Carnegie Mellon

Hearing without Listening

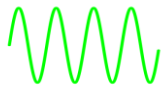
Bhiksha Raj (CMU)

Co-authors**: **Manas Pathak (Walmart Labs)
Jose Portelo (Recently graduated from INESC, Portugal)
Abelino Jimenez (Carnegie Mellon University)
Isabel Trancoso (INESC, Portugal)
Shantanu Rane (Xerox Parc)
Petros Boufounos (MERL)
Paris Smaragdis (UIUC)
Madhu Shashanka (PetaSecure – company working on enterprise security)

Recall

- Secure multi-party computation can be used to protect voice
- Enables the computation required for pattern matching without revealing the Alice's voice to the system Bob or Bob's system parameters to Alice
- Lets consider some real speech problems

A Brief Primer on Speech Processing Tasks



Biometrics

Speaker Verification

Speaker Identification

Speech Recognition

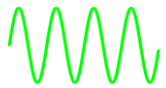
Are you really Bob? Yes/No (binary)

Which one of Alice, Bob, Carol, Dave, … **are you?** (multi-class)

You said ”hello, world!”

- All are pattern classification tasks

Privacy Challenges in Biometrics



Are you really Bob? Yes/No (binary)

Which one of Alice, Bob, Carol, Dave, … **are you?** (multi-class)

You said ”hello, world!”

Most common application: Verification

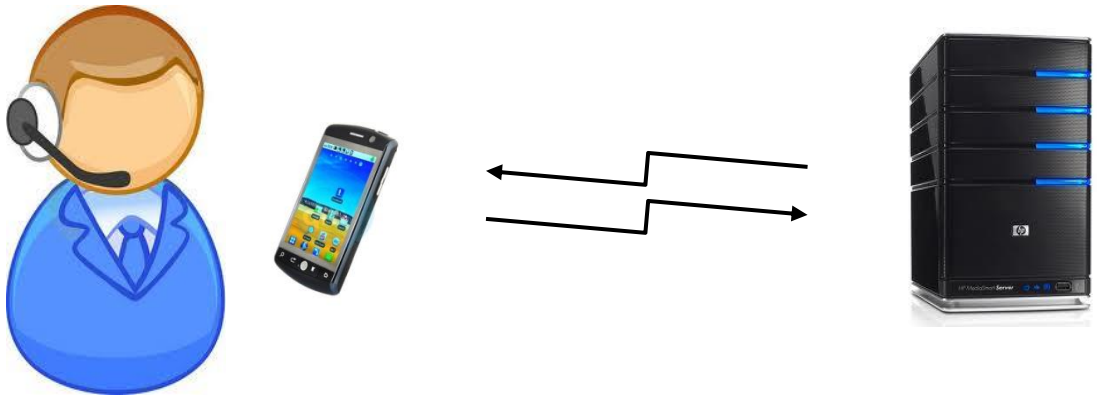
$$\hat{C} = \arg\max_{C \in \{\text{Target}, \text{Imposter}\}} Score(X; \Lambda_C)$$

- Task: Verify if a speaker is who he/she claims to be
- Setup: Speaker registers with the system in a registration phase
  - System builds model for speaker
- Operation: Speaker says something to the system
  - System compares recording to stored model

Verification: Privacy challenge

- Speaker does not trust system with his speech
  - System (or hacker) may abuse recordings
- Speaker does not trust system with model
  - System (or hacker) may abuse model
    - To detect speaker elsewhere
    - To impersonate speaker
- System must still verify the speaker accurately

Operational Setup

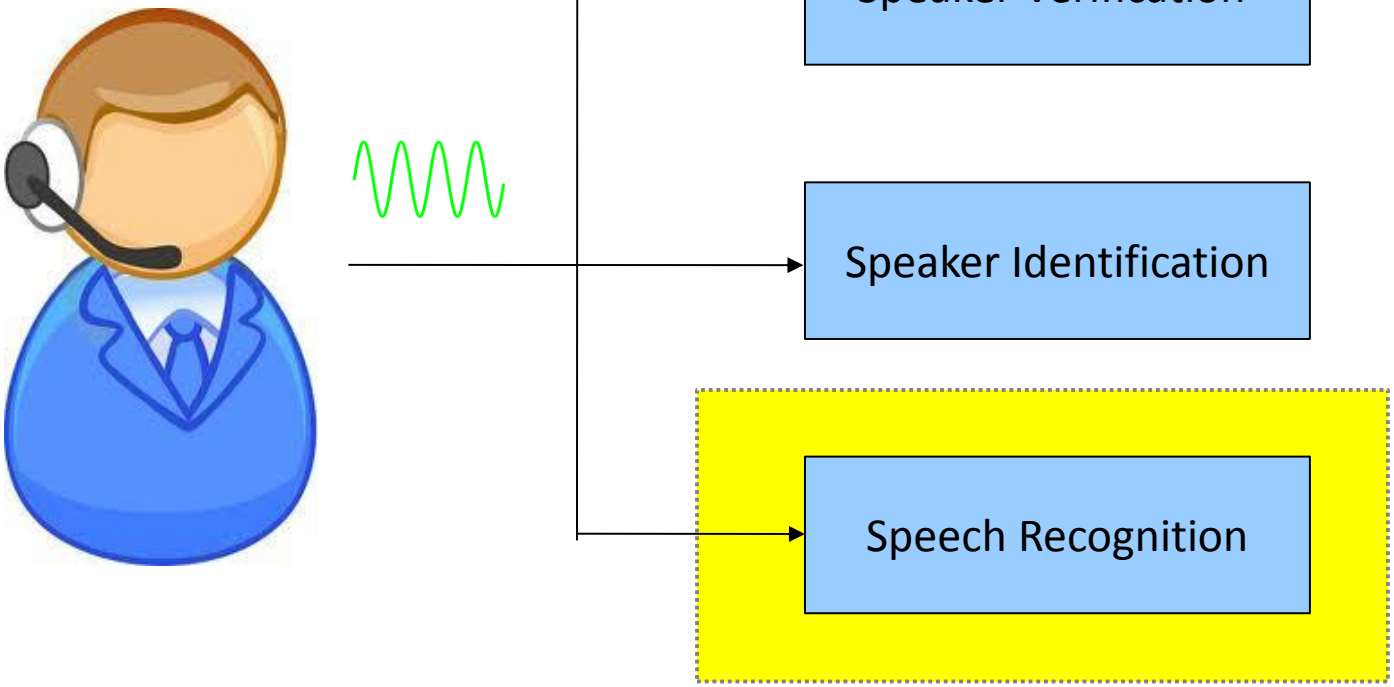


- User has a smart phone or computation capable device
  - Communicates with server using this device
- Encryption key maintained by device or user

Verification: SMC solution

- During registration system only sees *encrypted* version of user's speech
- System only retains an *encrypted* version of the model for the speaker
- System only receives encrypted version of speech during operation

Privacy Challenges in Recognition



Recognition Challenges

- Speech source distinct from recognizer
- E.g. User using remote recognition service
  - Recognizer must not see speech
  - Recognizer must not know result

- E.g. Data mining agency searching remote speech corpus for patterns
  - Recognizer must not see speech
  - Data corpus server must not see recognition result

Recognition: SMC Solution

- Speech remains encrypted
- Results are only available to the appropriate entity
  - User using remote service: Only user gets recognition output
  - Mining Surveillance: Mining agency only gets to know if target pattern was found

Privacy-Preserving Solutions

- SMC / Homomorphic encryption based solutions possible in all cases
  - Although details have not been worked out for *all* problems
- But how efficient are they?
  - Public key encryption is very expensive
  - Communication overhead is high
  - How expensive… ?

An Isolated Word Recognition Task

Activity	256-bit keys	512-bit keys	1024-bit keys
Alice encrypts speech	253 sec	1945 sec	11045 sec
Bob computes P(X|s) per HMM	80 sec	230 sec	461 sec
Bob compute P(X) per HMM	16 sec	107 sec	785 sec

- 10-word (digit) isolated word recognition task
  - HMM based recognizer
- 3.2GHz Pentium 4
- Time taken for computing the scores of all models *per second* of speech
  - Does not include communication overhead
  - Paillier cryptosystem
- Greater security with larger keys..

- Recognition accuracy identical to that obtained without privacy worries
  - Regular computation

A Speaker Verification Task

- Using primitives-based SMC

Steps	Time (256-bit)	Time (1024-bit)
Encrypting $\bar{x}_t$ $\forall t$	138 s	8511 s
Evaluating Adapted	97 s	1809 s
Evaluating UBM	same as adapted	same as adapted
Comparison	0.07 s	4.01 s
Total $=E[\bar{x}_t]$ + adapted + UBM + compare	331 s $\sim$ 5.47 min	12133 s $\sim$ 3 hr, 32 min

- Computation details per second of incoming speech
  - Core-2 duo, 2 GHz
  - BGN cyrptosystem (Paillier an order of magnitude faster)
  - Does not include communication overhead
- "Insecure" computation: Negligible time
- Classification accuracies indistinguishable from insecure version

Other Possible Contributions to Cost

- Have not considered many conventional processing steps
  - Speech Rec: *Pruning* reveals information
    - Not pruning makes recognition computationally infeasible for most tasks
  - Verification: Factor analysis methods add complexity
- Protecting against malicious users, e.g. through zero-knowledge proofs, is *much* more expensive
- Training on private data (e.g. for verification) is particularly expensive

Efficiency

- More efficient implementations and protocols possible
  - 10 to 100x faster
  - Techniques based on garbled circuits can be very fast
  - *STILL TOO SLOW*
- Better homomorphic cryptoschemes?
  - Don’t expect dramatic improvements anytime soon

The 1-mile view

- Privacy-preserving pattern matching tasks are possible
  - Work required on improving security under malicious model
- Have seen a “Cryptographic” approach
  - Based on Encryption
  - “Correctness” based – result with “secure” computation must be identical to that with regular computation
- But computationally infeasible
- Do we have an alternate approach?..

An Alternate Perspective

Encryption-based Mechanisms

- Encryption-based techniques attempt to retain *all* information
  - Final computation must be identical to what we would obtain without encryption
- Encryption-based techniques assume *everything* about the speech must be hidden
  - No inference must be possible at all outside of pattern matching results

Alternate Setup: *Usable* Privacy

- Do not need perfect Encryption
  - Just enough to discourage
- Do not need identical results to non-private computation
  - Similar results will suffice..
- In following slides, use speaker verification as platform

Password based schemes

- Text passwords are safe and efficient
  - Highly secure
  - Near instantaneous response
- Reason: *Based on exact match*
  - System stores text password encrypted by a one-way hash function
    - E.g. SHA-*
    - Even the *system* cannot decrypt
  - Incoming passwords are encrypted identically
  - Encrypted incoming password is matched to stored encrypted password

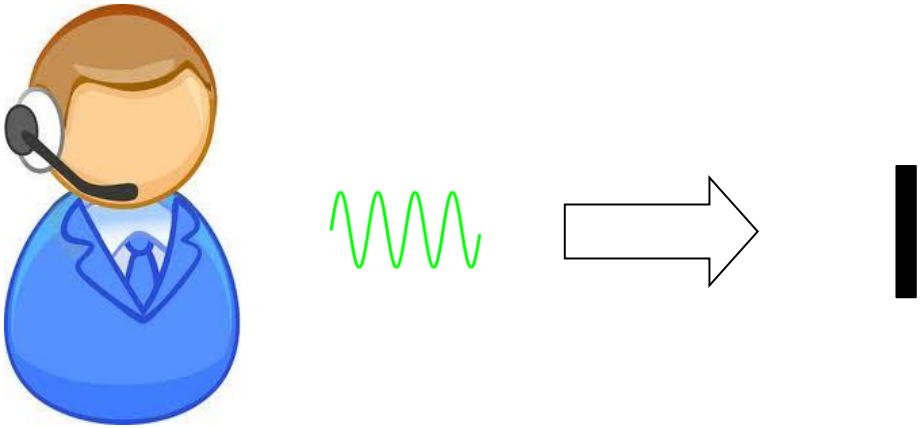
- **Cryptographic hash functions are extremely fast to compute**
  - Can we use a similar process?

