Information Security – Hands-On

Prof. Dr. Marta Gomez-Barrero

Hochschule Ansbach

VISUM Summer School, 09/07/2020

Bloom Filters

Why Bloom filters?

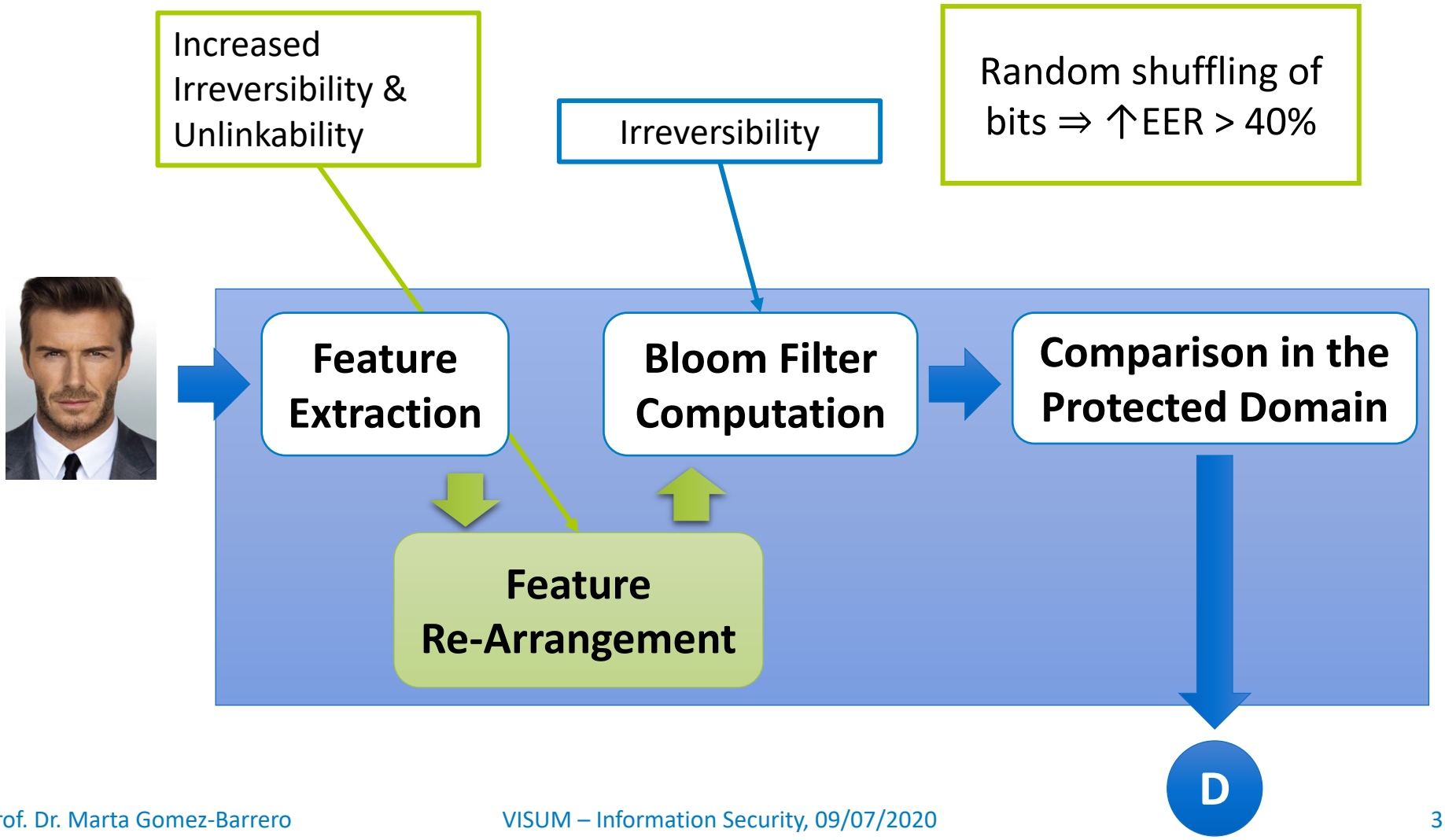
[Bloom, *Comm. of the ACM* 1970]
[Broder and Mitzenmacher, *Internet Mathematics* 2004]

- Biometric Template Protection based on Bloom filters:
  - **General**: successfully applied to iris, face, fingerprint, fingervein
  - **Multimodal**: feature level fusion is possible
  - **Irreversibility** achieved
  - **Accuracy**, depending on the configuration, preserved
  - **Template size**: similar or compressed
  - **Verification speed** similar
  - **Unlinkability** can be added to the original approach

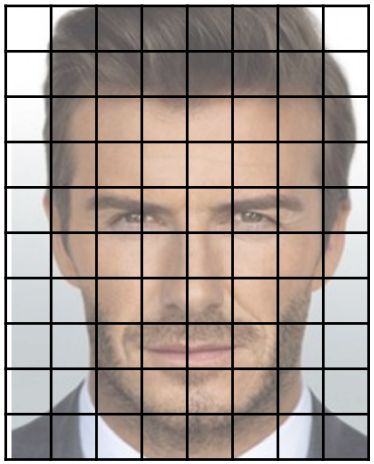
[Gomez-Barrero *et al.*, *Information Sciences* 2016]
[Gomez-Barrero *et al.*, *Information Fusion* 2018]

