Evaluación Comparativa de Herramientas de Software de Código Abierto para Propósito en la Verificación del Hablante

Nicolás Bohórquez Guerrero, Luider Andrés Heredia Heredia

Trabajo de Grado para Optar al Título de Ingenieros Electrónicos

Director

Franklin Alexander Sepúlveda Sepúlveda

Doctorado en Ingeniería

Codirector

Dagoberto Porras Plata

Ingeniero Electrónico

Universidad Industrial de Santander

Facultad de Ingerías Físicomecánicas

Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones (E3T)

Bucaramanga

2017

En primer lugar a mi Dios por la vida y el amor que nos brinda, a mi madre que siempre estuvo presente con un apoyo incondicional, dándome aliento y brindándome un cariño maternal, a mi padre porque nunca nos desamparo y creyó en mí y en que esto fuera posible, a mis abuelos que siempre me apoyaron y nunca me desampararon, por ese amor y cariño, mis amigos por esos momentos inolvidable y por ultimo a todas aquellas personas que no creyeron en mí, ya que fueron las persona que me ayudaron a ser más fuete y caminar con la frente en alto.

Luider Andres Heredia Heredia

A mi madre y a mi abuela materna, por ser mi motivación en todo esto y estar siempre a mi lado apoyándome con su amor incondicional

A mis amigos, por el tiempo compartido durante esta etapa.

Al profesor Franklin Alexander Sepúlveda por brindarme la oportunidad de trabajar en este proyecto de grado. Asimismo, como su tiempo y experiencia.

Por último, a aquellas personas que no creyeron esto posible, pues su escepticismo fue la fortaleza en este arduo camino.

Nicolás Bohórquez Guerrero

Tabla de contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Glosario

ASV: automatic speaker verification

CMS: cepstral mean subtraction

CMVN: cepstral mean variance normalization

CQCC: constant q cepstral coefficients

DCF: detection cost function

EER: equal error rate

EM: expectation maximization

GMM: gaussian mixture model

HTK: hidden markov model toolkit

MAP: maximum a posterior

MFCC: mel-frequency cepstral coefficients

MLSA: mel log spectral approximation

PLDA: probabilistic linear discriminant analysis

PLP: perceptual linear prediction

RASTA: relative spectra

SPRO: speech signal processing

SS: speech synthesis

TV: total variability

UBM: universal background model

VAD: voice activity detection

VC: voice conversión

RESUMEN

Título: "Evaluación Comparativa de Herramientas de Software de Código Abierto para Propósito en la Verificación del Hablante"*

Autores:
Nicolás Bohórquez Guerrero
Luider Andrés Heredia Heredia **

Palabras clave: MSR Identity, SIDEKTI, SPEAR, ALIZE, Verificación del hablante, GMM, ivector, código abierto

Descripción

Este proyecto presenta y describe la comparación de 4 herramientas de código abierto usadas en el ámbito de la verificación automática del hablante. Para este objetivo, se llevó a cabo la implementación de dos sistemas, el primero está basado en modelos de mezclas gaussianas y el segundo modela las variabilidades dependientes del hablante como un subespacio. Además de esto, se utilizaron los archivos de audio contenidos en la base de datos ASVspoof 2015, de uso público, la cual contiene archivos de audio mixtificados que son creados con sintetizador y conversor de voz, usados para observar la vulnerabilidad de un sistema ante este tipo de ataques; implementando así medidas ampliamente aceptadas para evaluar estos sistemas biométricos y dando una apreciación de manera subjetiva.

Ahora bien, el sistema se implementa en las herramientas configurando parámetros en común con un mismo valor. De igual manera se usa un conjunto de audios para entrenamientos y otro para pruebas.

Por último, los resultados permiten observar el desempeño de estas herramientas e igualmente permite compararlas no solo en este aspecto sino también en sus características. Adicionalmente se muestra la estructura de los archivos necesarios para la implementación de los dos tipos de sistemas utilizados, dando así una visión holística de estas herramientas.

---

* Trabajo de grado
** Facultad de Ingenierías Físico-mecánicas, Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director. PhD Franklin Alexander Sepúlveda Sepúlveda

ABSTRACT

Título: "Comparative evaluation of open code software tools for the purpose of speaker verification"*

Authors:
Nicolás Bohórquez Guerrero
Luider Andrés Heredia Heredia **

Keywords: MSR Identity, SIDEKTI, SPEAR, ALIZE, Speaker Verification, GMM, ivector, Open Source

Description

This project presents and describes the comparison of 4 open source tools used in the field of automatic speaker verification. For this purpose, the implementation of two systems was carried out, the first one is based on Gaussian mix models and the second one models the dependent variability of the speaker as a subspace. In addition to this, the audio files contained in the ASVspoof 2015 database, for public use, which contains mystified audio files that are created with a synthesizer and a voice converter, used to observe the vulnerability of a system before this, were used. type of attacks; thus, implementing widely accepted measures to evaluate these biometric systems and giving an appreciation in a subjective manner.