Speaker Verification as String Matching

Simple approach:

Convert speech into a “password”

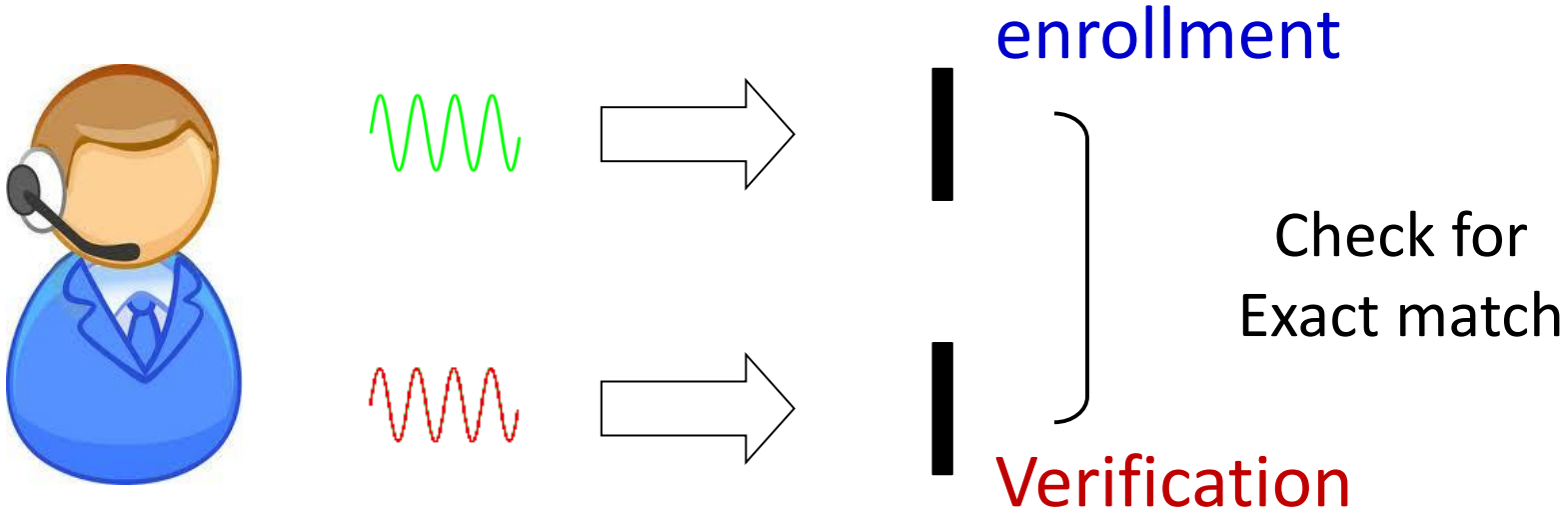
- ***Uninformative*** fixed-length bit string



Similar to password systems

Speaker Verification as String Matching

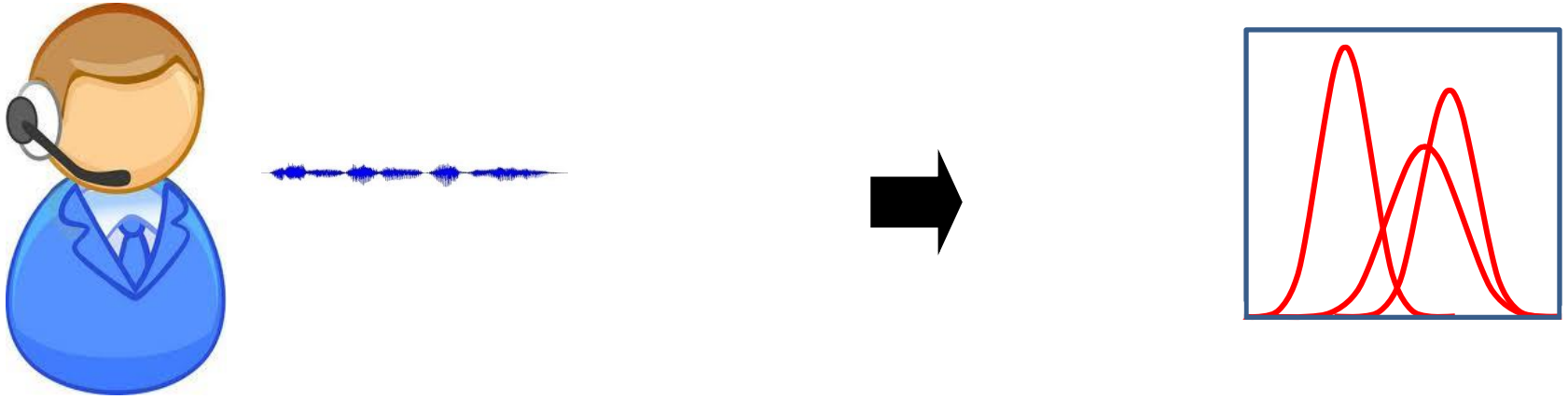
Speaker Verification by comparing bit strings



- **Problems**:
  - How do we convert speech to fixed length bit string?
    - Speech recordings vary in length
  - How to work with exact match?
    - Enrollment and test recordings never identical

Converting Speech to Fixed-Length Representations: Supervectors

- Typical approach: Estimate the distribution of the measurements in the recording



Concatenate the distribution parameters

supervector $s = (\mu_1 \,||\, \ldots \,||\, \mu_M)$

Supervectors

- The Supervector for any recording represents the distribution of feature vectors in the recording
- The length of the supervector is fixed, regardless of the length of the recording
- I.e. a fixed-length representation of the recording

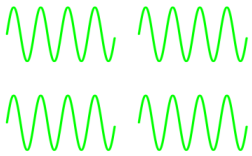
Verification: Modified Approach

- System obtains multiple enrollment recordings $\mathbf{X}_1$, $\mathbf{X}_2$, … from the speaker
- System generates supervectors S_1, S_2.. for each enrollment recording

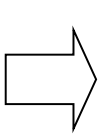
- System obtains a collection of *imposter* recordings
- System generates supervectors I_1, I_2, ... for each imposter recording

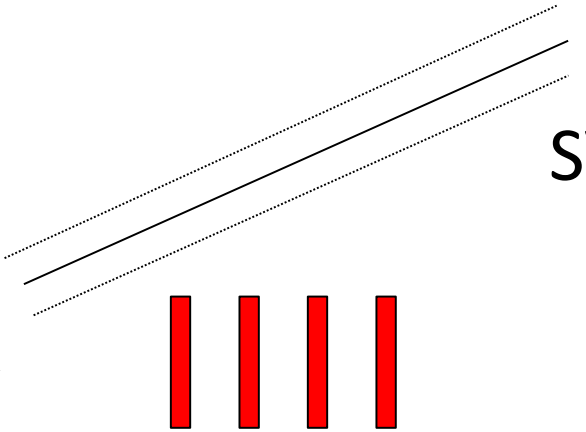
Verification by Supervectors

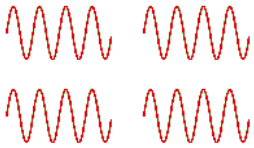
Train SVM classifier across the speaker & imposter supervectors



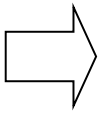
speaker







imposter



For a “Password” Version

- Supervectors are good fixed-length representations of the audio
  - But are informative
- Can they be converted somehow to uninformative “password” strings?
  - On which we can expect *exact match* for authentication?
- Recall
  - **Text-based password systems use cryptographic hashes..**

Locality Sensitive Hashing

- Locality sensitive hashing [Indyk & Motwani, 1998] is a method of mapping data to bit strings or *keys*:

  $X \rightarrow H(X)$

- It has the following property:
  - The hash keys H(X) and H(Y) of X and Y are identical with high probability if d(X,Y) is small (for some distance function d())
  - H(X) and H(Y) are different with high probability if d(X,Y) is large
- The function H(X) depends on the definition of d(X,Y)

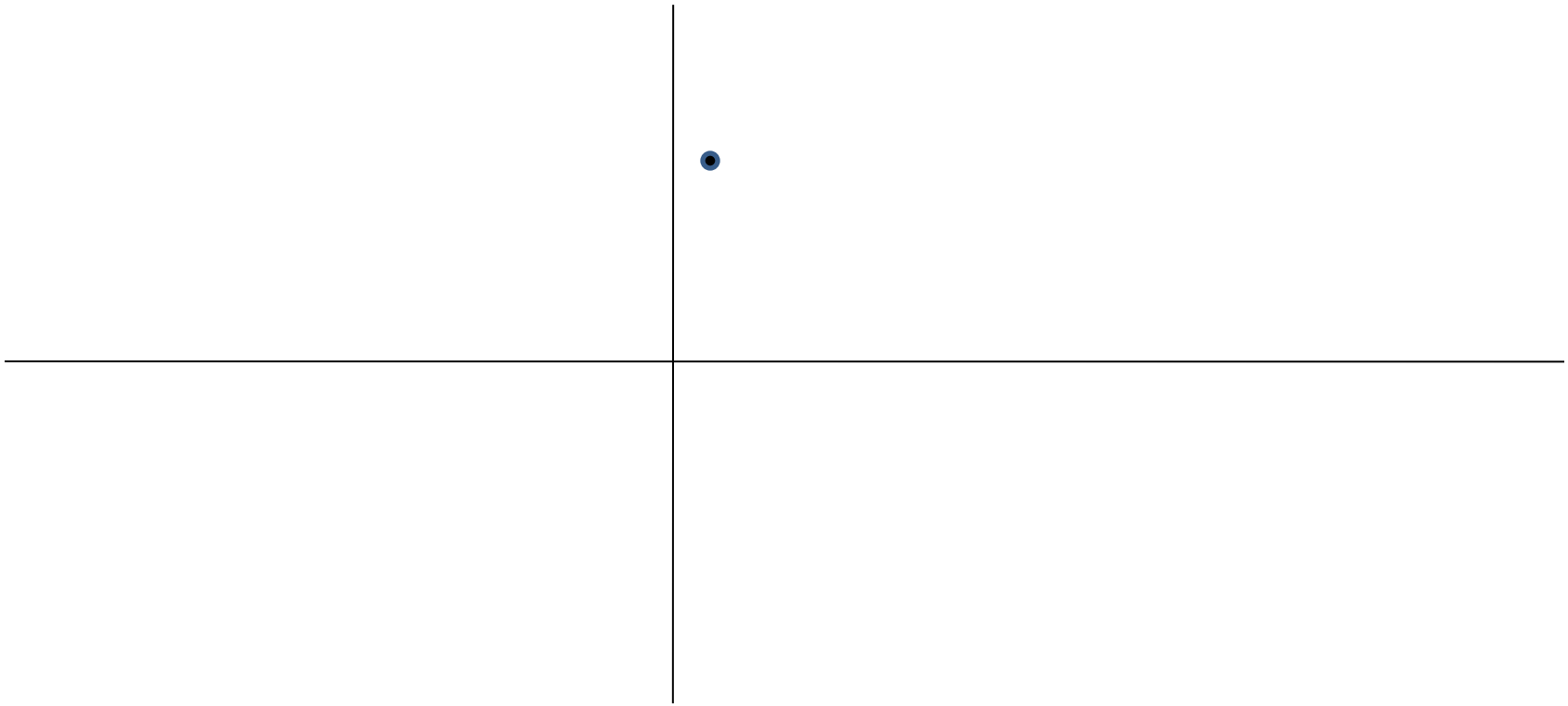
LSH with Euclidean Distance

- A vector X gets converted to a vector of M numbers $\mathrm{H(X)} = [\mathrm{h_1(X)}\ \mathrm{h_2(X)}\ \mathrm{h_3(X)}\ \ldots\ \mathrm{h_M(X)}]$

$$h_i(X) = h_i(X; V_i, b_i) = \left\lfloor \frac{X^T V_i + b_i}{w} \right\rfloor$$