Bloom filters

General architecture



Bloom filters



Re-Arranged Block

nBits

0	0	1	0	0	1	1	0	0	1
1	1	1	0	1	0	0	1	1	0
1	0	0	0	1	1	0	0	1	0
0	0	0	1	0	0	1	0	0	1

nWords

6 4 6 9

0	1	0	0	1	0	1	0	0	1	1	0	1	0	0	0

2^{nBits}

Bloom Filter

1				0					1
1				1					0
...	...	...	...	...	...	...	...	...	...
0				0					1

Protected Template

1 BF per block, of 2^{nBits}

Pre-processing data

- First, we need to pre-process the data in order to be able to use the provided scripts

- We have one file, that we need to read into Python:

```
with open('irisCodes.npy', 'rb') as f:
    irisCodes = np.load(f, allow_pickle=True)
```

- And we need to store each iriscode in a separate file with a loop:

```
filename = str(user) + '_' + str(sample) + '.txt'
np.savetxt('database/' + filename,
    irisCodes[user,sample], fmt='%d',delimiter='')
```

Hands-On


```
usage: BF_extraction_iriscodes.py [-h] [--DB_BFtemplates [DB_BFTEMPLATES]]
                                  DBdir

Extract unprotected LGBPHS and protected Bloom filter templates from the FERET
DB.

positional arguments:
  DBdir                 directory where the compressed face DB is stored

optional arguments:
  -h, --help            show this help message and exit
  --DB_BFtemplates [DB_BFTEMPLATES]
                        directory where the unprotected face templates will be
                        stored
  --nXORKeys [NXORKEYS]
                        number of keys for the XOR operation for the feature
                        level fusion. for a unimodal system it should be the
                        default: 0
```

Computing BF Templates

- Python **BF_extraction_iriscodes.py** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**database** --DB_BFtemplates /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**database_BF**

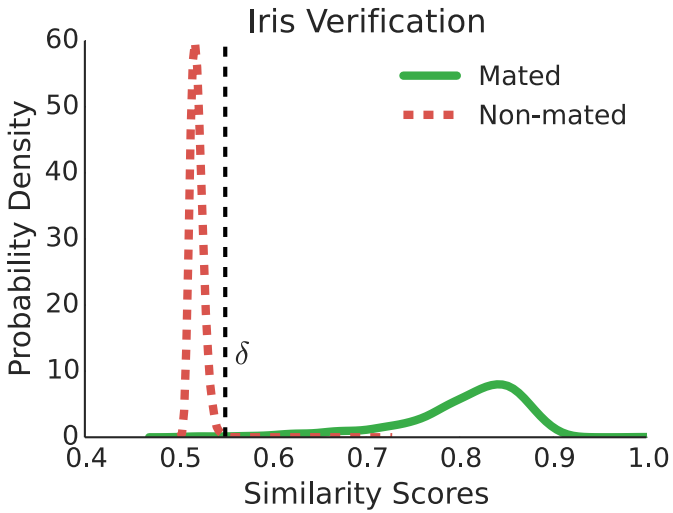
```
usage: computeScores.py [-h] [--scoresDir [SCORESDIR]]
                        [--matedScoresFile [MATEDSCORESFILE]]
                        [--nonMatedScoresFile [NONMATEDSCORESFILE]]
                        DB_BFtemplates matedComparisonsFile
                        nonMatedComparisonsFile

Compute protected Bloom filter scores from a given DB and protocol.

positional arguments:
  DB_BFtemplates        directory where the protected BF templates are stored
  matedComparisonsFile  file comprising the mated comparisons to be carried
                        out
  nonMatedComparisonsFile
                        file comprising the non-mated comparisons to be
                        carried out

optional arguments:
  -h, --help            show this help message and exit
  --scoresDir [SCORESDIR]
                        directory where unprotected and protected scores will
                        be stored
  --matedScoresFile [MATEDSCORESFILE]
                        file comprising the mated scores computed
  --nonMatedScoresFile [NONMATEDSCORESFILE]
                        file comprising the non-mated scores computed
```

Computing BF Scores

- We need to create lists of the comparisons we will make
- 224 subjects may take too long, so now we reduce to only 100

- From each subject, we have 5 samples
- Mated comparisons: all possible: 0vs1, …, 0vs4, 1vs2, …, 3vs4
- Non-mated comparisons: all possible is too much, so we reduce to

  userX_0 vs userY_1, with x != Y



Creating the lists

```
matedComparisonsFile = '/…/matedComparisonsBF.txt'
nonMatedComparisonsFile = '/…/nonMatedComparisonsBF.txt'

matedF = open(matedComparisonsFile, 'w')
nonMatedF = open(nonMatedComparisonsFile, 'w')

N_USERS = 100

for user in range(N_USERS):
    for sample in range(5):
        for sample2 in range(sample + 1,5):
            line = str(user) + '_' + str(sample) +
                   '_BFTemplate.txt ' +
                   str(user) + '_' + str(sample2) +
                   '_BFTemplate.txt\n'
            matedF.write(line)
```

Creating the lists

```
for user in range(N_USERS):
    for user2 in range(user+1,N_USERS):
        line = str(user) + '_0_BFTemplate.txt ‘ +
                    str(user2) + '_1_BFTemplate.txt\n'
        nonMatedF.write(line)
```

Computing BF Scores

- Python **computeScores.py** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**database_BF** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**matedComparisonsBF.txt** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**nonMatedComparisonsBF.txt** --scoresDir /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**BFScores**

Hamming distance for Bloom Filters

```
def hamming_distance(X, Y):

    dist = 0

    N_BLOCKS = X.shape[0]
    for i in range(N_BLOCKS):
        A = X[i, :]
        B = Y[i, :]

        suma = sum(A) + sum(B)
        if suma > 0:
            dist += float(sum(A ^ B)) / float(suma)

    return dist / float(N_BLOCKS)
```

Hamming Distance for Iriscodes

- When we acquire an image of the iris, we may rotate our head
- Therefore, we compensate for that with circular shifts of the iriscodes
- In Python, we can shift a row of the iriscode like this:

```
B = numpy.roll(Y[i, :],s)
```

- Now, you turn to implement the HD!

