

# Fingerprint-based Fuzzy Vault

**Umut Uludag and Anil K. Jain**

**Department of Computer Science and Engineering  
Michigan State University**

**{uludagum, jain}@cse.msu.edu**

**<http://biometrics.cse.msu.edu>**

# Overview

- **Introduction**
  - **Biometrics & Cryptography**
- **Previous Work**
  - **Fuzzy Vault**
- **Proposed System**
  - **Encoding**
  - **Decoding**
- **Experimental Results**
  - **Genuine Accept Rate (GAR)**
  - **False Accept Rate (FAR)**
- **Conclusions**

# Introduction

- Traditional cryptography:

- Encryption:  $C = E_{KE}(P)$

- Decryption:  $P = D_{KD}(C)$

- P: plain text, C: cipher text, KE & KD: keys

- Without KD, it is *infeasible* to convert C to P

- Current algorithms:

- AES, RSA, 3DES...

- High, proven security: They are based on “hard” problems

- Assumption: Keys should be kept secret

- Need long, random keys: 128 bits (AES)

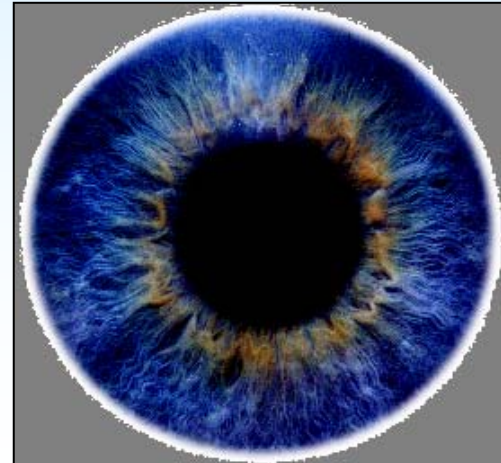
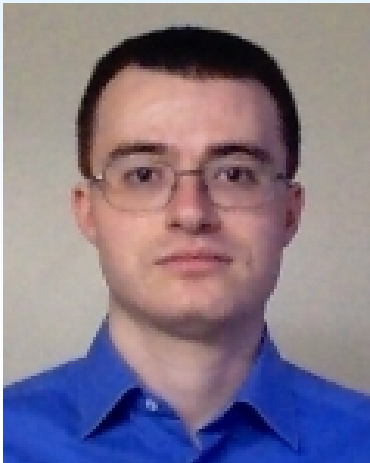
- Since users can not memorize them, they are stored (e.g., smart card) using simple passwords

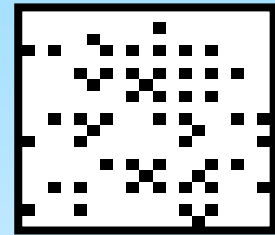
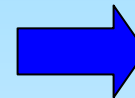
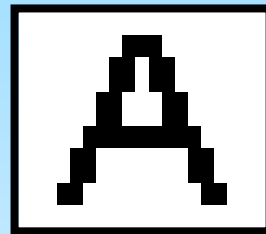
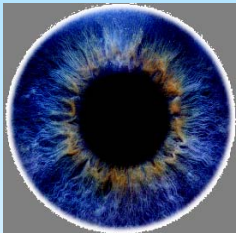
- **Password-based authentication:**

- Simple passwords are easy to remember; compromise security
- Complex passwords are difficult to remember; expensive to maintain