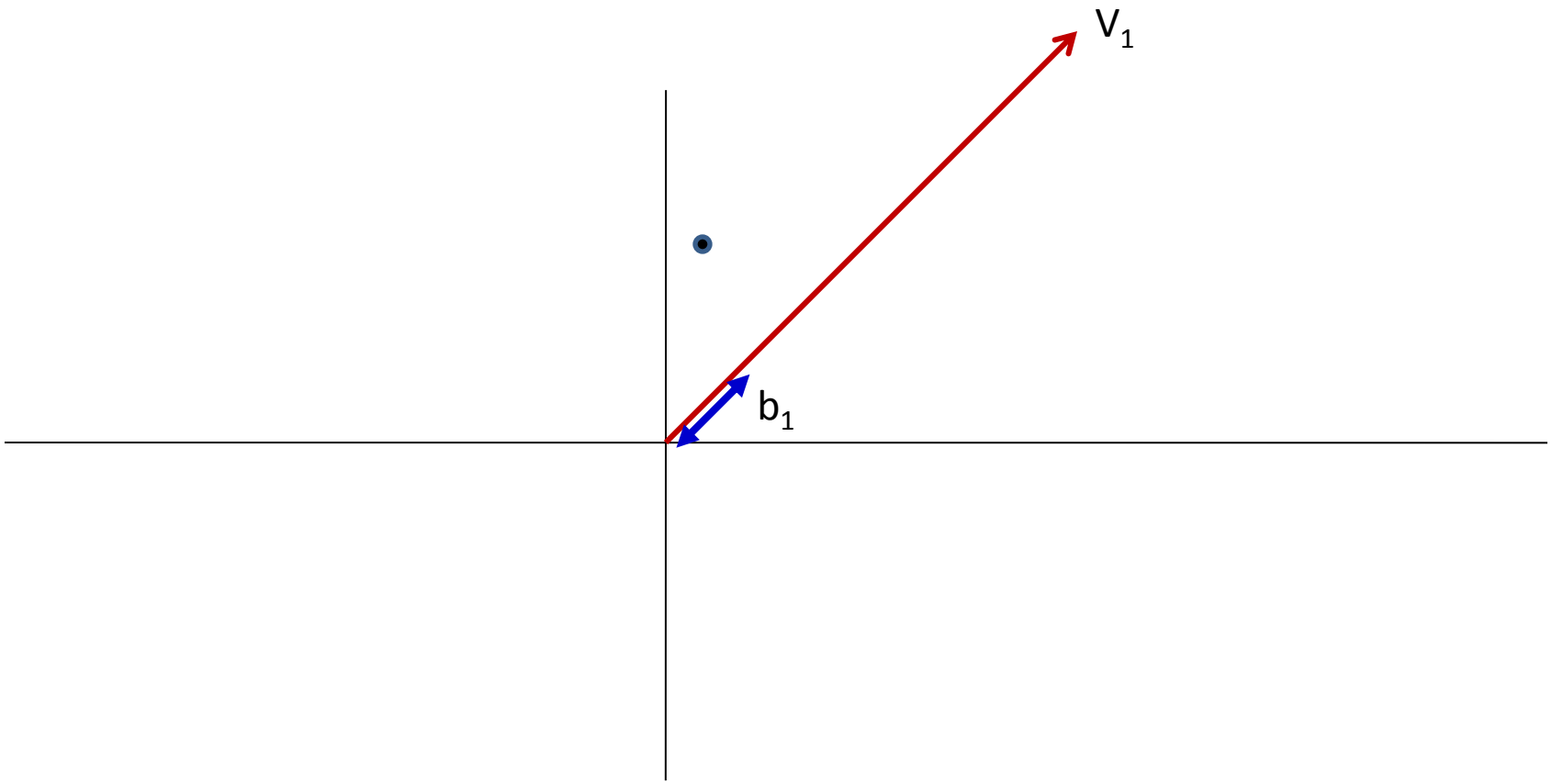
- V_i is a random vector drawn from a normal distribution
- b_i is a random number between 0 and w
- w is the quantization width

Euclidean LSH



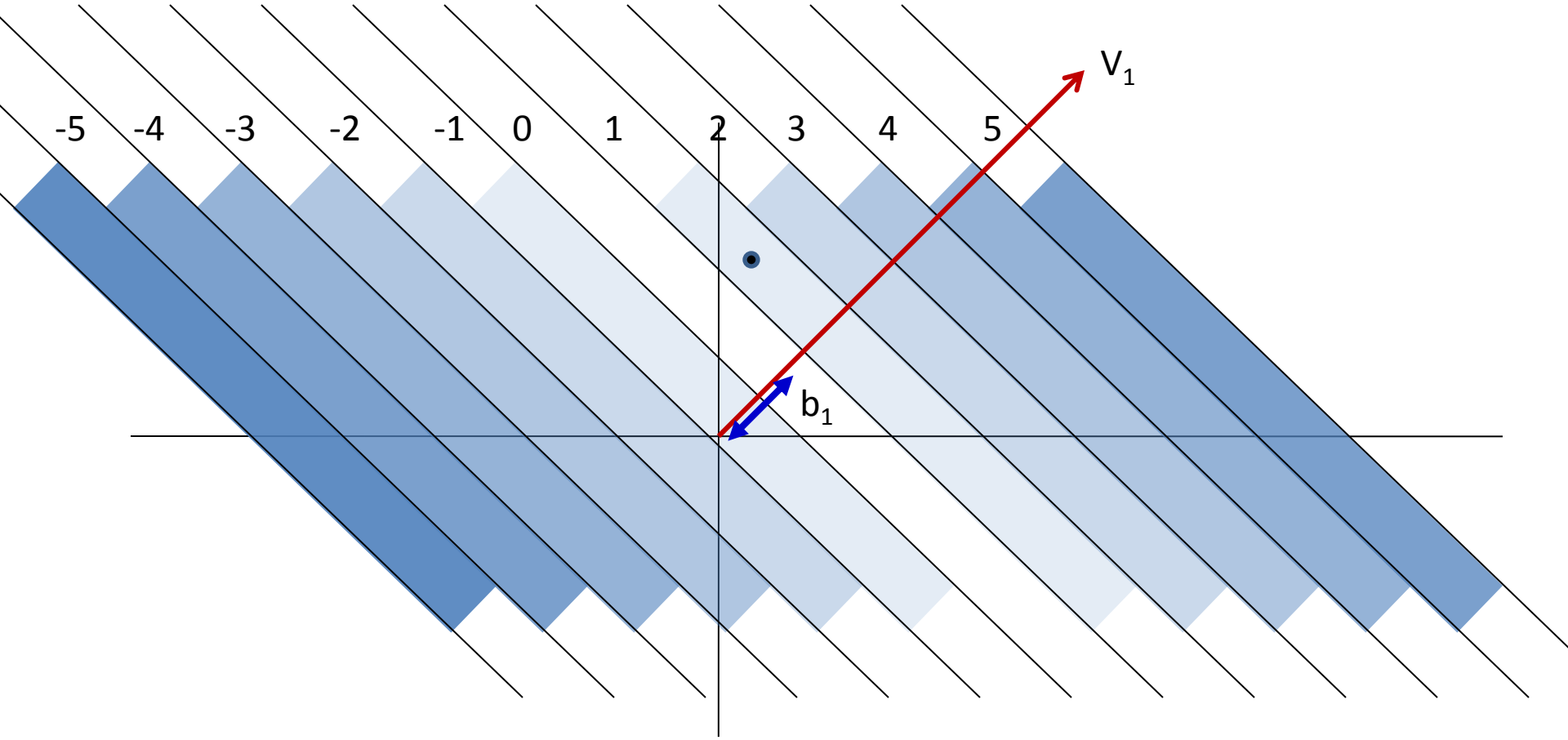
- A 2-D example

Euclidean LSH



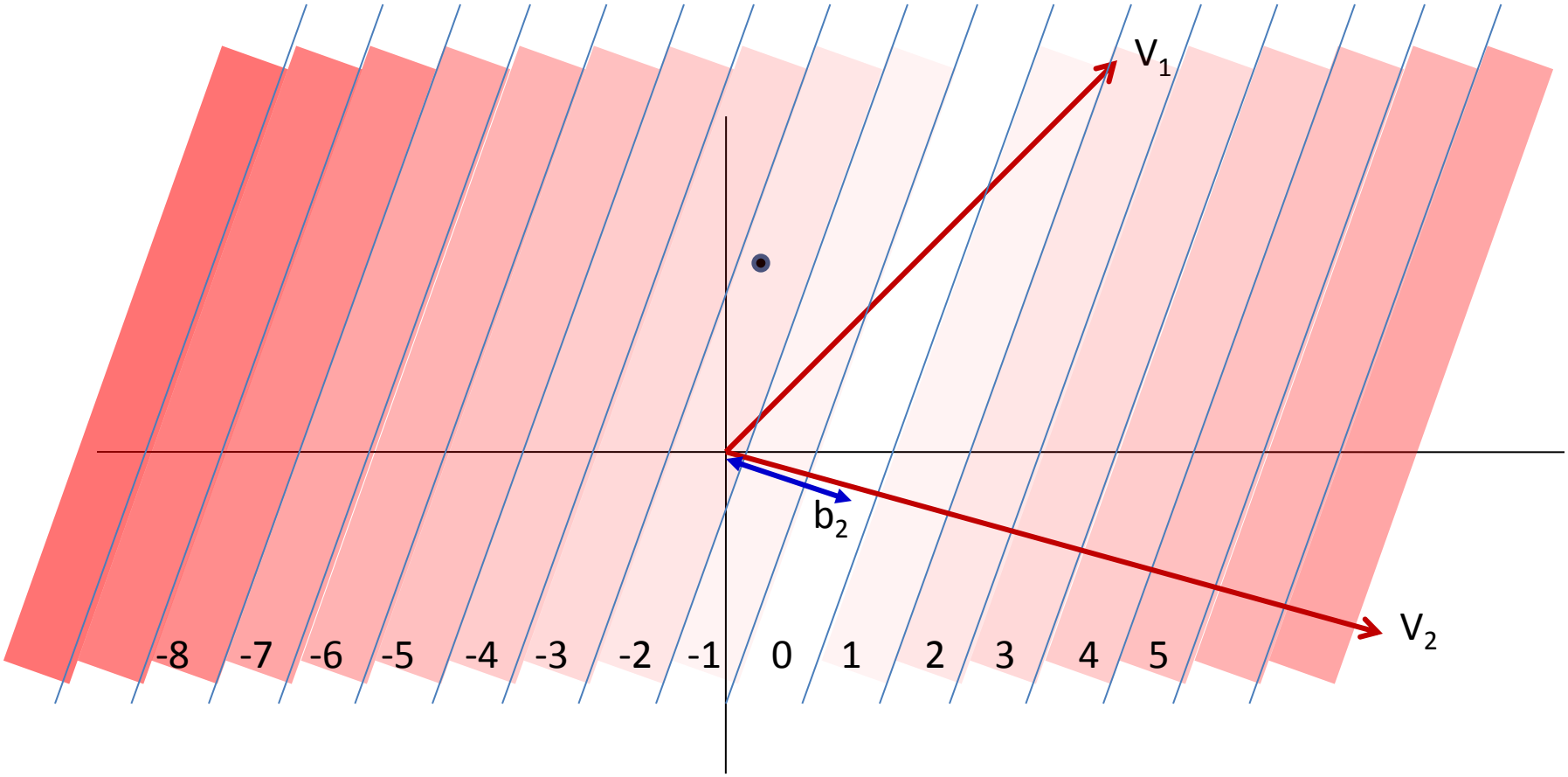
- A 2-D example
- The first component in the hash key: $h_1(X)$

Euclidean LSH



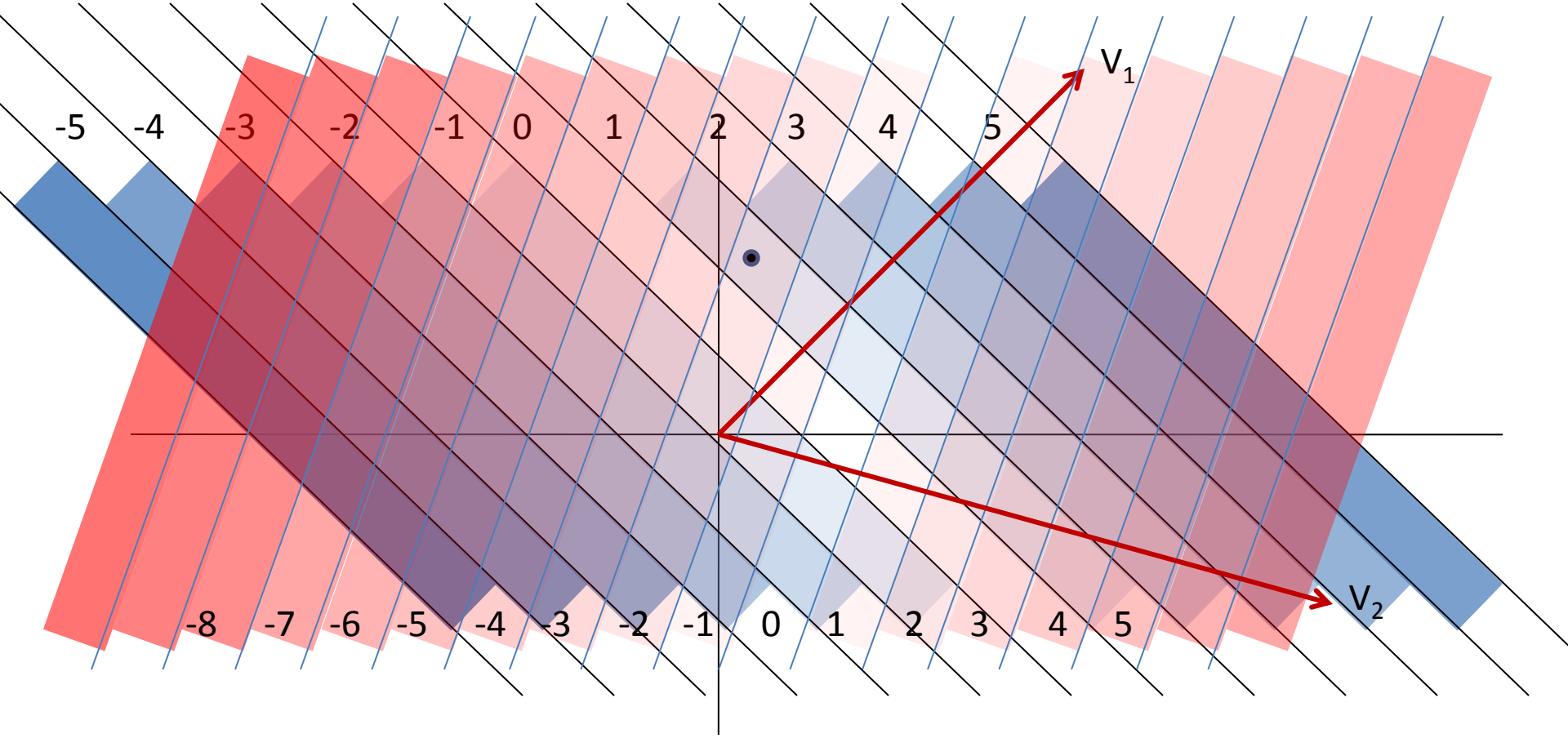
- A 2-D example
- The first component in the hash key : $h_1(X) = 1$

