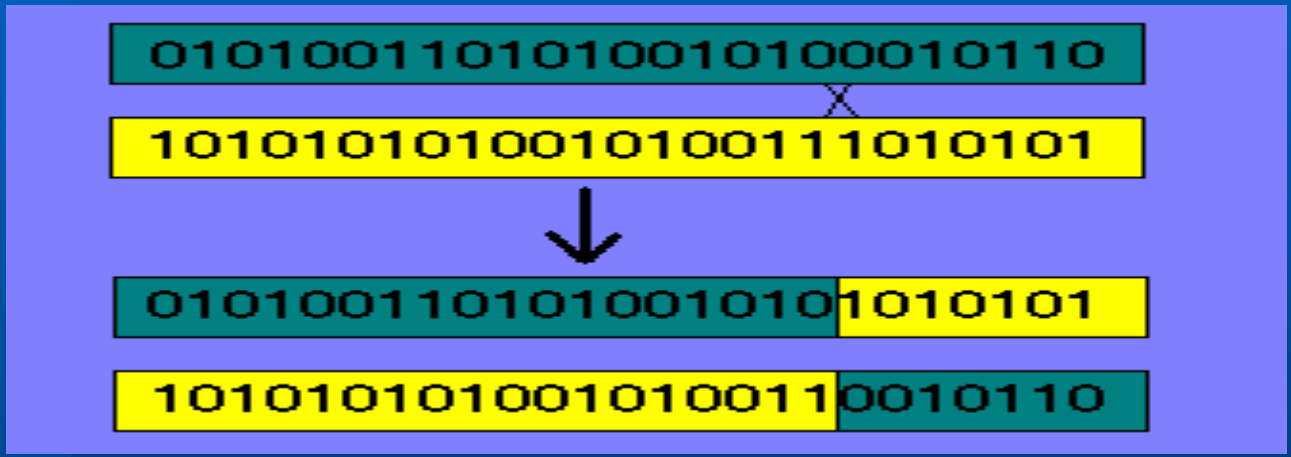
Natural Selection

- **The likelihood of any individual becoming a parent is directly proportional to its relative fitness**

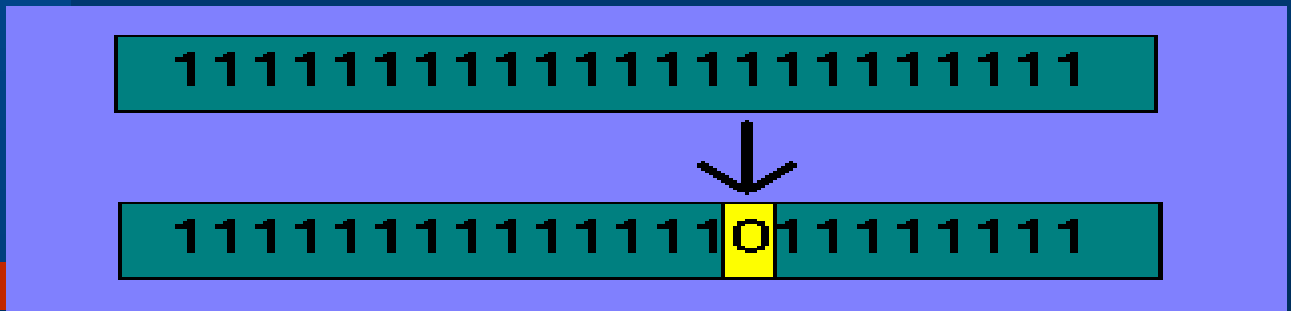
Image from http://www.genetic-programming.org