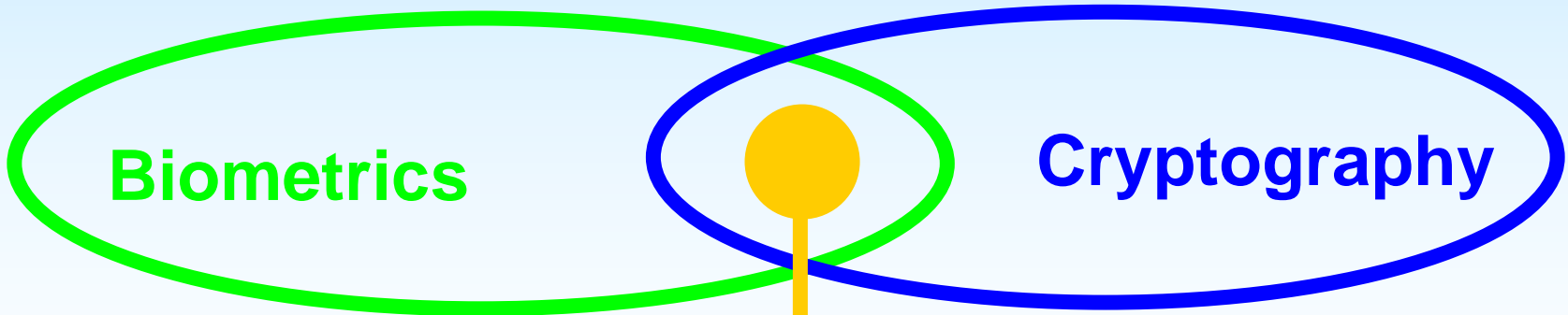
- **Alternative → Biometric authentication:**

- Using anatomical and behavioral traits (e.g., fingerprint, iris) for authentication
- **More reliable:** can not be lost or forgotten; difficult to copy and share; provides non-repudiation





Key: 01001101010110...



**Crypto-biometric systems**

# Fuzzy Vault

- Juels and Sudan (2002):
  - Alice places a secret  $K$  in a vault and locks it using an unordered set  $A$
  - Bob uses an unordered set  $B$  to unlock the vault (access  $K$ ): successful iff  $B$  and  $A$  overlap **substantially**

## Alice locking her phone number into the vault:

Alice:

$K = 555\ 4321$  +

$A = \{ \text{Rambo 1,} \\ \text{Rambo 2,} \\ \text{Rocky 2,} \\ \text{Spider-man} \}$



555 4321

## Unlocking attempts:

Bob: succeeds

555 4321

+

$B = \{ \text{Rocky 2,} \\ \text{Cat-woman,} \\ \text{Rambo 1,} \\ \text{Rambo 2} \}$



555 4321

⇒ "Fuzzy" vault

Charlie: fails

555 4321

+

$C = \{ \text{Titanic,} \\ \text{Love Actually,} \\ \text{Casablanca,} \\ \text{Annie Hall} \}$



555 4321

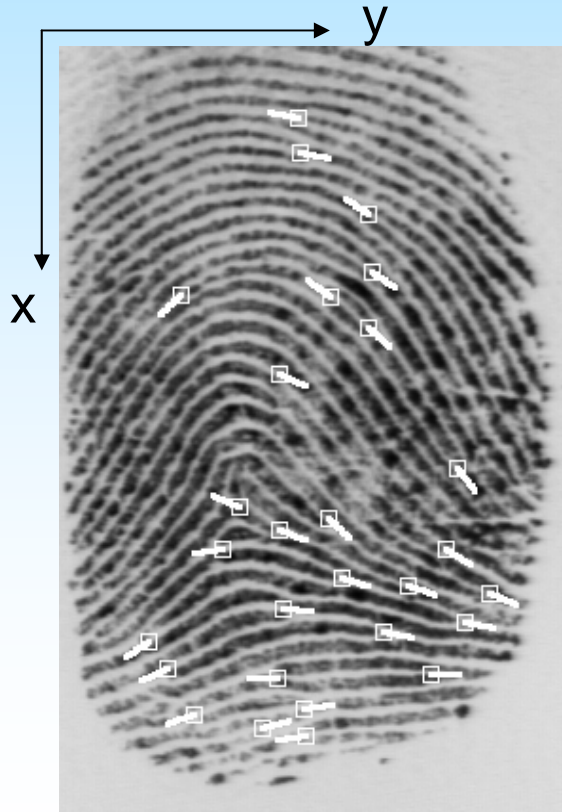
- Locking the vault:
  - Alice selects a polynomial  $p$  of variable  $x$  encoding  $K$
  - Computes polynomial projections  $p(A)$
  - Adds randomly generated chaff points to get point set  $R$ 
    - chaff points are necessary for hiding the secret  $p$
- Unlocking the vault:
  - Bob uses his own set  $B$
  - If  $B$  and  $A$  are similar, many points of  $R$  will lie on  $p$ 
    - Using error-correction coding, he can reconstruct  $p$  (hence  $K$ )
- Security: Based on infeasibility of polynomial reconstruction
- Appropriate for biometric data:
  - Works with unordered sets (e.g., fingerprint minutiae)
  - Tolerates differences between  $A$  and  $B$