Euclidean LSH



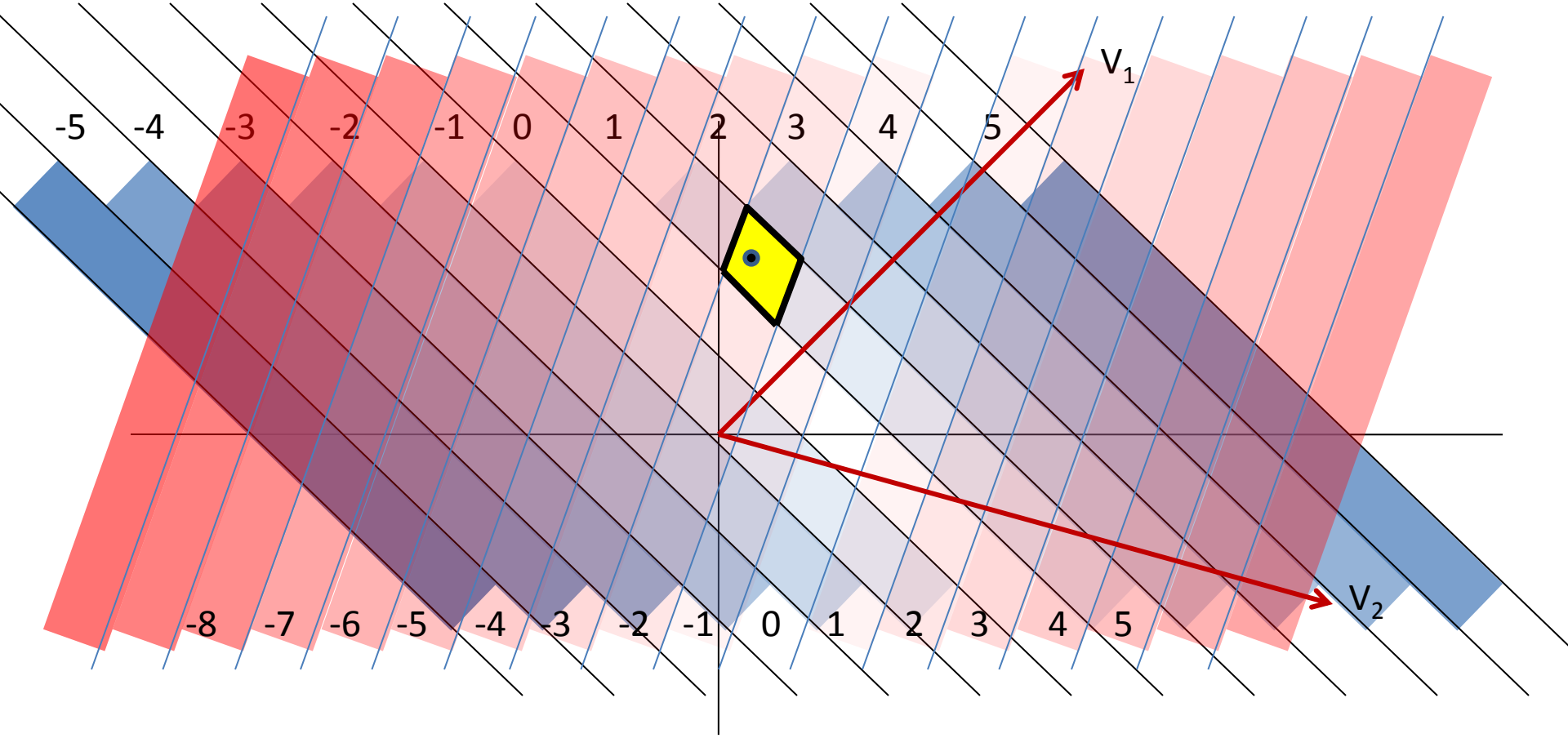
- A 2-D example
- The second component in the hash key : $h_2(X) = -2$

Euclidean LSH



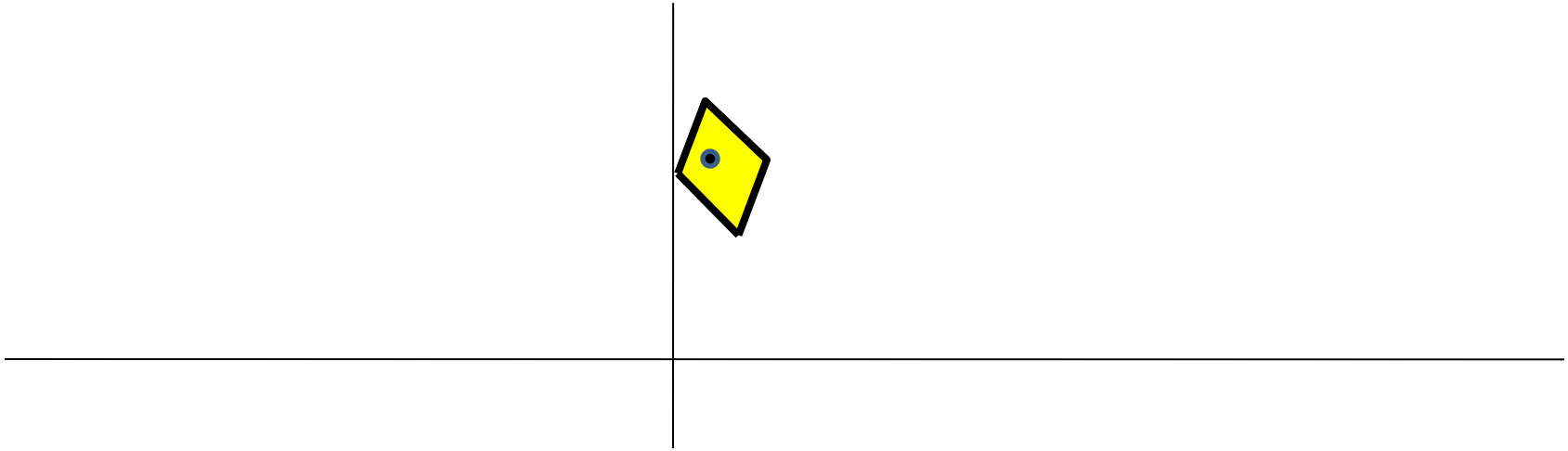
- $H(X) = [h_1(X)\ h_2(X)] = [1\ -2]$

Euclidean LSH



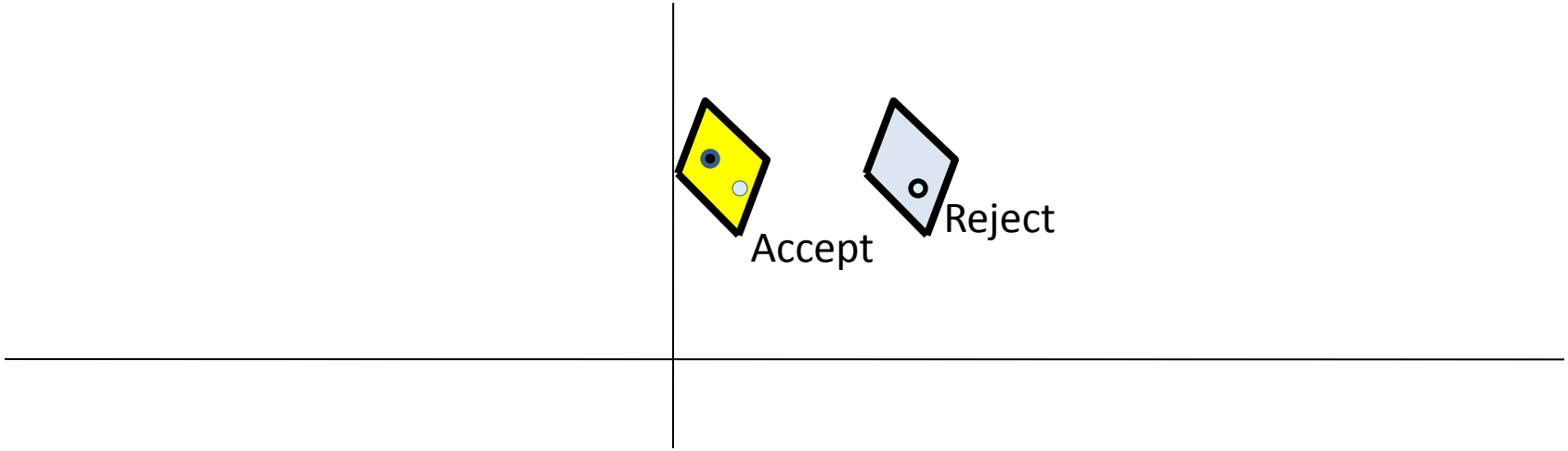
- H(X) = [1 -2]
- All vectors in the highlighted cell will have the same LSH key

Speaker Verification with Euclidean LSH



- Enrollment: Assume one enrollment utterance
  - Convert the supervector for the enrollment utterance to a hash key
    - Find a random cell in which it resides

Speaker Verification with Euclidean LSH



- Test: Find the hash key for the test supervector
  - Find the cell it resides in
- If its identical to the enrollment key, accept
  - Test and enrollment utterances are in the same cell
- Else reject
  - They are in different cells

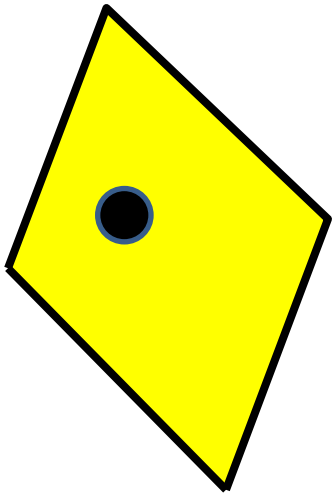
Securing the Key

- LSH keys are informative
- BUT : Two vectors in the same cell have exactly the same key
  - Even if the key is cryptographically hashed!
- **Apply a cryptographic hash to the LSH key**
  - ***User* retains the private key to cryptographic hash**
- **Converted speech to *uninformative* bit strings**
  - **On which comparison can be performed via exact match!**
- In all subsequent discussion, we assume that LSH keys are cryptographically hashed!
  - And are uninformative

The size of the cell

- Increasing the number of components in **H**(X) makes the cell smaller
- **H**(X) = [h_1(X) h_2(X)]

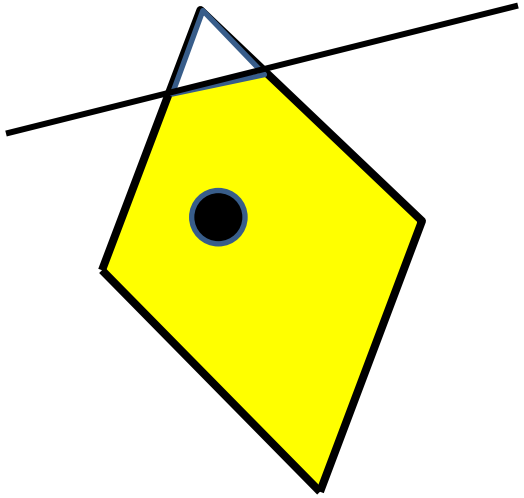
  = [1 -2]