Genetic Operators

- **Cross-over**

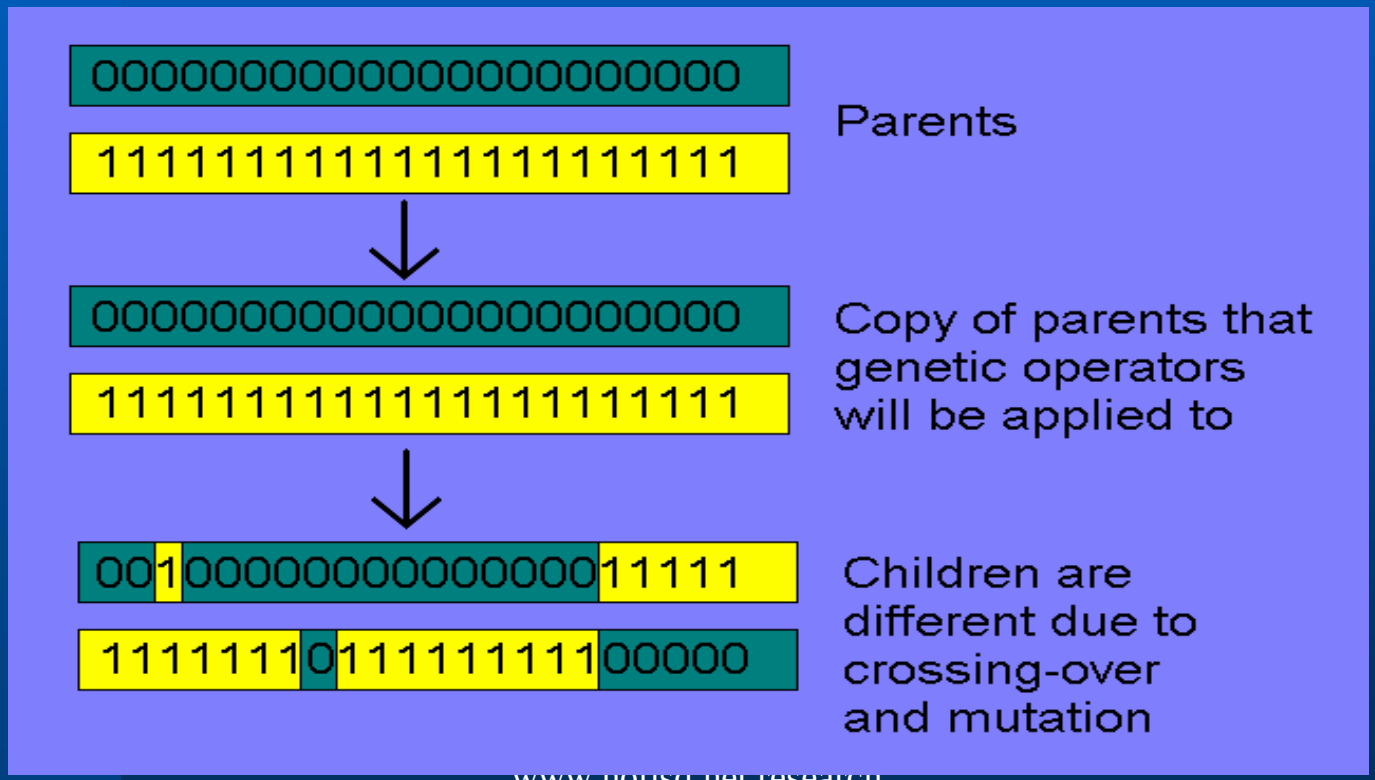


- **Mutation**



Production of New Chromosomes

- **2 parents give rise to 2 children**



Generations

- As each new generation of *n* individuals is generated, they replace their parent generation
- To achieve the desired results, 500 to 5000 generations are required

Ultimate Goal

- **Each subsequent generation will evolve toward the global maximum**
- **After sufficient generations a near optimal solution will be present in the population of chromosomes**