- Clancy et al. (2003): Fingerprint Vault

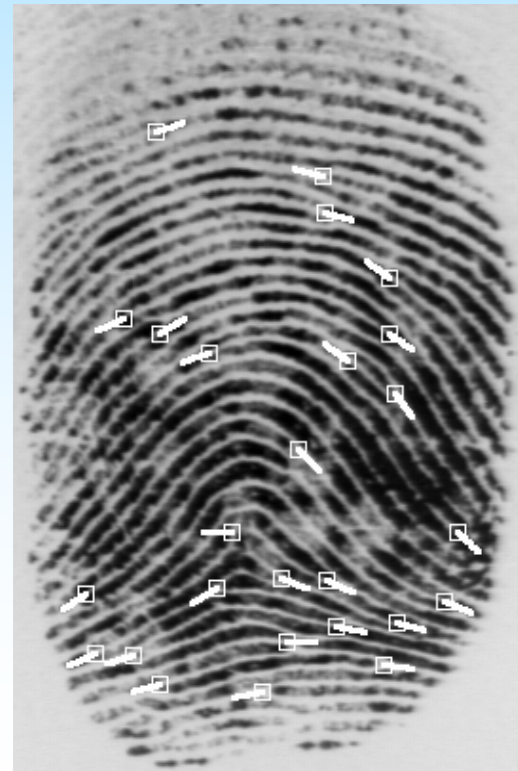
- Using multiple minutiae location (x,y) sets; canonical minutiae positions are found and used as elements of **A**
  - minutiae angles are not used
  - performance: FAR =  $10^{-20}$  at 0.2 FRR
  - data acquisition characteristics are not known
  - error-correction is simulated
- **Assumption:** Fingerprints used for locking and unlocking the vault are *pre-aligned*.

# Proposed System

- Fingerprint minutiae locations  $(x,y)$  are used for locking and unlocking the vault:
  - Concatenation of two 8-bit quantities gives 16-bit locking/unlocking unit:  $u = x|y$
  - All arithmetic is done in Galois field  $GF(2^{16})$



Template (28 minutiae)