Hamming Distance for Iriscodes

```
def hamming_distance(X, Y):

    N_SHIFTS = 4
    N_ROWS = X.shape[0]

    N_ELEM = X.shape[0] * X.shape[1]

    dist = numpy.zeros(2*N_SHIFTS + 1)
    ctr = 0
    for s in range(-N_SHIFTS, N_SHIFTS+1):
        for i in range(N_ROWS):
            A = X[i, :]
            B = numpy.roll(Y[i, :],s)
            dist[ctr] += float(sum(A ^ B))
        ctr += 1

    return min(dist) / float(N_ELEM)
```

Hamming Distance for Iriscodes

```
Python computeScores_Iriscodes.py
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/database
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/matedComparisons.txt
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/nonMatedComparisons.txt
--scoresDir /Users/marta/Documents/hs-
ansbach/Python/VISUM2020/IrisCodeScores
```

Plotting a DET

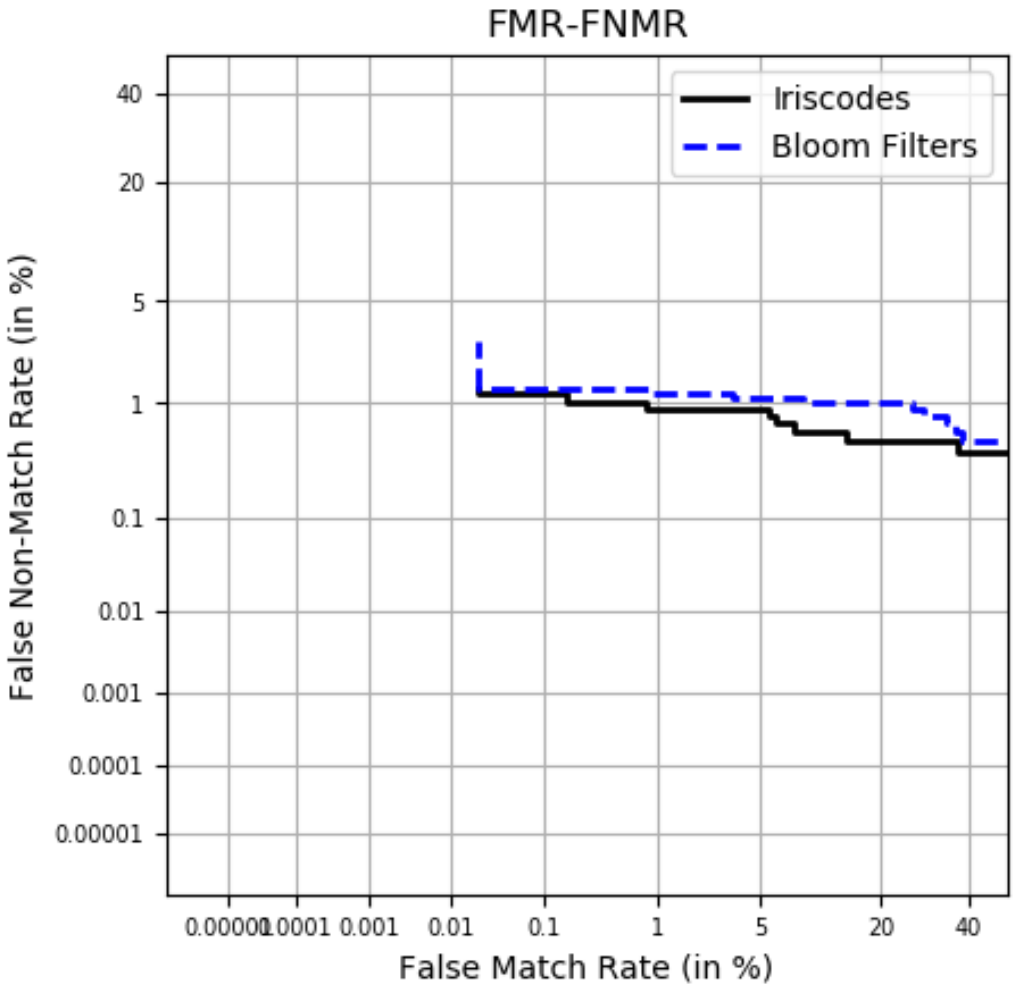
```
import numpy


## load your scores files!

from DET import DET
det = DET(biometric_evaluation_type='algorithm',
plot_title='FMR-FNMR')

det.create_figure()


det.plot(tar=1-matedScores, non=1-nonMatedScores,
label='Iriscodes')

det.plot(tar=1-matedScoresBF, non=1-nonMatedScoresBF,
label='Bloom Filters')
det.legend_on()
det.save('example_algorithm', 'png')
```

Plotting a DET

- If you get an error, you probably need to run in the terminal the following line:

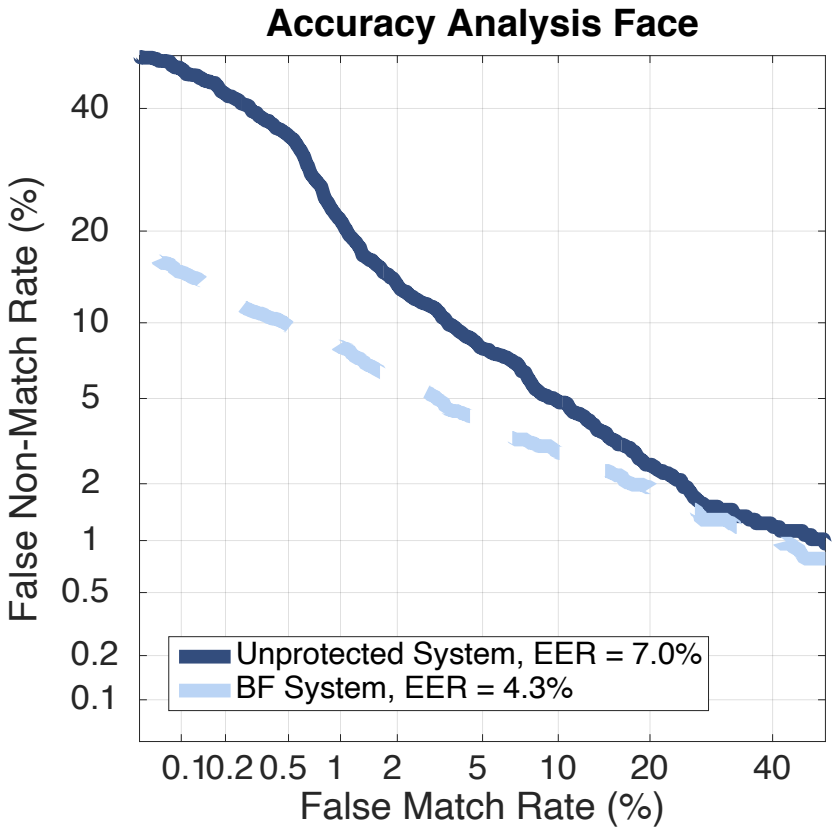
```
pip install tikzplotlib
```

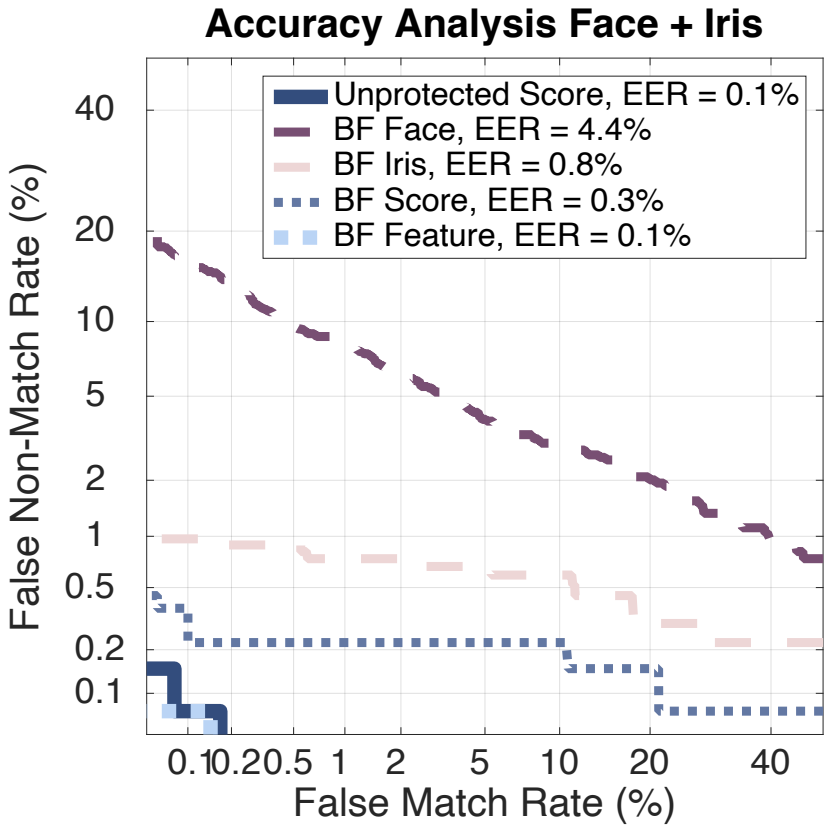
Accuracy Analysis



Bloom filters

Accuracy Analysis



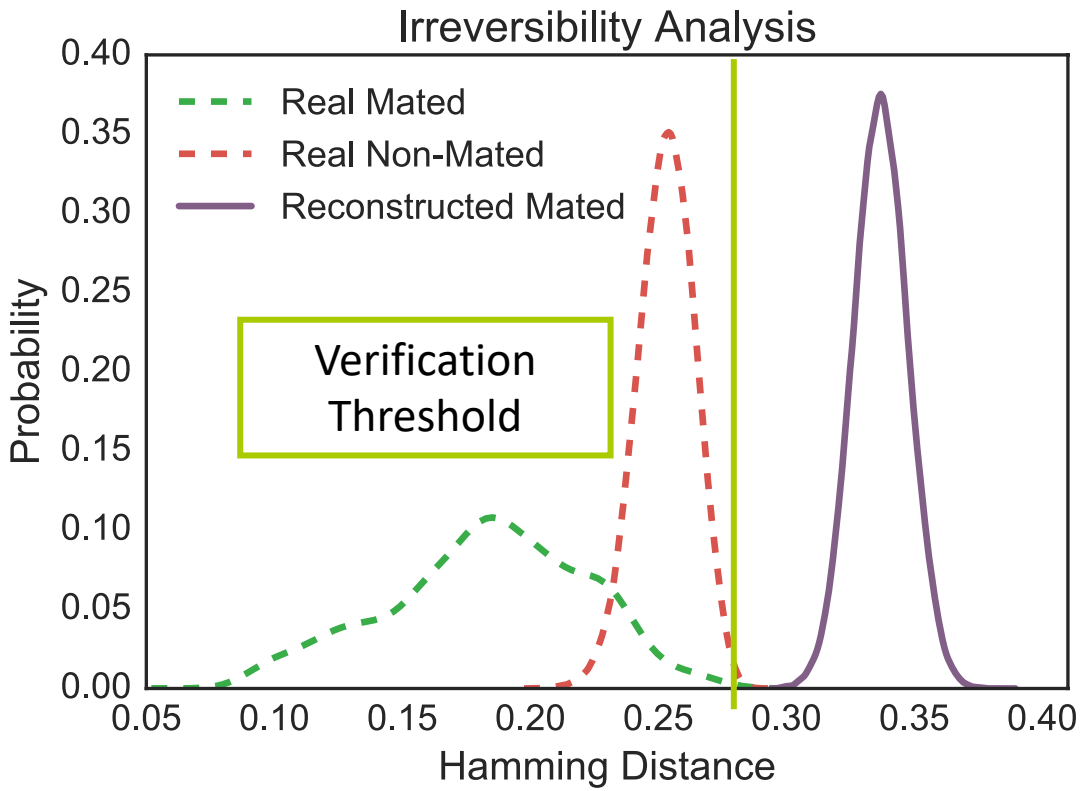


Accuracy is preserved at all operating points

For the fusion, best accuracy for protected feature level

Irreversibility analysis

- Are the reconstructed unprotected templates similar to the original ones?



[Bringer *et al.*, *ICB* 2015]

Irreversible: HD bigger than impostor comparisons

Irreversibility

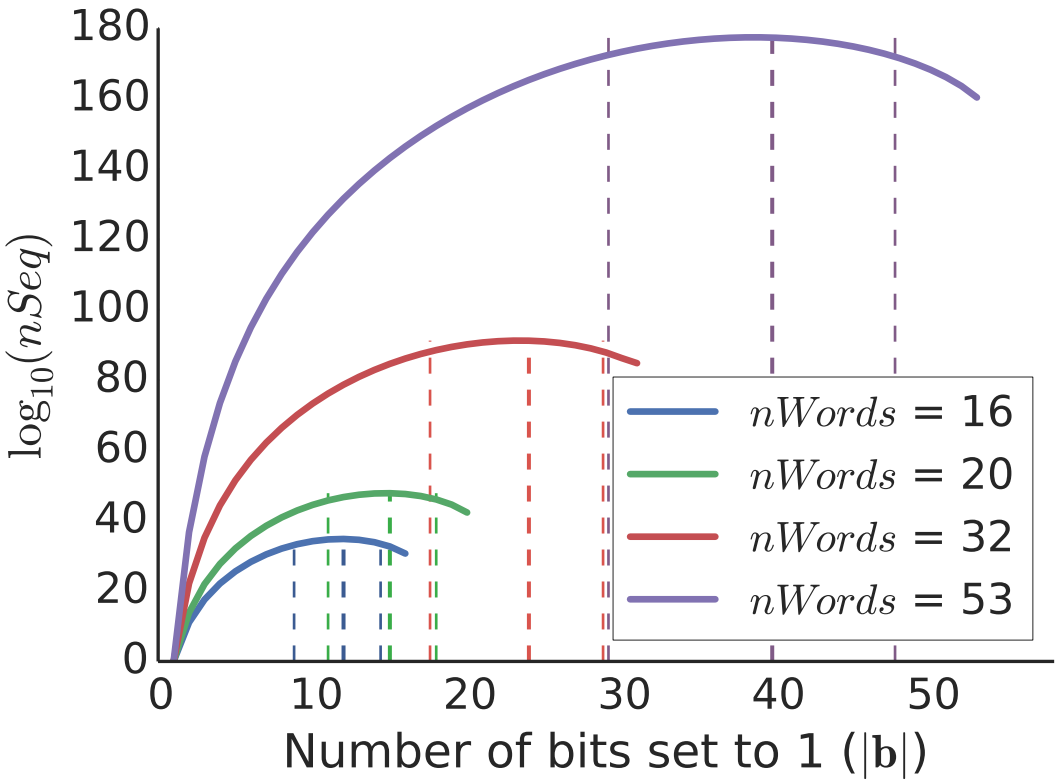
- Question: How many original sequences lead to a single protected template?
- Bloom filter indexes are visible to an attacker ⇒ the reconstruction of the corresponding binary block involves an arrangement of I**b**| < *nWords* (I**b**| = # activated bits) different words to a binary block of length *nWords*.
- By the inclusionexclusion principle, the total number of possible sequences *nSeq* is:

$$nSeq = \sum_{i=1}^{|\mathbf{b}|} (-1)^{|\mathbf{b}|-i} \binom{|\mathbf{b}|}{i} i^{nWords}.$$

- And then, we have to undo the permutation

Bloom Filters

Irreversibility



- We estimate $|\mathbf{b}|$ over a particular database (e.g., Biosecure Multimodal DB):