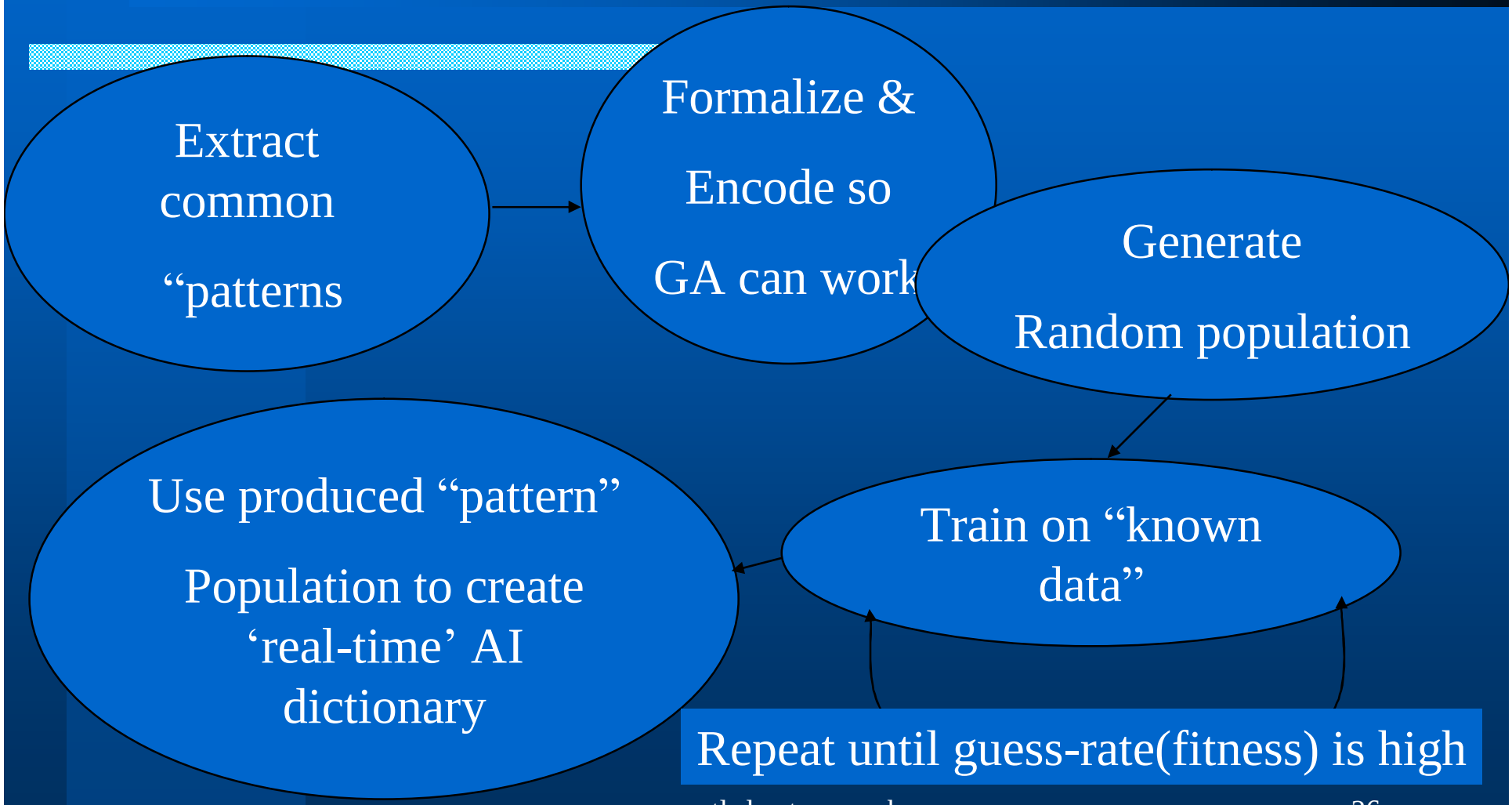
That’s it in short….