The size of the cell

- Increasing the number of components in **H**(X) makes the cell smaller
- **H**(X) = [h_1(X) h_2(X) **h_3(X)**]

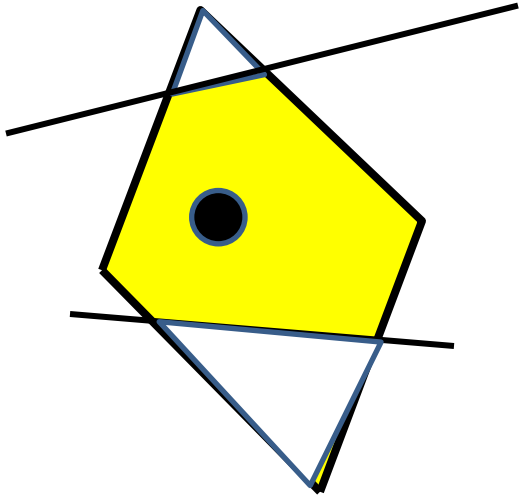
  = [1 -2 **7**]



The size of the cell

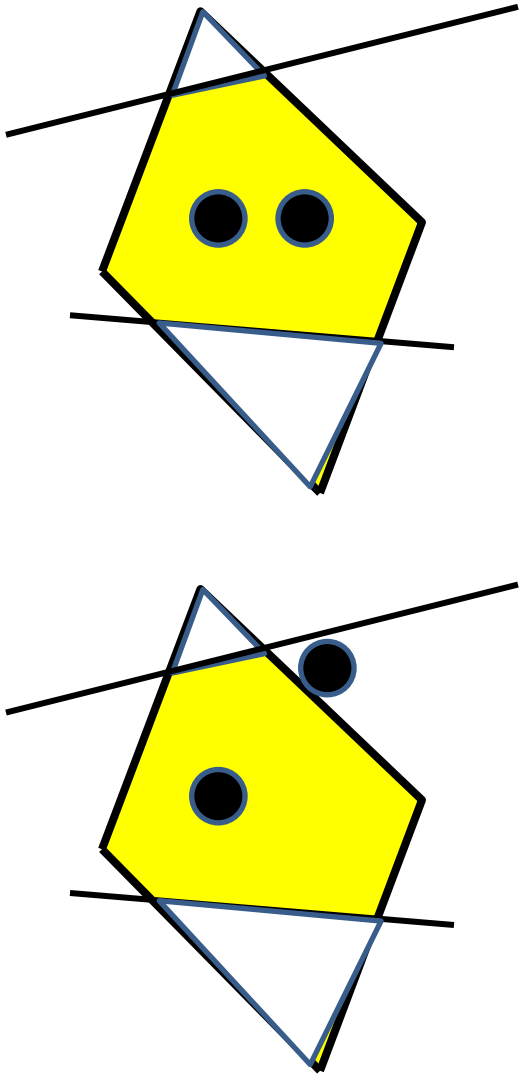
- Increasing the number of components in **H**(X) makes the cell smaller
- **H**(X) = [h_1(X) h_2(X) **h_3(X) h_4(X)**]

  = [1 -2 **7 0**]



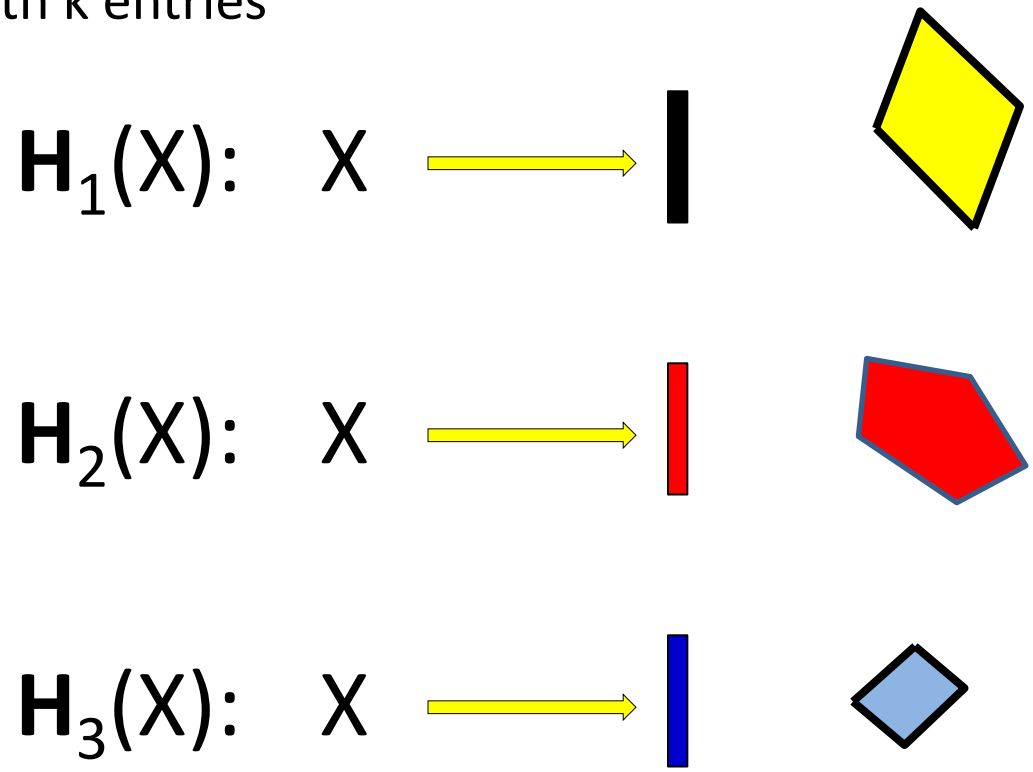
The size of the cell

- Increasing key length → reduced cell size
- Reduced cell size → more likely that two vectors that fall in the same cell (have same LSH key) belong to the same speaker
  - Very Good!
- Also makes it *more* likely to *miss* valid vectors
  - Which may fall outside the cell simply because of the vagaries of its shape



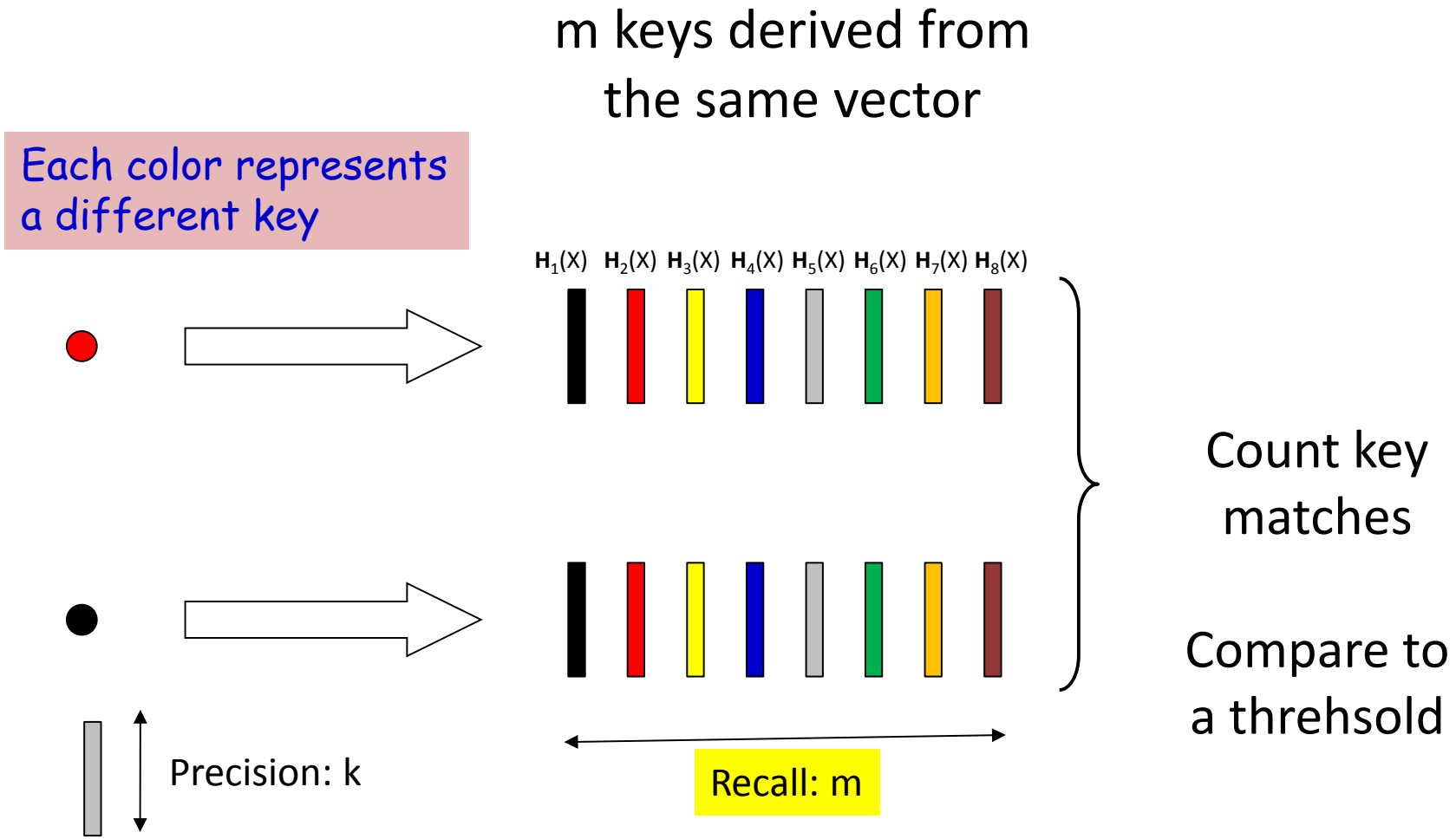
Solution: Use many LSH keys

- Use multiple LSH Hash functions to produce multiple LSH keys
  - Each with k entries



- Each key represents a cell of a different shape and size

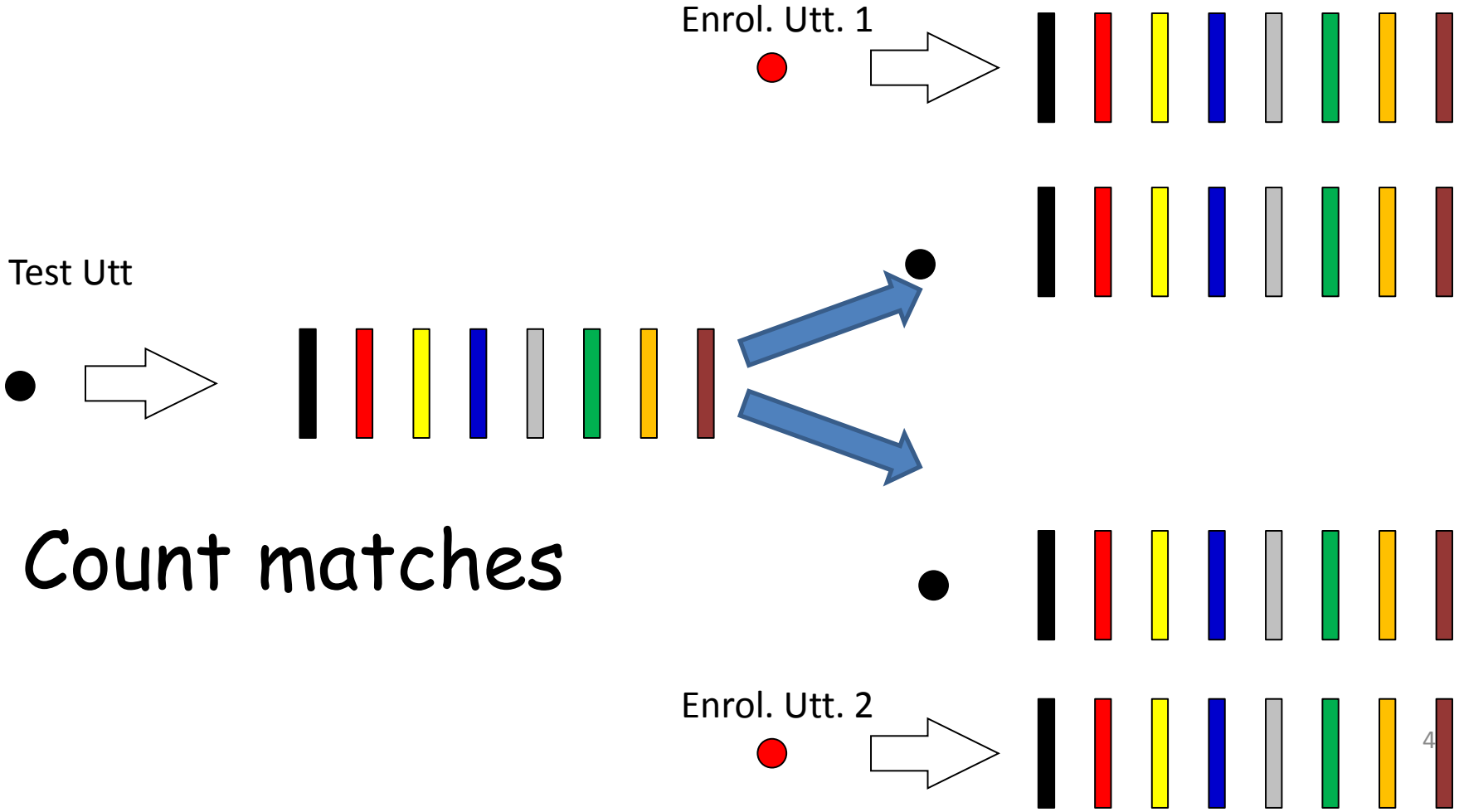
Using multiple LSH functions



Multiple Enrollment Utterances

- A single enrollment utterance is insufficient
- Usually multiple enrollment utterances

