Parameterizing Fingerprints to Protect Against “Sniff and Suppress” Attacks

Marcus Aqui and Terence Pocklington
Advisor: Dr. Leiss
Motivation

● Passwords: easy to generate and replace, near infinite possibilities
  ○ Do not guarantee authenticity of the user
● Biometrics: Unique to each person and guarantee authenticity
  ○ Not easily replaceable if compromised
  ○ Vulnerable to Sniff and Suppress attacks
Prototype a method of protecting fingerprint data in transit from sniffing attacks
Objectives

- Determine a method to parameterize fingerprint data
- Create a mathematical function to call on those parameterizations that will scramble a fingerprint, making it useless if intercepted without knowing the right key
- Develop prototype software that utilizes the function
Expected Impact

- Creation of a method of fingerprint parameterization allows for others to improve on current method and implement similar schemes in a real world environment
- Step towards securing biometrics for authentication
Deliverables

● C++ source and executable that will scramble and unscramble the internal representation of a fingerprint
● Various Bash scripts, including:
  ○ massUnscrambler and massScrambler: call the C++ executable with certain parameters
  ○ fingerprintMatcher: compares xyt files for matches, output stored in <testedFile>.txt
Methods: Objective 1

- Generate xyt files from fingerprint data with mindtct algorithm
- Read the documentation about the xyt files
- Set the xyt files as the parameter for the code
Results: Objective 1

- Read in each minutia, maximum, and minimum values for the x and y coordinates from the xyt files

(a) [Diagram of minutia patterns]

(b) [Diagram of linear patterns]
Methods: Objective 2

Using the given parameters in addition to the key, compute a formula that allows a wide range of output but reversible.
Results: Objective 2

- Developed two formulae, one for theta and one for the x and y coordinates.
- Each is based on adding or subtracting some constant to the original values, with the constant based on various attributes and the random numbers generated from key attributes. The new value is bounded by a range determined by the input xyt file.
Methods: Objective 3

- Write scripts that execute the bozorth3 algorithm, which compares fingerprint data
- Devise tests for scrambled print data vs original print data and unscrambled print data vs original print data
The bozorth3 script iterates through a set folder of print data and records the results in a txt file.
● Run more tests with different keys: determine factors of key quality
● Simulate more “Sniff and Suppress” attacks
● Collect all the separate scripts and executables into a single program
Conclusions

- Based on the current test results, the function proves successful: the 11 keys all produced scrambled versions of the fingerprint that did not match the original versions, but whose unscrambled forms were identical to the original version.
- If unscrambled with the wrong key, resulting fingerprint is not a match to originals.
The REU project is sponsored by NSF under award NSF-1659755. Special thanks to the following UH offices for providing financial support to the project: Department of Computer Science; College of Natural Sciences and Mathematics; Dean of Graduate and Professional Studies; VP for Research; and the Provost's Office. The views and conclusions contained in this presentation are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.