Now, the system is implemented in the tools by configuring parameters in common with the same value. In the same way, a set of audios is used for training and another for tests.

Finally, the results allow us to observe the performance of these tools and also allows us to compare them not only in this aspect but also in their characteristics. Additionally, the structure of the files necessary for the implementation of the two types of systems used is shown, giving a holistic view of these tools

* Degree work
** Faculty of Physical-Mechanical Engineering, School of Electrical Engineering, Electronics and Telecommunications. Director. PhD. Franklin Alexander Sepulveda

Introducción

La era digital que vivimos presenta oportunidades para la implementación de nuevas formas de control e identificación más avanzadas. Entre ellas se mencionan el reconocimiento automático de un individuo, basado en sus características biológicas medibles, las cuales se caracterizan por ser formas de identificación que se apoya firmemente en el reconocimiento de una característica física de una persona como lo son: las huellas dactilares, firma, voz, iris, patrones de la retina, geometría de la mano, termografía del rostro, entre otras. Debido a que estas características no son imitables ni transferibles con facilidad, como si lo es una contraseña o una tarjeta, el utilizar estas señales resulta de importancia para mejorar la seguridad en otros sistemas. Así mismo, en la ciencia forense y el sistema penal de justicia se hace uso de algunos de estos sistemas biométricos para la identificación de víctimas y victimarios involucrados en diferentes tipos de casos y denuncias, permitiendo evitar condenar de forma errónea a un inocente e identificar de forma precisa al culpable.

La biometría traduce nuestros rasgos biológicos en datos, para luego identificar o verificar la identidad de una persona. Algunos ejemplos de sistemas biométricos son el reconocimiento de la huella dactilar, de la cara, del iris, de la retina, geometría de la mano y verificación del hablante, entre otros. Debido a que estas características no son imitables ni transferibles con facilidad, como si lo es una contraseña o una tarjeta, el utilizar estas señales resulta de importancia para mejorar la seguridad en otros sistemas. Así mismo, en la ciencia forense y el sistema penal de justicia se hace uso de algunos de estos sistemas biométricos para la identificación de víctimas

y victimarios involucrados en diferentes tipos de casos y denuncias, permitiendo evitar condenar de forma errónea a un inocente e identificar de forma precisa a la culpable.

Los sistemas biométricos tienen ventajas y desventajas. Por ejemplo, el reconocimiento facial es uno de los métodos que involucra la detección de características distintivas del rostro por lo cual no es intrusivo y es probablemente la característica biométrica más comúnmente usada por las personas para reconocer a otros. Sin embargo, este sistema posee restricciones a la hora de obtener las imágenes ya que a menudo son sensibles al entorno y la iluminación. Adicionalmente se tienen dificultades a la hora de hacer coincidir imágenes de dos diferentes vistas, realizadas con diferente iluminación y distintos tiempos (Handbook of Biometrics).

El sistema biométrico de interés en este estudio es la verificación automática del hablante (ASV, por sus siglas en inglés), la cual tiene por tarea el verificar la identidad de una persona a partir del análisis de la voz. El análisis entrega una decisión de tipo binario, es decir, se acepta o se rechaza la identidad de la persona.

Los sistemas ASV junto a los de identificación automática del hablante (ASI, por sus siglas en inglés) son uno de los métodos más económicos y naturales para resolver los problemas acceso (Campbell Jr, 1997). Adicionalmente, es muy versátil, no es intrusivo en comparación con otros sistemas, y además de ser relativamente preciso, no requiere ningún equipo especial. Finalmente, se cuenta con herramientas de código abierto para la investigación y el desarrollo de los mismos (Bonomo Laynez, 2012).

Sin embargo, la validez y confiabilidad varían dependiendo de las técnicas utilizadas en las herramientas en particular a utilizar. Validez y confiabilidad se traducen en precisión y exactitud respetivamente. Por tanto, surge la necesidad de conocer y medir que tan exactos y precisos son los sistemas de verificación del hablante que se pueden obtener por medio de librerías de código

abierto. Por esta razón el evaluar dichos sistemas se convierte en una tarea de carácter importante.

El presente informe se centra en realizar pruebas sobre herramientas usadas para la verificación del hablante disponibles, de código abierto y reportadas en trabajos académicos. Para la realización de las pruebas se plantea el uso de bases de datos disponibles en la web para propósitos de evaluación de sistemas de verificación del hablante; las cuales, generalmente se encuentran disponibles para el idioma inglés. En el presente trabajo se analizan herramientas de desarrollo de sistemas de verificación del hablante del tipo código abierto. Entre ellas se destacan *MSR Identity Toolbox* y *ALIZE/LIA_RAL*.