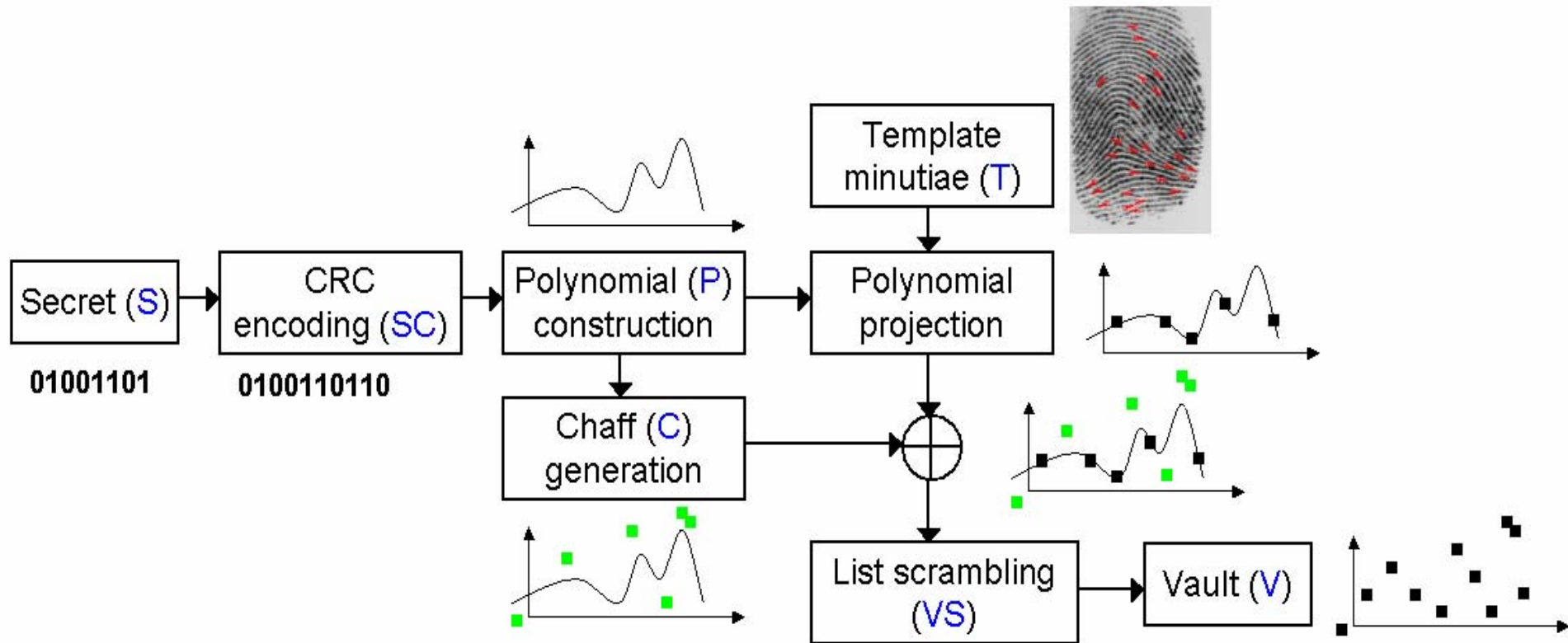


Query (26 minutiae)

## Encoding:

- Use minutiae (**T**) to secure secret (**S**): transform the secret to a 2D point set (**V**), using projection of polynomial  $P=f(S)$  on minutiae points
  - **S**: encryption key, SSN, password...
- Minutiae generate *genuine* points (**G**) in the set. For increased security, **M chaff** points (**C**), that do not pass through  $P$ , are also added to the set (**V**)
  - CRC (Cyclic Redundancy Check) bits are added to **S** to detect errors during decoding
    - Primitive polynomial:  $g(a) = a^{16} + a^{15} + a^2 + 1$
- **V** is scrambled to prevent information leak