Cartoon of Authentication Process



Overall LSH-based Procedure

- **The User obtains a set of (vector) Hash functions**

  $\mathbf{H}_1(.), \mathbf{H}_2(.), \ldots$

  – From the system

- **Enrollment:**
  - User records a set of enrollment utterances $X_1, X_2, ..$
    - And computes supervectors from all of them
  - User computes keys $\mathbf{H}_1(X_1)\ \mathbf{H}_1(X_2)\ ..\ \mathbf{H}_2(X_1)\ \mathbf{H}_2(X_2)\ldots$
    - **Encrypts them with SHA (and retains SHA key)**
    - And sends them to the system
  - **The system stores all encrypted hashes**

Overall LSH-based Procedure

- **Verification**:
  - User records test utterance Y
  - User computes $\mathbf{H}_1(Y)$ $\mathbf{H}_2(Y)$ …
  - User sends keys to system
  - System counts
    - Score = $\Sigma_i \Sigma_J$ $H^j(Y) == H^j(X_i)$
  - If Score > threshold : Accept
  - Else reject

Experiments

LSH & cryptographic hash functions are fast

- For 200 LSH keys per instance overhead was 28 ms!
- Negligible compared to the protocol using homomorphic encryption
  - Independent of the sample length

Experiments

Error on YOHO dataset (EER)

LSH	13.80%	SVM	9.1%

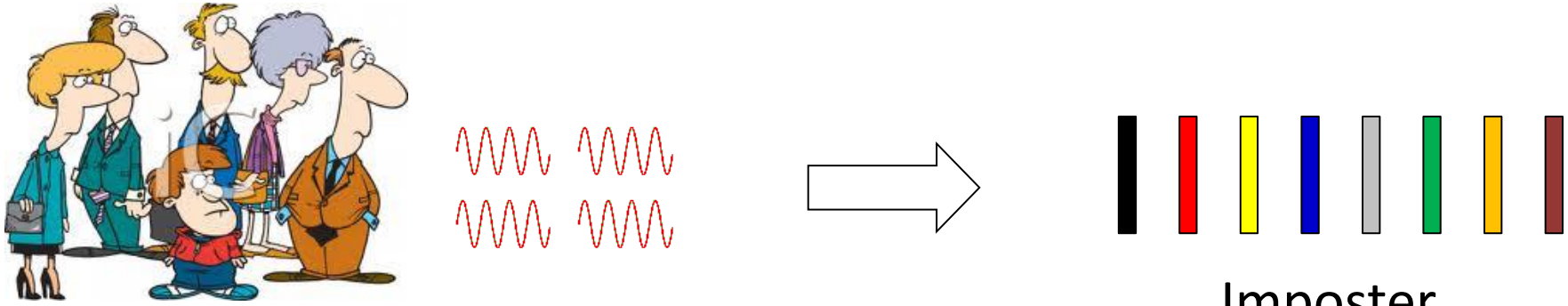
Negative Instance

- Solution so far only considered *positive* instances of data
  - I.e. only consider closeness to enrollment instances from speaker

- Ignore the nature of *imposter* data

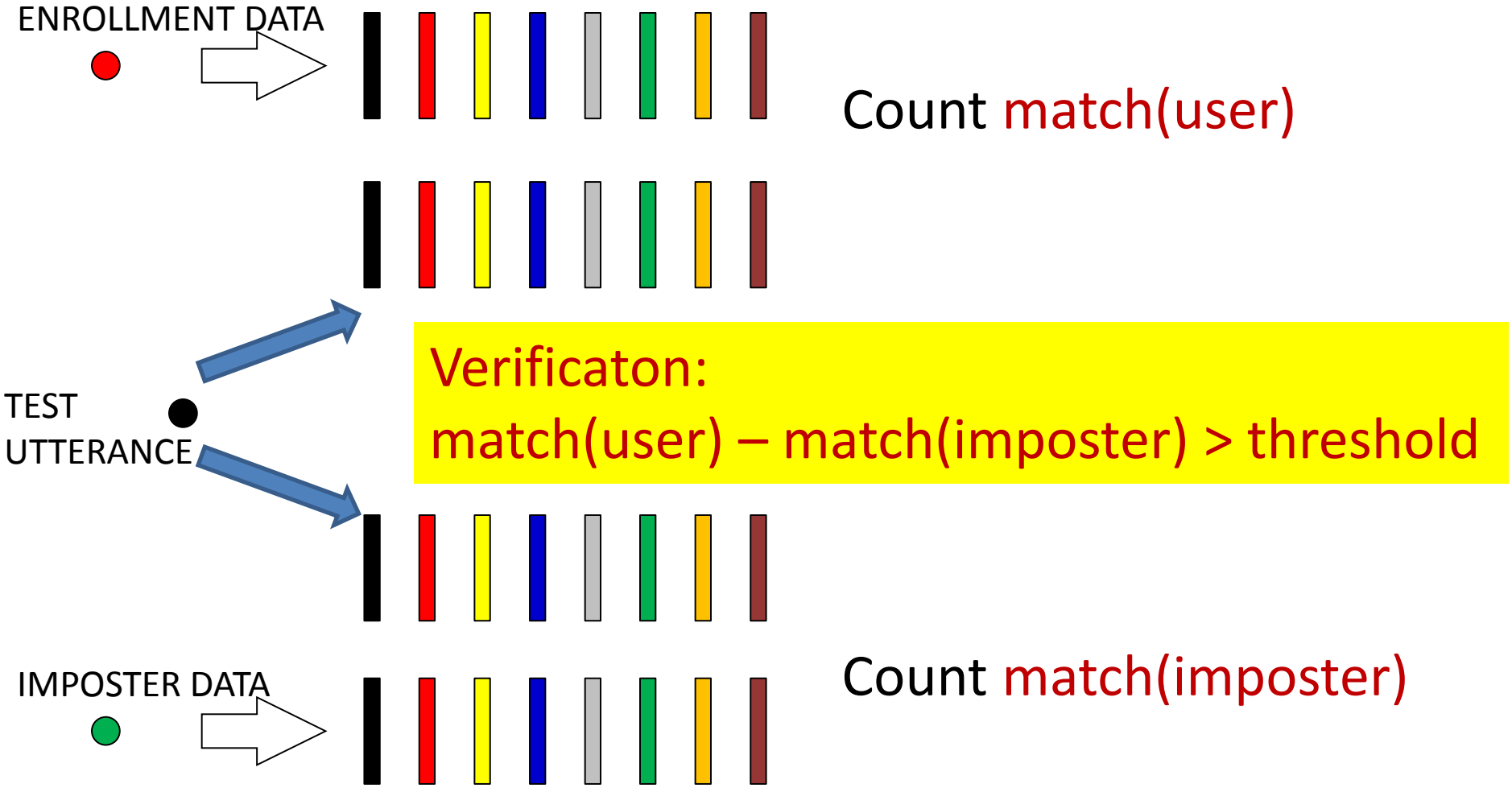
Imposter Data

Improvement : Also use LSH keys from imposter data



- User may records or *generate* imposter data
  - **Record**: download from trusted repository, or from server
  - **Generate**: Algorithms exist for generating negative instances

Considering imposters

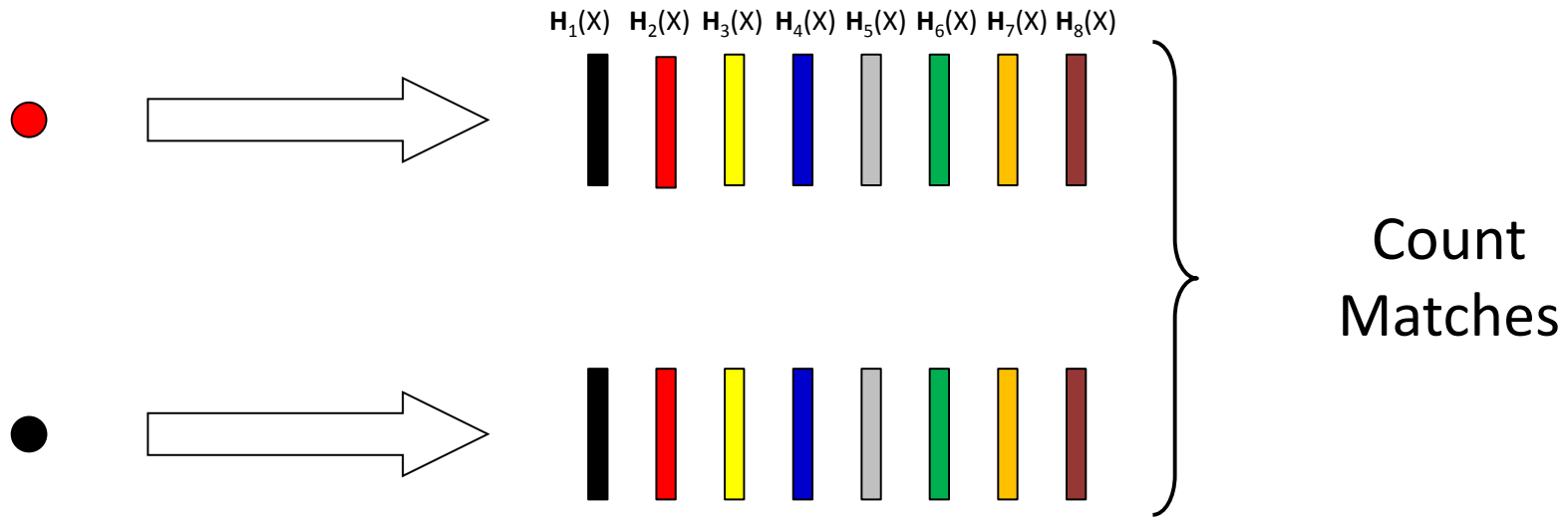


Experiments

Error on YOHO dataset (EER)

only speaker	with imposter	SVM
13.80%	**11.86%**	9.1%

What are we doing



- Estimating the distance between vectors without revealing the vectors!!