Finalmente, la evaluación de las herramientas usadas en estos sistemas nos ofrece una visión clara de sus fortalezas y debilidades, esto es útil pues proporcionaría información sobre cual herramienta se adapta a sus necesidades.

El informe viene estructurado de la siguiente manera: En el principio se hará una breve introducción al contenido teórico de los sistemas de verificación del hablante, en la cual se muestran los pasos que deben seguir estas herramientas al momento de hacer un análisis. A continuación, se describe cada una de las herramientas que se usaron para llevar a cabo la implementación de los sistemas. Después de esto se habla de la configuración de las herramientas usadas para realizar las pruebas. Por último, se mostrarán los resultados obtenidos de los sistemas implementados.

1 Objetivos

Como objetivo general de esta investigación se ha establecido comparar, mediante medidas usadas en el estado del arte, el desempeño de herramientas de software de verificación del hablante de código abierto.

- ✓ Definir las condiciones de prueba y los criterios de comparación para medir el desempeño de sistemas de verificación del hablante.
- ✓ Implementar algoritmos de verificación de hablante basado en librerías y herramientas de código abierto.
- ✓ Analizar el desempeño de los sistemas de verificación del hablante desde un punto vista cuantitativo y cualitativo.

2 Sistema de Verificación del Hablante

El reconocimiento del hablante se divide en 2 ramas, la verificación del hablante y la identificación del hablante. En la verificación se devuelve un resultado del tipo binaria. Esto quiere decir que acepta o rechaza la hipótesis de la pertenencia de una huella de voz desconocida a una identidad previamente inscrita. Por otra parte, la identificación realiza una comparación de la huella de voz desconocida con los modelos inscritos previamente encontrando la mejor concordancia y así identificar aquella persona que hablo dentro de un conjunto de personas predeterminado (Mendoza Marín & Porras Plata, 2016).

Figura 1. Proceso verificación del hablante

Se hará una breve descripción de las diferentes tareas de reconocimiento que puede llevar a cabo un sistema de verificación del locutor.

Todo sistema de reconocimiento del habla debe tener en general un área de identificación y otra de verificación. Donde el área de identificación se encarga de crear un modelo para cada uno de los N locutores por medio de la clasificación de las señales de audio, esta parte se puede dividir en dos casos. Una parte de identificación en conjunto cerrado, en este subproceso tenemos como resultado una asignación de identidad a uno de los locutores, el cual es procesado por el sistema y reconocido como "usuario". El otro caso es llamado identificación de conjunto abierto, en el cual se analiza la posibilidad de que el locutor identificado no pertenece a los "usuarios".

el área de verificación es la encargada de darle aceptación o rechazo a la hipótesis "de que ambas locuciones pertenezcan al mismo locutor o *persona*", en donde una de las locuciones es el audio por verificar y la otra es la identidad del locutor a verificar (Sigüenza, 2008).

2.1 Representación de la Señal de Voz

En esta etapa la señal de audio es procesada para convertirla en información útil para el sistema; es decir, información más compacta, menos redundante y más adecuada (Bimbot, et al., 2004). Las características más usadas son *PLP (Perceptual Linear Prediction)* y *MFCC (Mel-Frecuency Cepstral Coefficients)*. Para obtener estos parámetros la señal de audio es dividida en franjas de N* [ms], estas franjas se transforman en un vector de nuevos parámetros.

* Franja de tiempo en la cual se divide la señal de audio la cual se da en milisegundos

En el presente trabajo se van a utilizar los coeficientes de la escala de MEL (MFCC) ya que son los más utilizados actualmente en los reconocedores comerciales debido a su robustez; además, hacen uso de la Transformada de Fourier y la Transformada rápida de Fourier para poder obtener las frecuencias de la señal (Bonomo Laynez, 2012).