  $|\mathbf{b}| = 56.3$

- With that value, we have

  $nSeq = 2^{40}$

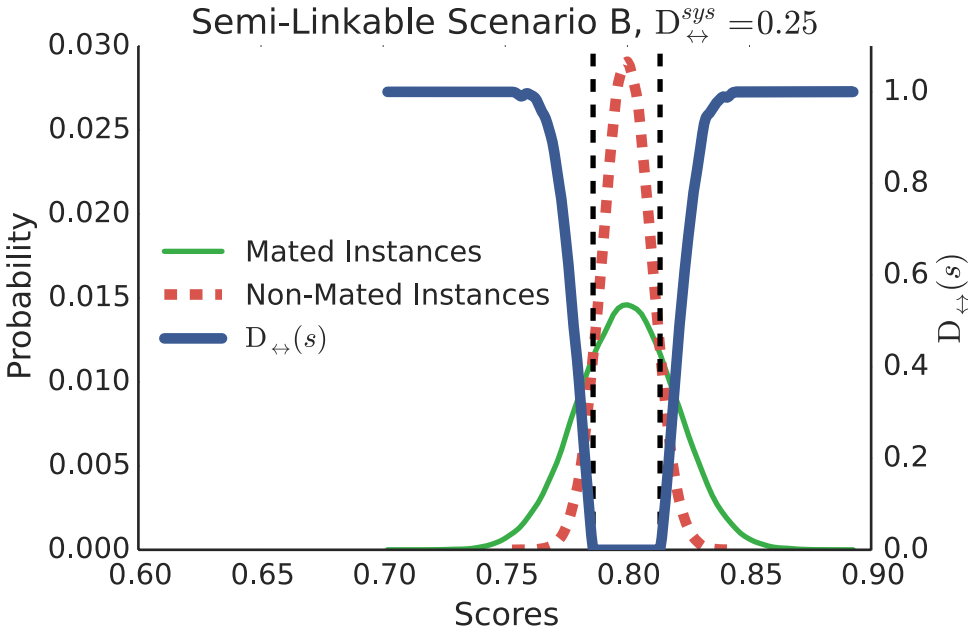
- For a full disclosure model, the probability of a reconstruction is

  $\mathbf{2^{-40,960}}$

[Gomez-Barrero *et al.*, *Information Sciences* 2016]

Unlinkability Analysis

- We have to compute again mated and non-mated scores
- But now each sample is enrolled in a different system!
- That means, a different key
- Ideally, at least 10 different keys, but let‘s now do only 5
- So now, create database_BF_2 to database_BF_5



Unlinkability Analysis

- Now, we can reuse the comparison lists that we have, but we have to adapt the `computeScores.py` script to work with 2 databases

- Create now `computeScores_Unlink.py`

- `parser.add_argument('DB_BFtemplates_2', help='directory where the protected BF templates are stored', type=str)`

- `DB_BFtemplates_2 = args.DB_BFtemplates_2`

- `filename2 = DB_BFtemplates_2 + '/' + r[1]`

Unlinkability Analysis

- Python **computeScores_Unlink.py** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**database_BF** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**database_BF_2** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**matedComparisonsBF.txt** /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**nonMatedComparisonsBF.txt** --scoresDir /Users/marta/Documents/hs-ansbach/Python/VISUM2020/**BFScores_Unlink**

Unlinkability Analysis

- Instead of all possible DB combinations, let‘s now compare DB 1 to BD 2 to 5

- And store the score files in a new folder BFScores_Unlink

```
usage: evaluateUnlinkability.py [-h] [--omega [OMEGA]] [--nBins [NBINS]]
                [--figureTitle [FIGURETITLE]]
                [--legendLocation [LEGENDLOCATION]]
                matedScoresFile nonMatedScoresFile figureFile

Evaluate unlinkability for two given sets of mated and non-mated linkage
scores.

positional arguments:
  matedScoresFile       filename for the mated scores
  nonMatedScoresFile    filename for the non-mated scores
  figureFile            filename for the output figure

optional arguments:
  -h, --help            show this help message and exit
  --omega [OMEGA]       omega value for the computations, if none provided,
                        omega = 1
  --nBins [NBINS]       number of bins for the computations, if none provided,
                        nBins = 100
  --figureTitle [FIGURETITLE]
                        title for the output figure
  --legendLocation [LEGENDLOCATION]
                        legend location
```