A Modified Encryption Scheme

- Conventional Encryption:
  - Each data vector is uniquely encrypted
    - Encrypted vector cannot be decrypted without key
  - Cannot compare data vectors

- A new proposal: A *weakly* private encryption scheme
  - Distance between data vectors can be determined from encrypted vectors
  - But only if they are close enough
  - ***Locality Sensitive Encryption***

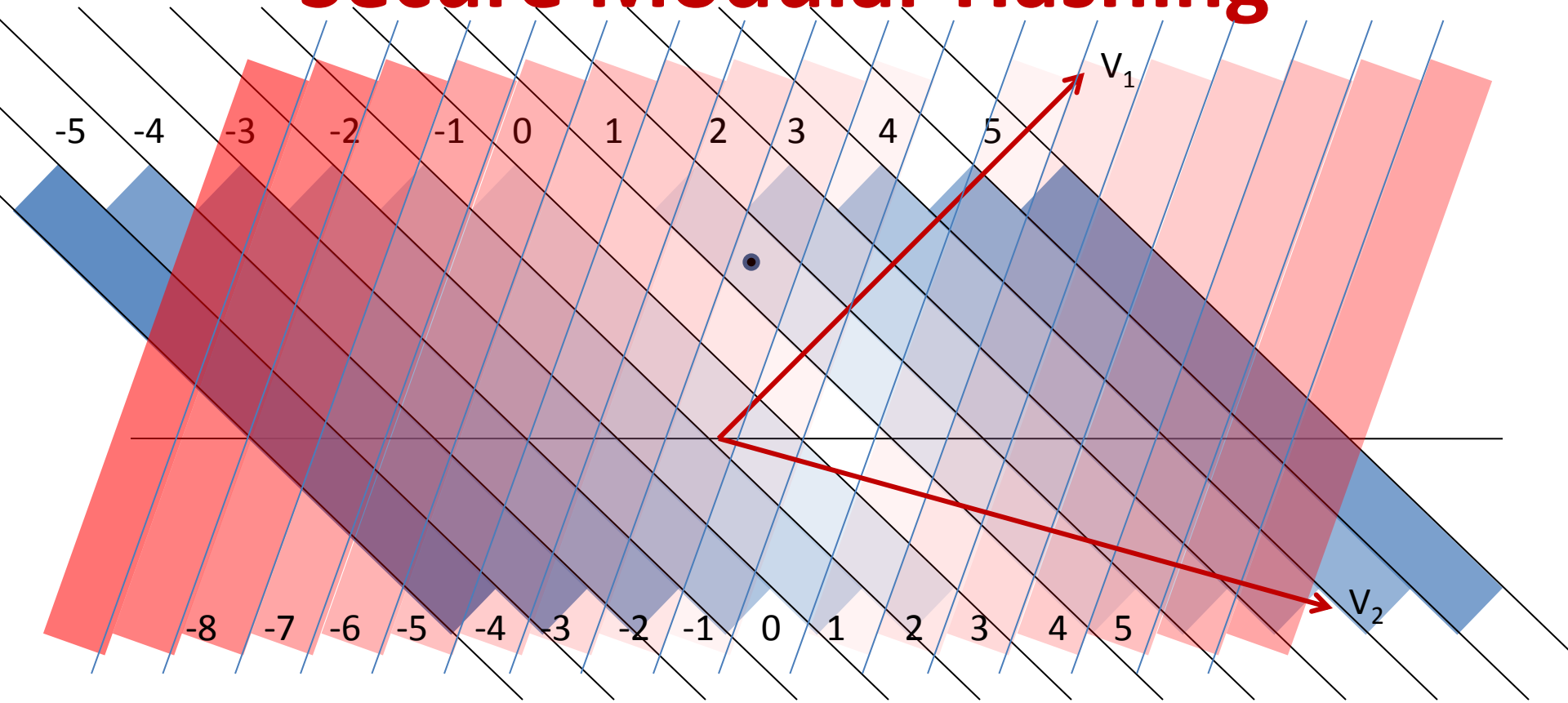
Secure Modular Hashing

$$Q_i(X) = (X^T V_i + B_i) \bmod K$$

$$\mathbf{Q}(X) = [Q_1(X), Q_2(X), \ldots, Q_M(X)]$$

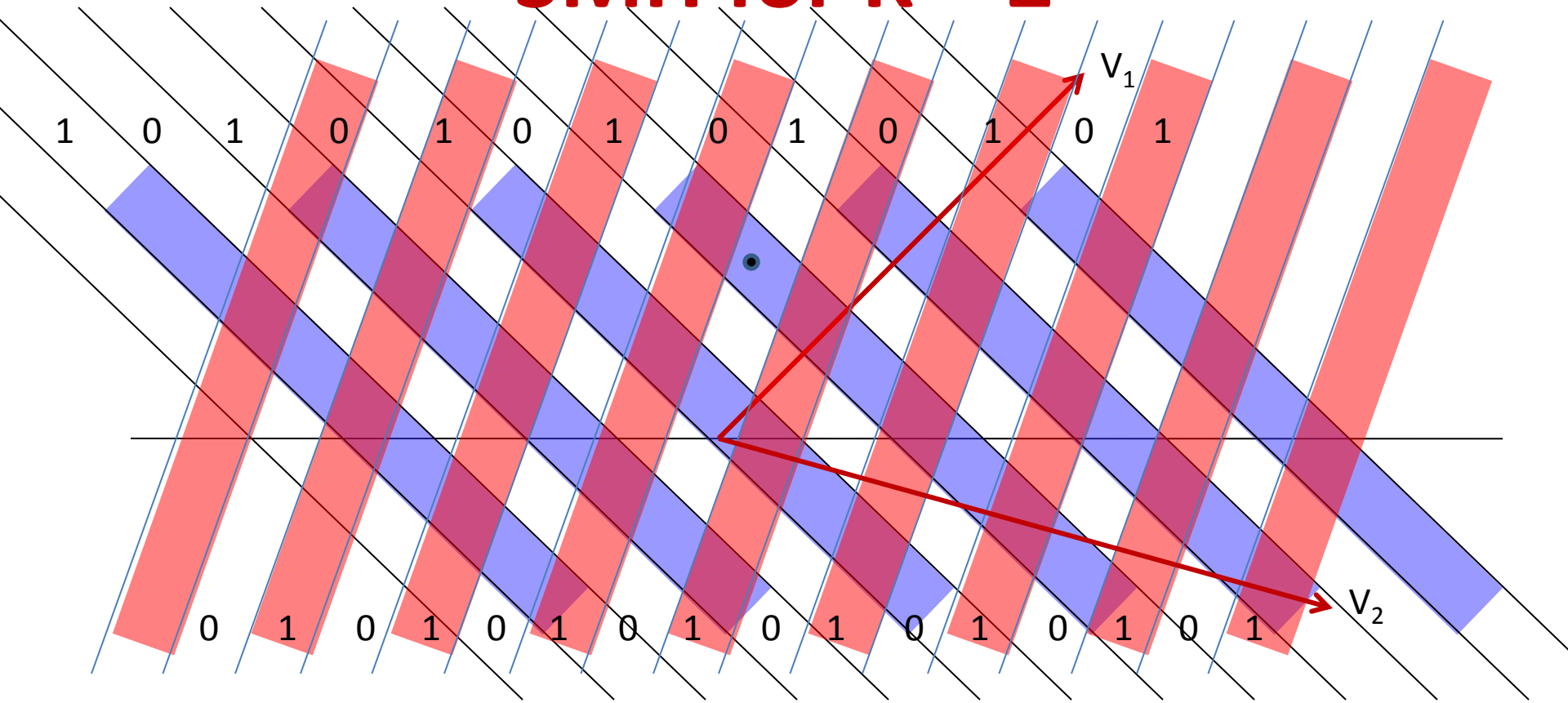
- V_i is a random Gaussian vector where each component is drawn from a Gaussian with mean 0 and variance $1/\delta$
- B_i is a random number from the uniform probability distribution $[0,\mathrm{K}]$

Secure Modular Hashing



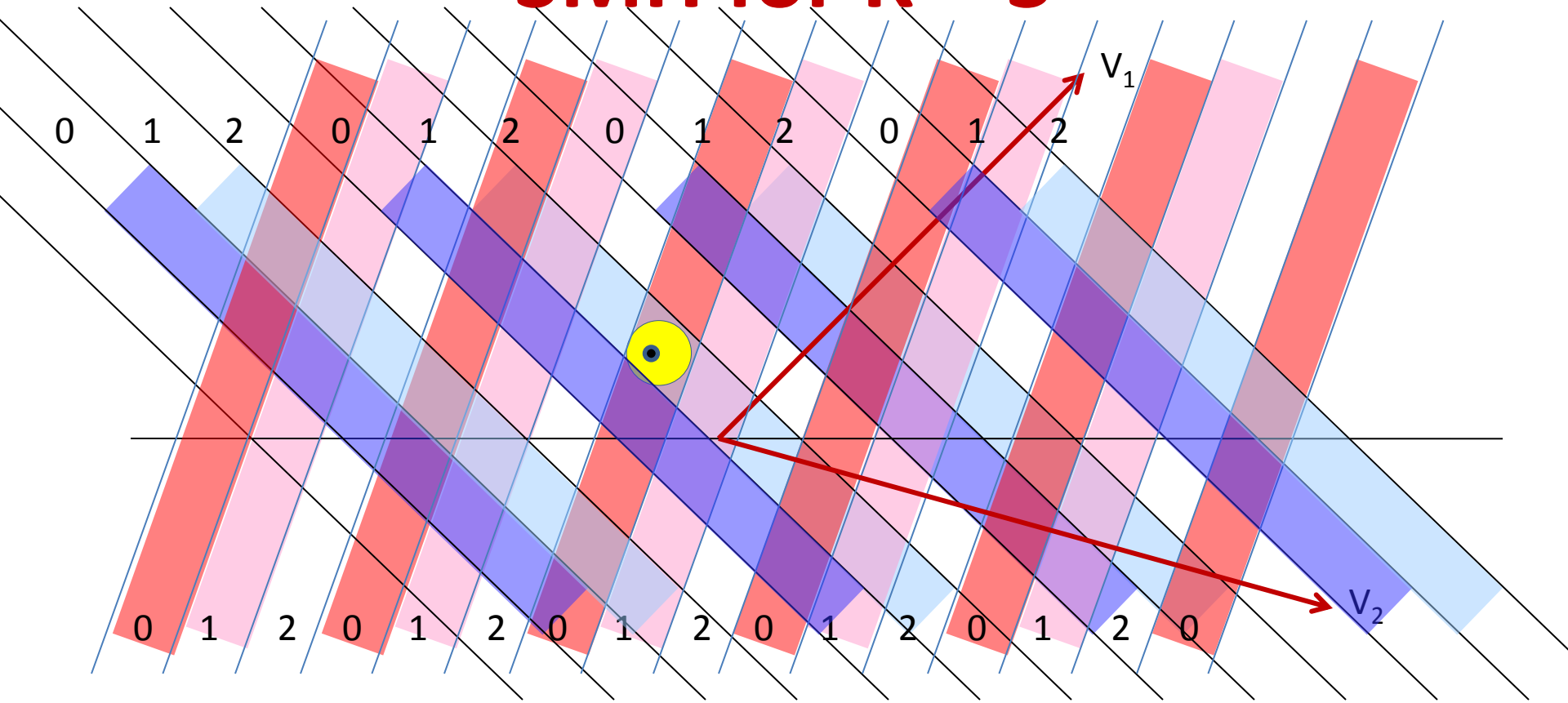
- Conventional LSH

SMH for K = 2



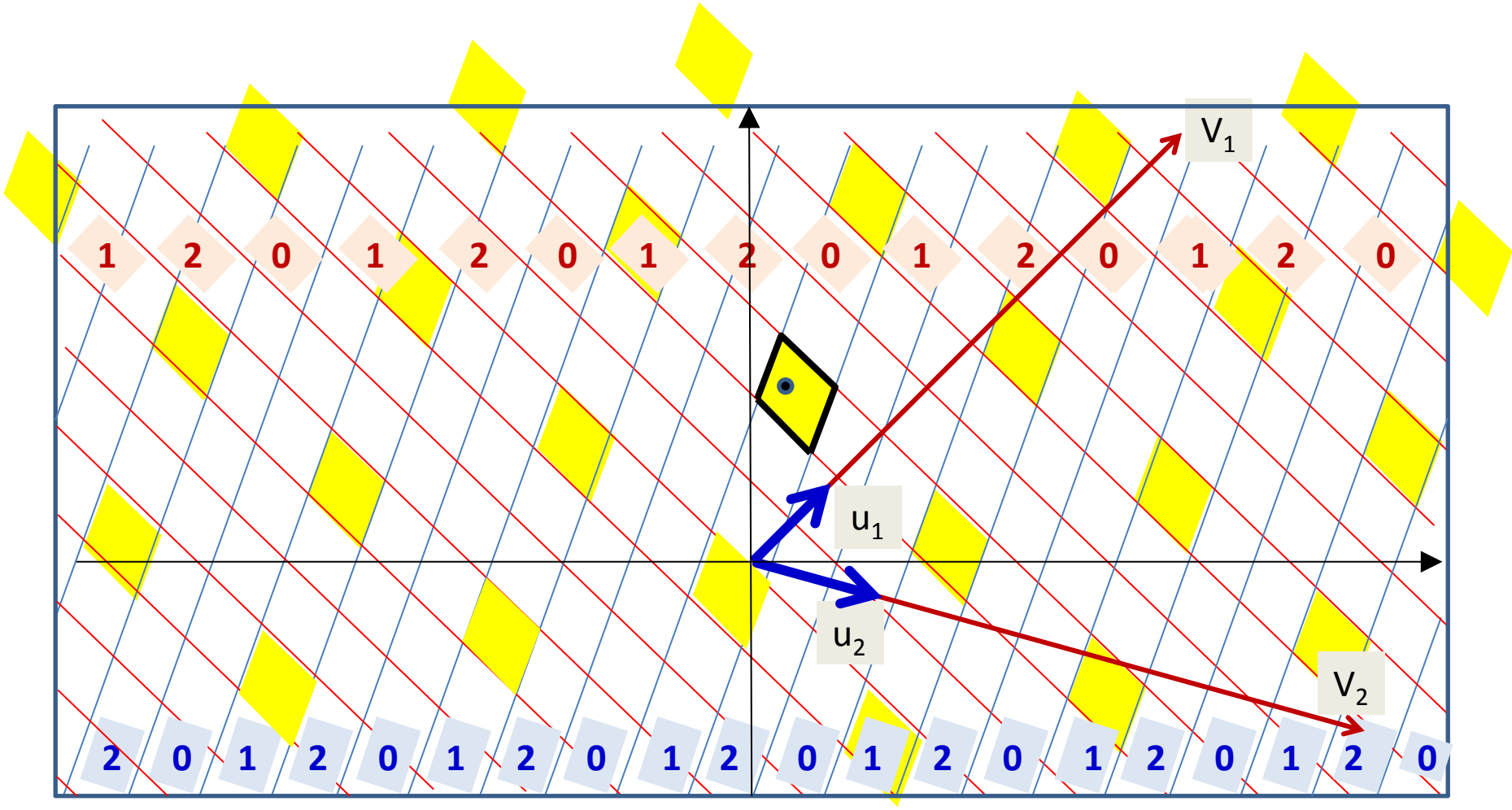
- **Solution: Banded Hashing**
  - Euclidean LSH with binary output