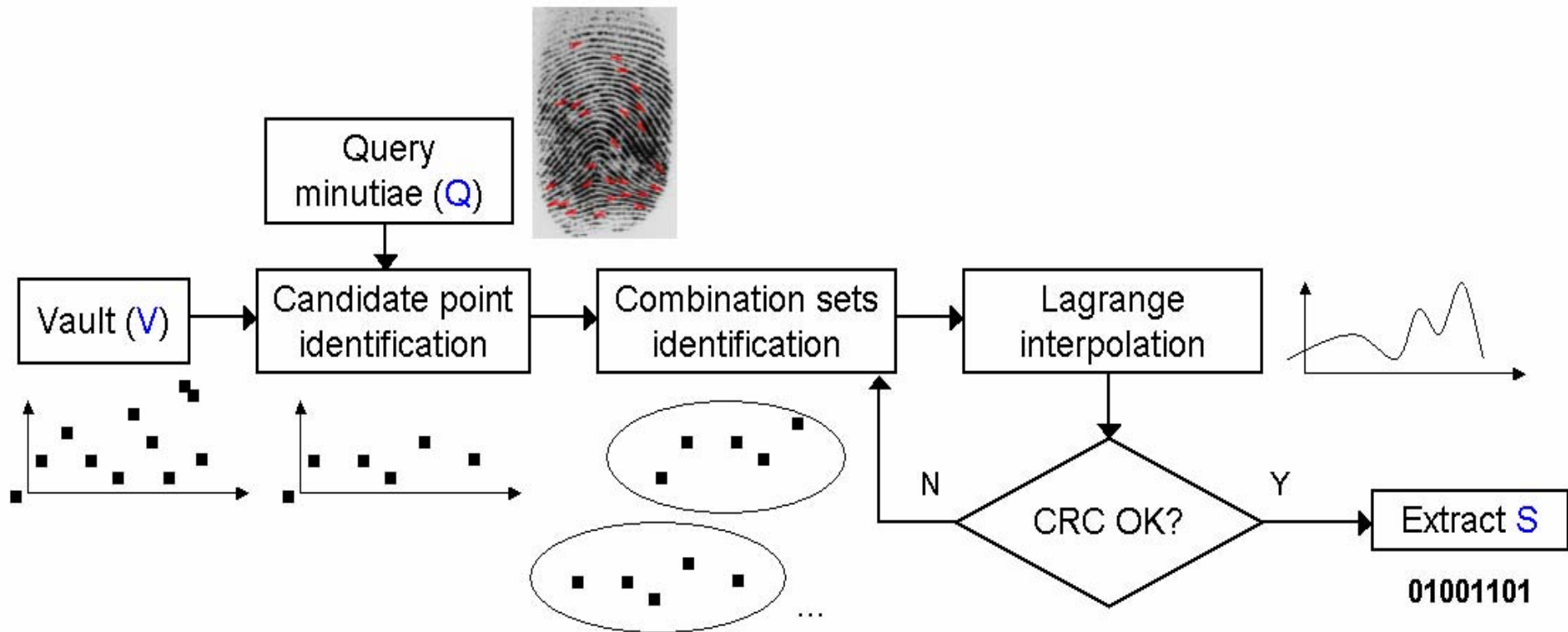
# Encoding: Block Diagram



## Decoding:

- Based on polynomial reconstruction problem: solve for the degree  $D$  polynomial  $P$ , given  $D+1$  points passing through it.
  - Any set with at least  $D+1$  points will suffice
  - A genuine finger can separate at least  $D+1$  genuine points from chaff points, and reconstruct the polynomial  $P$ , allowing access to  $S$ 
    - Lagrange interpolation decodes the polynomial  $P$
    - Multiple point combinations need to be evaluated  
→ CRC detects errors in decoding
  - An imposter finger can not separate sufficient number of genuine points  $D+1$  for reconstruction, so she can not access the secret  $S$ .

# Decoding: Block Diagram



## Encoding: Details

- S: 128-bit AES key
- No error-correction scheme is used; decoding produces multiple candidate secrets, whose accuracy is checked using 16-bit CRC.
- Final payload is SC = S | CRC (128+16=144-bits)
- To account for variations in detected minutiae, minutiae are first quantized (2D block size: 7x7)
- 144-bit SC can be represented as a polynomial with 9 (144/16) coefficients in  $GF(2^{16})$ , with degree  $D=8$ :

$$p(u) = c_8u^8 + c_7u^7 + \dots + c_1u + c_0$$

- **Genuine set G**: evaluating  $p(u)$  on the template minutiae features (T).
- **Chaff set C**: points that **do not** pass through  $p(u)$