- **If something isn’t clear, ask now..**

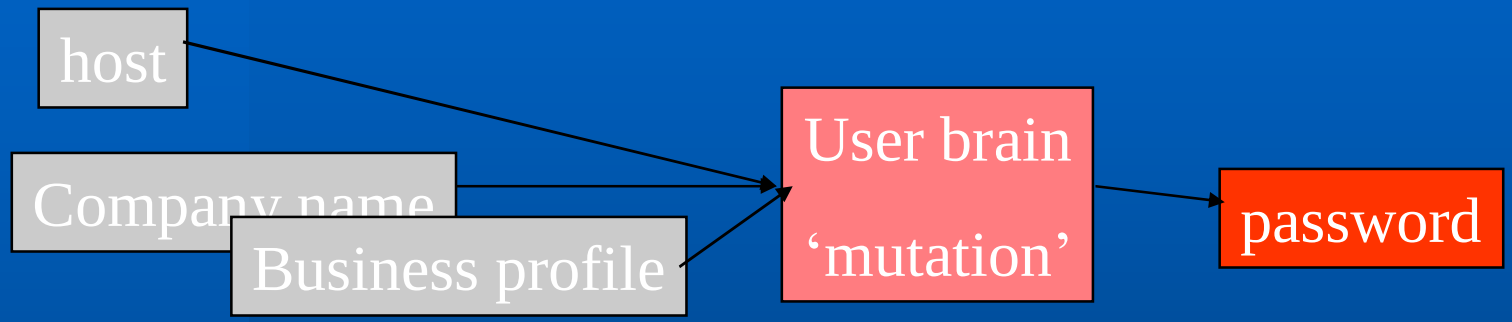
Methodology Breakdown

- Analyze users habits (manual)
- Formalize (manual)
- Generate population, train on the “known” data (automatic) until desired fitness achieved
- Use generated “population” to create “user brain” (run-time AI dictionary)

How it will work:

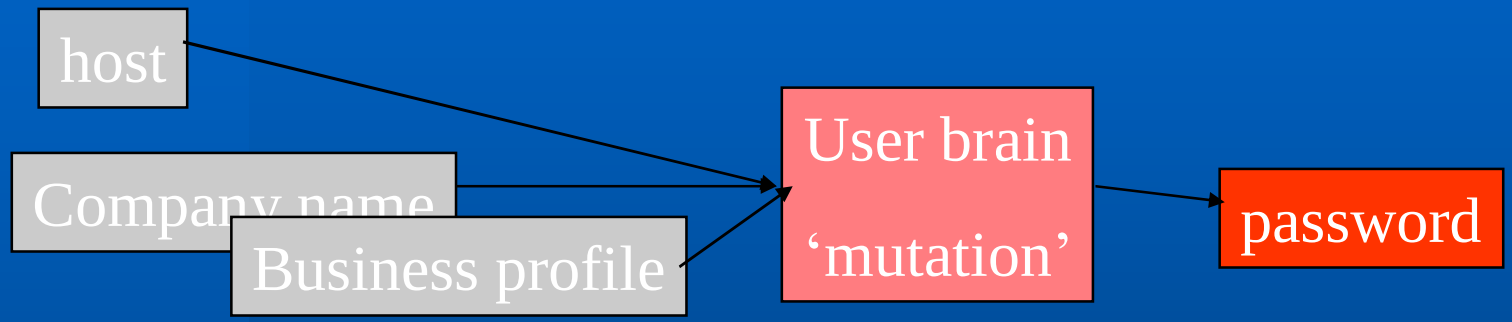


User habits:



- **Environment based (hostname, company, domain, platform, application, business purpose)**

User habits:



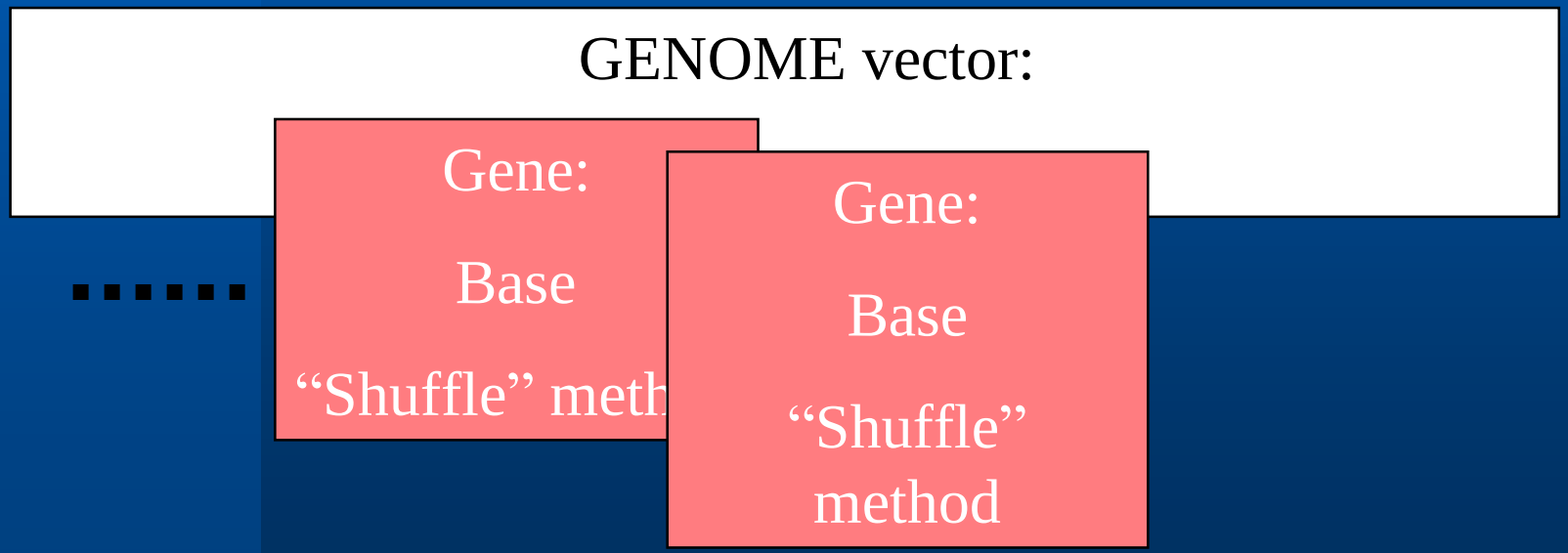
- **Personal ID based (user id, name, dates)**
- **Cultural (language specific dictionary, business profile specific dictionary)**

User habits

- **“shuffle” the base (character swap, capitalisation, “l33t” language etc)**
- **Combine “bases”**

Problem model design

- **Represent our problem as genome:**



Problem model design

- **Crossover:**

Gene exchange (AA + BB = AB BA

Gene drop (AA + BB = AA AB)

Gene combination AA + BB = AABB

- **Mutation:**

Random change of “base” or “shuffle” methods within Gene(s)

Fitness function

- $\Pr(h_i) = fitness(h_i) / \{S_{j=1..P}\ fitness(h_j)$.

- **Fitness function measures how the produced ‘string’ match the password (on number of characters).**

Practical Implementation

- **IBRUT: a toolkit and a library**

Principles

- Uses “Evolving objects” library for GA and evultional algorithms
- Implemented as a set of tools to gather “knowlegde” and as a library which will use “patterns” to patch your favourite brute-forcing tool

DEMO

- **No laughs!..**

Effectiveness compared with standard bruteforce tools

- **Standard dictionary of frequently used passwords**
- **Hydra checks**
- **John the ripper**

Future of “Intelligent guessing”

- SNMP community strings
- Web folders “blind” mapping
- Intelligent fuzzers
- …

Potential problems

- **Representation of the problem**
- **Trained data & computational time**
- **Hard to avoid redundancy in produced data**

Conclusion

- “new tool” -> “new knowledge”
- Fresh look on old problem

That’s about it…

Thanks !

Last chance to ask your questions ;)

www.notlsd.net research