SMH for K = 3



- **Solution: Banded Hashing**
  - Euclidean LSH with ternary output
  - Vector is encrypted to [0 1]

All areas mapped to [0 1]



- The complete set
- Not unique

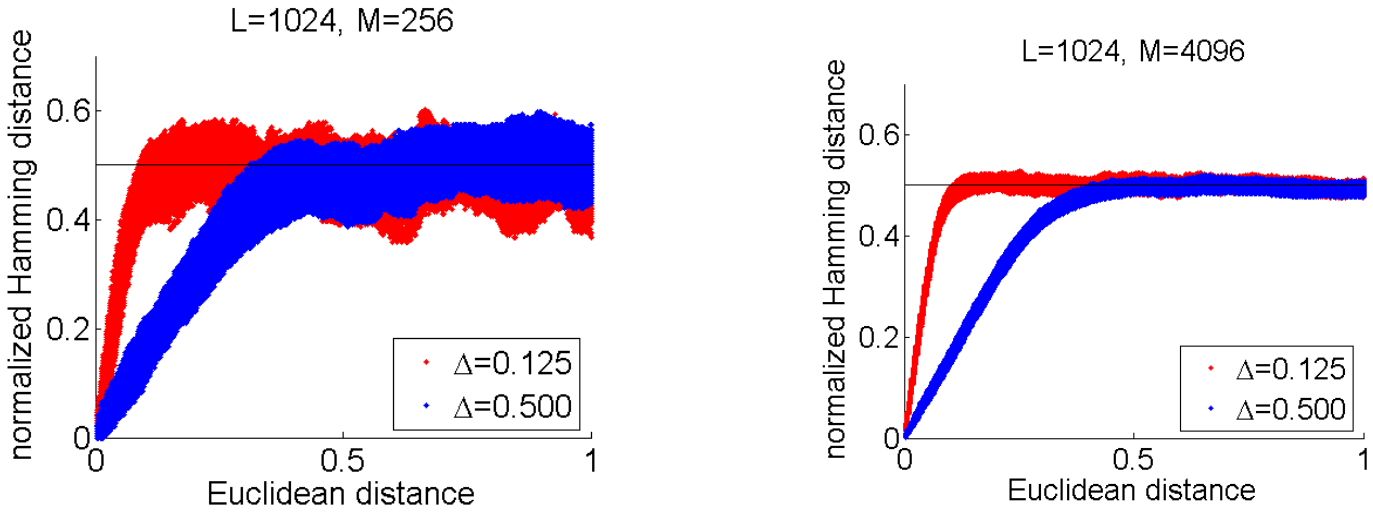
Secure Modular Hashing

$$Q_i(X) = (X^T V_i + B_i) \bmod K$$

- An interesting property
- Hamming(Q(**X**),Q(**Y**)) is proportional to d(X,Y) for d(X,Y) below a threshold
- Above the threshold it is uninformative

SMH K=2

Simulations: L-dimensional vectors, M bit hashes

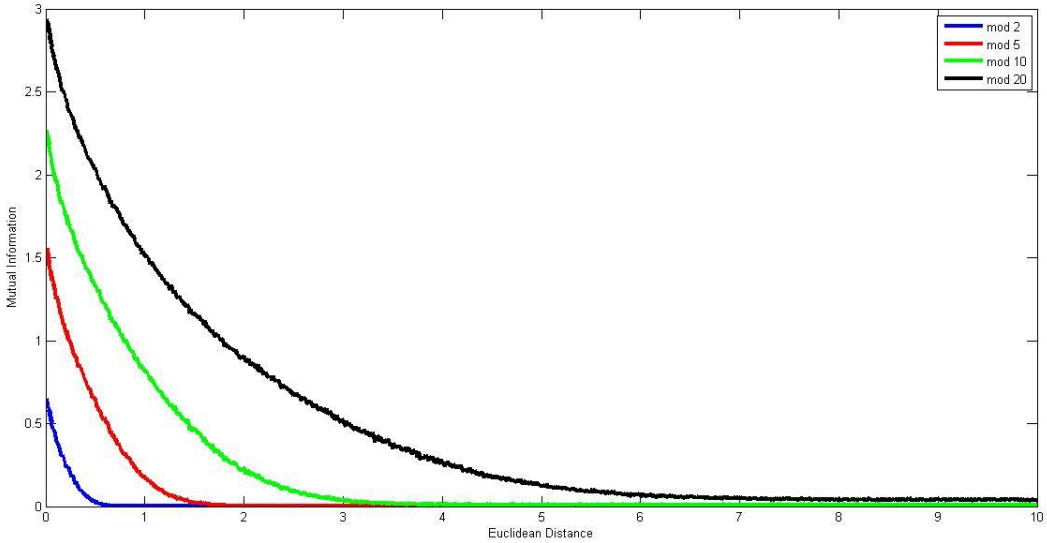


- Plot of Hamming(**Q**(X),**Q**(Y)) vs Euclidean d(X,Y) for different values of Δ, and different numbers of bits in Q(X)

We have a *local* distance measure..

- Gives you true distance, but only within a cell
- Can we use it?
- Revisit the SVM classifier..
- But first: If we're giving away the distance, is it really *secure*?

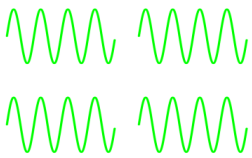
Mutual Information Between Two Vectors



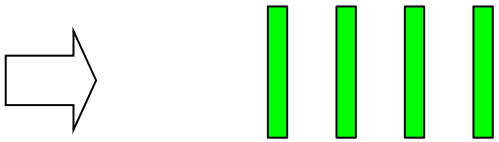
- How much does knowing the encryption of one vector tell you about the encryption of another?
- Beyond the neighbourhood of a vector, the encryption is **information theoretically secure**

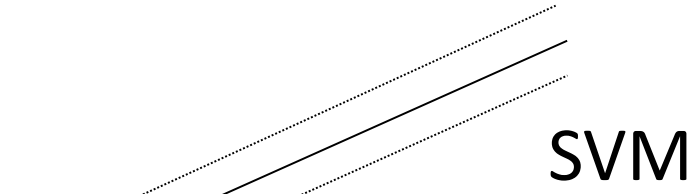
Revisiting SVM based verificaiton

Train SVM classifier across the speaker & imposter supervectors

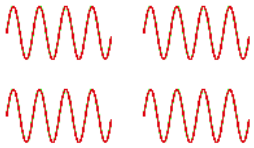


speaker

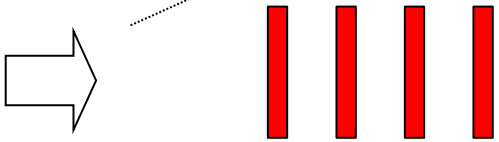




SVM



imposter



SVM with RBF Kernel

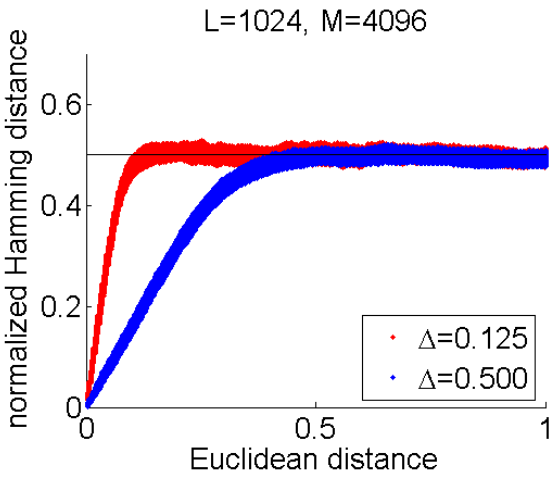
- Kernel has form:

$$K(x_1, x_2) = \exp\left(-\gamma \parallel x_1 - x_2 \parallel^2\right) = \exp\left(-\gamma d(x_1, x_2)^2\right)$$

A Hack

$$K(x_1, x_2) = \exp\left(-\gamma d(x_1, x_2)^2\right)$$

- For small $d(x_1,x_2)$:



$$d(x_1, x_2) \propto Hamm\left(Q(x_1), Q(x_2)\right)$$

- Replace $d(x_1,x_2)$ by the Hamming distance:

$$K(x_1, x_2) = \exp\left(-\beta Hamm\left(Q(x_1), Q(x_2)\right)^2\right)$$

- No longer satisfies Mercer's conditions (not a true Kernel)

SVMs with SMH

- *User* generates SMH function Q()
- User sends SMH of enrollment and imposter data
- System trains pseudo-RBF kernel SVM classifier

$$K(x_1, x_2) = \exp\left(-\beta Hamm\left(Q(x_1), Q(x_2)\right)^2\right)$$

- Verification: User sends SMH of test utterance
- System classifies it with learned pseudo-RBF kernel SVM

How it performs

#Gaussians	4	8	16	32	64	128
F-measure	76.8	89.2	92.8	94.0	94.1	94.4

SVM

Δ	13.5	14.0	14.5	15.0	15.5
bpc=4	85.4	87.9	89.0	90.9	91.6
bpc=8	90.7	92.1	93.0	93.7	93.9
bpc=16	94.0	94.0	94.6	94.6	94.9

SVM on SMH

- SMH based classification *better* than SVM performance
  - More realistically, can claim to be comparable

Import of all this

- Possible to develop authentication system with desired traits
- System never observes user's speech
  - Only obtains SMH vectors, which it cannot interpret
    - Since *User* retains hashing function Q()
- System cannot abuse user's models
  - Only works with SMH vectors generated by *User*
- System can authenticate with high accuracy

Efficiency and Security with SMH

- Efficiency:
- Data vector conversion to SMH vector
  - Insignificant time

- Distance computation with SMH vector instead of data vector
  - Actually faster!

How secure are we really?

- Information theoretic security?
  - Not entirely
  - *Local* distance still given away

- *Practical security?*
  - Absolutely
  - System never observes user's speech
  - System does not possess user's models
  - System can authenticate with high accuracy

Does it generalize?

- Proposed SMH: Encryption scheme that permits computation of *local* distances without revealing data
- Can be used to compare *any* two data vectors
- Other uses:
  - Privacy-preserving word spotting
  - Privacy-preserving important passage retrieval from documents
- Cost we pay?
  - Not reversible
  - Locally leaky
    - Given a sufficient number of samples, a hacker may map the data

Conclusion

Privacy-Preserving Speech Processing is feasible & useful

- Discussed two approaches:
  - CRYPTOGRAPHIC --- based on secure multi-party computation
  - STRING MATCHING --- based on hashing
- Cryptographic methods show theoretical feasibility
  - But practically, still infeasible
- String matching methods are practically feasible
  - For small overhead

What in the Future

- Its all in the cloud
- Sensible people will always encrypt data
- We need methods to mine/index/match/recognize encrypted data
- Have shown a couple of workable solutions, but they are not the final solution
  - Too slow
  - Too leaky

- Work on encryption schemes continues
- Alternate strategies may also be required

Thanks!

Questions?