Unlinkability Analysis

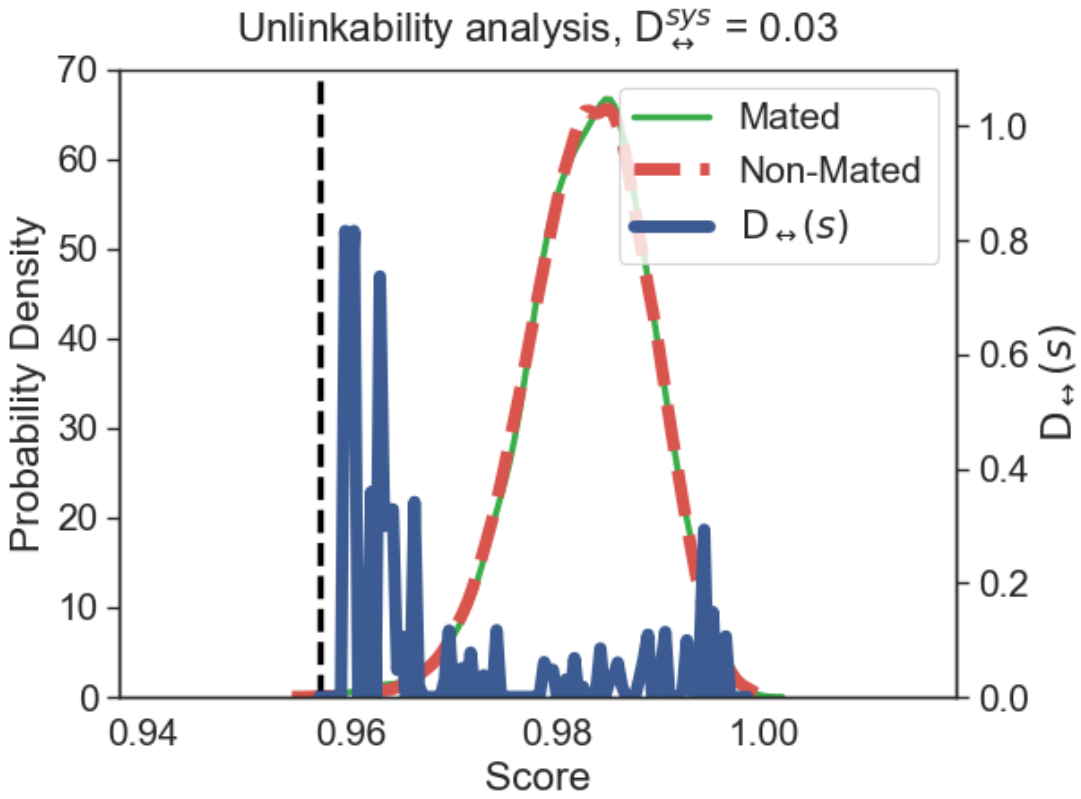
- Here we need a single list with all mated scores in a hdf5 file (and similar for the non-mated

```
matedScores = []

for i in range(2,6):
    mated = scoresDir + 'matedScoresBF_1' + str(i) + '.txt'
    matedScores.append(numpy.loadtxt(mated))


with open(scoresDir+'matedScoresBF.hdf5', 'w') as f:
    numpy.array(matedScores).tofile(f)
```

Unlinkability Analysis

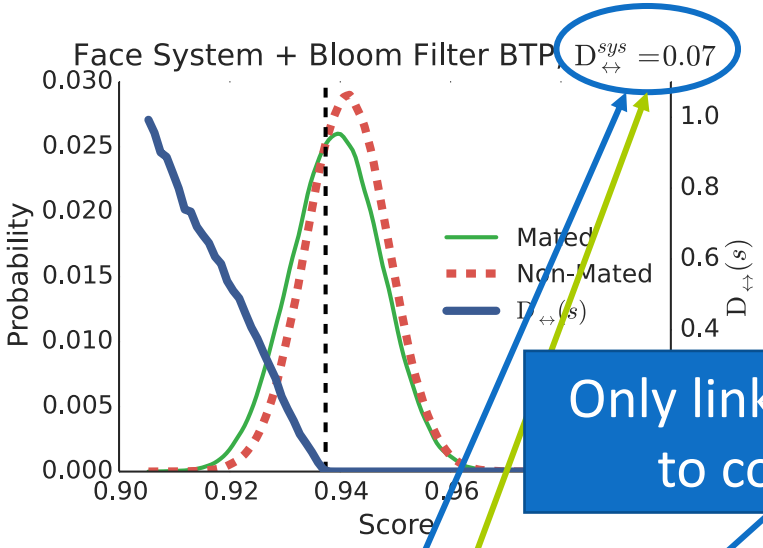
```
Python evaluateUnlinkability.py
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/BFScores_Unlink/matedScores
BF.hdf5
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/BFScores_Unlink/nonMatedSco
resBF.hdf5
/Users/marta/Documents/hs-
ansbach/Python/VISUM2020/unlinkAnalysis.png
```

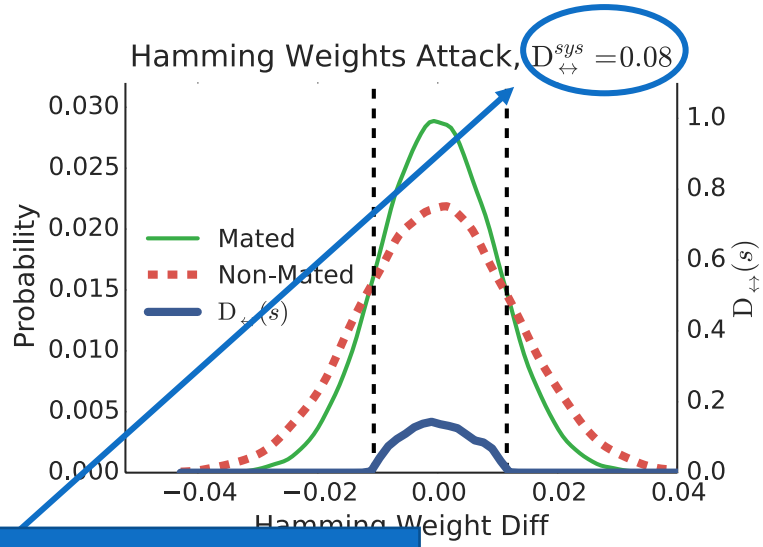
Unlinkability Analysis



Bloom filters

Unlinkability analysis (I)