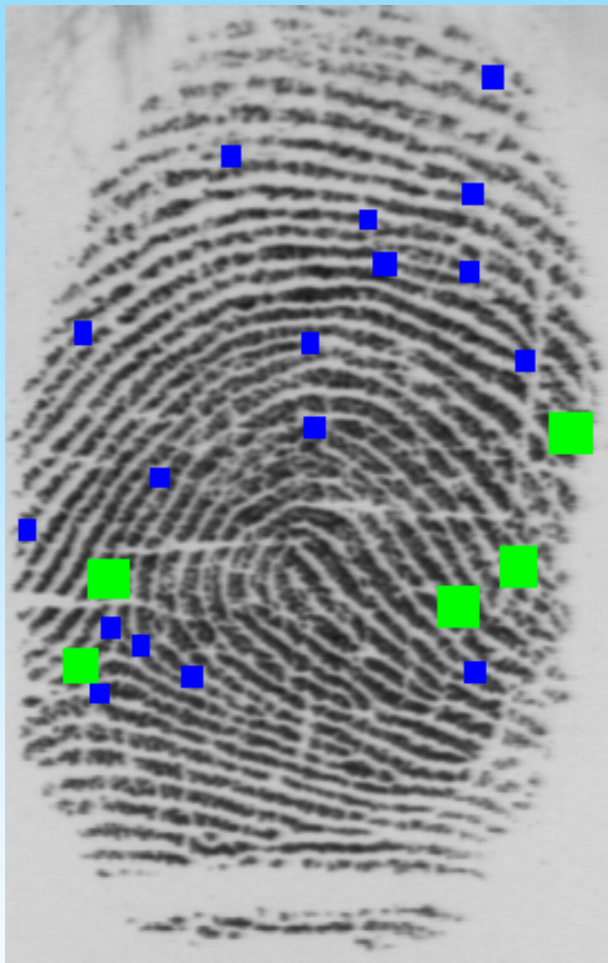
## Decoding: Details

- A user tries to unlock the vault using the query minutiae:
  - Assuming there are  $N$  query minutiae (Q):  $u_1^*, u_2^*, \dots, u_N^*$ 
    - Candidate points to be used in decoding are found by comparing  $u_i^*, i = 1, 2, \dots, N$  with the abscissa values of the vault  $V: v_l, l = 1, 2, \dots, (M + N)$  ( $M$ : # of chaff points)
      - If any  $u_i^*, i = 1, 2, \dots, N$  is equal to  $v_l, l = 1, 2, \dots, (M + N)$  the corresponding vault point is added to the list of points to be used
      - Assume that this list has  $K$  points, where  $K \leq N$ .
- For decoding a  $D$ -degree polynomial,  $(D+1)$  unique projections are necessary. We find all possible combinations of  $(D+1)$  points, among the list with size  $K$ , resulting in  $C(K, D+1)$  combinations.

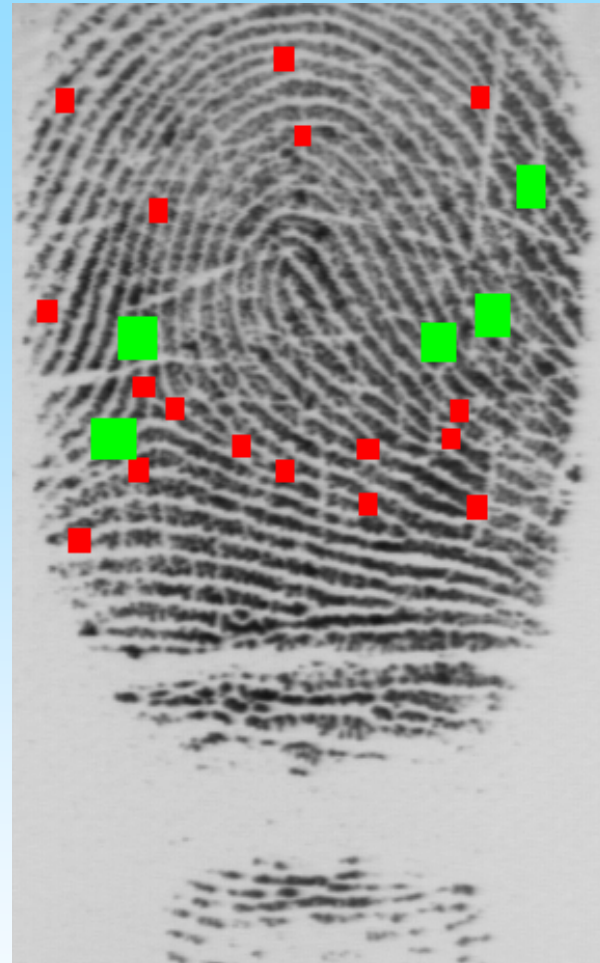
# Experimental Results

- IBM-GTDB database (229 fingerprint pairs, 500 dpi, image size ~300x400 pixels)
  - Minutiae correspondences marked by a human expert
    - 16-bit locking/unlocking units,
    - S: 128-bit AES key,
    - CRC: 16-bit
    - Degree 8 polynomial P
    - #genuine points = 22,
    - #chaff points = 200