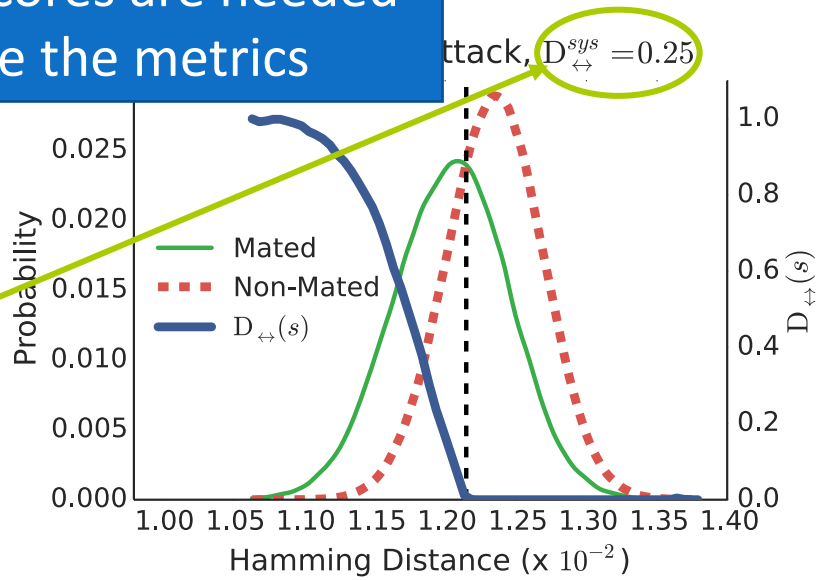




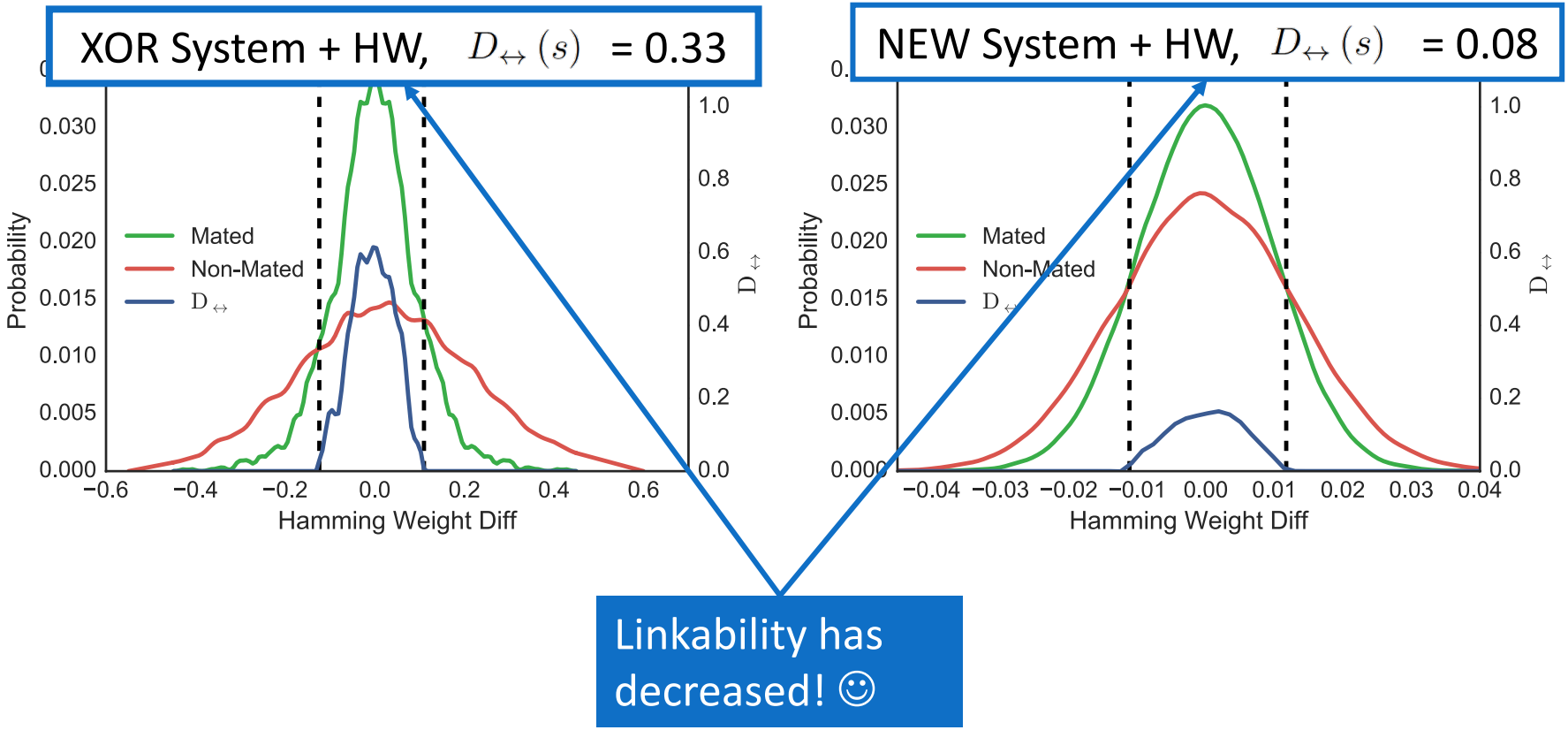
Only linkage scores are needed to compute the metrics

Linkability has barely increased ☺

Still room for improvement



Unlinkability analysis (II)



Prof. Dr. Marta Gomez-Barrero
(marta.gomez-barrero@hs-ansbach.de)

HOCHSCHULE
ANSBACH