- GAR is 0.85:
  - 34 out of 229 query templates **could not** separate sufficient no. of ( $\geq 9$ ) genuine points from chaff points
- False Accept Rate (FAR) is 0%
  - There are 228 imposter unlocking attempts for a distinct finger; a total of 52,212 ( $228 \times 229$ ) attempts for 229 users.
  - The unlocking templates are first aligned with the locking templates
  - **None** of the 52,212 attempts could unlock the vault.



template fingerprint  
(marked: 22 locking minutiae,  
**5 common**)



query fingerprint  
(marked: 22 unlocking minutiae,  
**5 common**)

False Reject: Only 5 genuine points found during decoding

# Conclusions

- We have implemented a fuzzy vault using fingerprint minutiae, without resorting to simulating the error-correction step.
- The vault performs reasonably well and the complexity of attacks that can be launched by imposters is high.
- A 128-bit AES keys can be secured using the proposed vault.
- Limitations: high time complexity and high FRR
- **Future work:**
  - Automatic alignment within the fuzzy fingerprint vault
  - Decreasing FRR
  - Increasing capacity (e.g., supporting 192-bit AES keys)
  - Decreasing time-complexity (e.g., error-correction that allows automatic alignment)

N template minutiae →

$$G = \{(u_1, p(u_1)), (u_2, p(u_2)), \dots, (u_N, p(u_N))\}$$

The template minutiae are selected to be unique:

$$u_i \neq u_k, \text{ if } i \neq k, i = 1, 2, \dots, N, k = 1, 2, \dots, N$$

M chaff points are needed →

- Generate M unique random points  $(c_1, c_2, \dots, c_M)$ , in the field  $GF(2^{16})$ , with the constraint that they do not overlap with  $u_1, u_2, \dots, u_N$ :

$$c_j \neq u_i, j = 1, 2, \dots, M, i = 1, 2, \dots, N$$

- Generate another set of M random points  $(d_1, d_2, \dots, d_M)$ , with the constraint that the pairs  $(c_j, d_j), j = 1, 2, \dots, M$  do not fall onto the polynomial  $p(u)$ :

$$C = \{(c_1, d_1), (c_2, d_2), \dots, (c_M, d_M)\} \quad \text{where} \quad d_j \neq p(c_j), j = 1, 2, \dots, M$$

- Union of these two sets,  $G \cup C$ , is passed through a list scrambler which randomizes the list, with the aim of removing any stray information that can be used to separate chaff points from genuine points. This results in vault set:

$$VS = \{(v_1, w_1), (v_2, w_2), \dots, (v_{N+M}, w_{N+M})\}$$

- Along with VS, the polynomial degree D forms the final vault, V.

- For a specific combination set given as  $L = \{(v_1, w_1), (v_2, w_2), \dots, (v_{D+1}, w_{D+1})\}$  the corresponding interpolating polynomial is (Lagrange method)

$$p^*(u) = \frac{(u - v_2)(u - v_3)\dots(u - v_{D+1})}{(v_1 - v_2)(v_1 - v_3)\dots(v_1 - v_{D+1})} w_1 + \frac{(u - v_1)(u - v_3)\dots(u - v_{D+1})}{(v_2 - v_1)(v_2 - v_3)\dots(v_2 - v_{D+1})} w_2 + \dots$$

$$\dots + \frac{(u - v_1)(u - v_2)\dots(u - v_D)}{(v_{D+1} - v_1)(v_{D+1} - v_2)\dots(v_{D+1} - v_D)} w_{D+1}$$

- This calculation is done in  $GF(2^{16})$  and yields

$$p^*(u) = c_8^* u^8 + c_7^* u^7 + \dots + c_1^* u + c_0^*$$

- The coefficients are mapped back to the decoded secret  $SC^*$ :
  - We divide the polynomial corresponding to  $SC^*$  with the CRC primitive polynomial  $g(a) = a^{16} + a^{15} + a^2 + 1$ 
    - If the remainder is not zero, we are certain that there are errors.
    - If the remainder is zero, with very high probability, there are no errors.  $SC^*$  is segmented into 2 parts: the first 128-bits denote  $S^*$  while the remaining 16-bits are CRC data.  $S^*$  is output.