Figura 2. Extracción de características MFCC

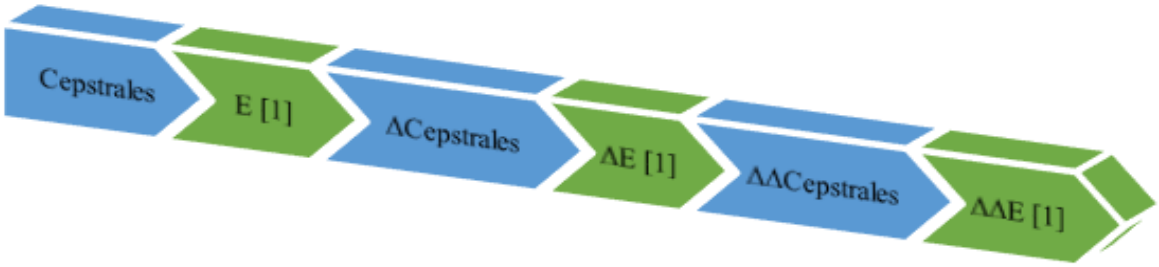


Figura 3. Formato del vector de características

Este vector de características tiene la forma que se observa en la figura 3, que generalmente se le asigna un tamaño a los cepstrales de M* coeficientes y el logaritmo de la energía siempre es de tamaño 1. Dando esto como resultado un vector de tamaño Z** (M+1+M+1+M+1).

Una vez la señal de audio se ha convertido en una nueva representación, dichos parámetros contienen además de la voz de los hablantes vectores sin ninguna información acerca del hablante. Esto es, partes de silencio o ruido de fondo que se deben descartar. Esta etapa se realiza mediante un procedimiento denominado *VAD* (*Voice Activity Detection*) y puede

* Numero de coeficientes cepstrales
** Tamaño de la matriz de coeficientes

realizarse directamente sobre las características *MFCC* usando el coeficiente asignado a la energía para así descartar los vectores *MFCC* con nivel de energía bajo.

2.1.1 Normalización CMVN (Cepstral Mean and Variance Normalization).

La normalización es una técnica de compensación que incrementa la robustez del sistema. Debido a los diversos factores que pueden degradar el rendimiento del sistema tales como condiciones de grabación, ruido y variabilidad del hablante, entre otras, se aplican técnicas para mitigar dichos efectos aproximando la representación de parámetros a distribución gaussiana o normal. Dentro de este tipo de técnicas se mencionan: *RASTA (RelAtive SpecTrA), CMS (cepstral mean subtraction) y CMVN* (*cepstral mean variance normalization*). En el presente trabajo se utiliza normalización por *CMVN*, la cual tiene la propiedad de reducir efectos ocasionadas por la variabilidad del canal (Bonomo Laynez, 2012).

2.2 Entrenamientos de modelos

En esta parte se mostrará cómo se entrenan los diferentes modelos necesarios para los sistemas *GMM* e *I-VECTOR*, lo cual es un paso necesario para la ejecución de este proyecto, en la cual veremos cómo se entrena el UBM, extracción de características *BW*, entrenamiento de la matriz *TV*, extracción de *I-VECTOR* y modelos *PLDA*.

2.2.1 UBM.

En esta etapa se entrenan modelos necesarios tales como el *UBM (Universal Background Model)*, que busca representar, como su nombre lo indica, el conjunto universal de hablantes. Es un modelo entrenado con una gran cantidad de datos y una adecuada cantidad de hablantes, pues debe representar el espacio de posibles alternativas y la variabilidad de la población. Este modelo corresponde a una función de densidad de probabilidad construida a partir de funciones gaussianas ponderadas cada una con sus matrices para la media y la varianza.

En la verificación del hablante el *UBM* se constituye mediante un modelo de mezclas gaussianas (*GMM*) independiente del hablante (Reynolds, Gaussian Mixture Models, 2015). Este modelo se usa como modelo de referencia para la inscripción de los hablantes, también es usado en el entrenamiento de la matriz *TV* el cual es la matriz de cofactores y modelo *PLDA* que es la técnica encargada de la varianza de intraclase e interclase como gaussianas multidimensionales el cual es un método bastante utilizado en algoritmos de reconocimiento *EM* (Matějka, et al., 2011) y (Prince & Elder, 2007, October). El algoritmo ampliamente usado para entrenar el UBM es el *EM (Expectation-Maximization)*. En cuanto al número de hablantes o la cantidad de oraciones no existe una medida objetiva (Reynolds, Universal Background Models, 2015).

Sin embargo, en trabajos anteriores se reporta que usan entre tatos y tantos hablantes para construir el UBM.

En los sistemas basados en mezclas gaussianas (*GMM*) se realiza la inscripción de los hablantes en el sistema usando *MAP (maximum a posterior)*, esta adaptación se expresa de la siguiente manera:

$$\theta_{MAP} = argmax_{\theta} f(\theta|X_1, X_2, \dots X_n)$$

Donde θ es el parámetro por maximizar, f define la función de probabilidad que relaciona el parámetro y X las características dadas. La inscripción consiste en tomar las características que identifican a un hablante y realizar un modelo del hablante el cual se usa para una posterior comparación con un audio desconocido.

El *UBM* se usa como modelo previo para realizar el tipo de modelamiento antes mencionado. Se genera adaptando del *UBM*, el cual consta de dos pasos al igual que la estimación *EM*. En el primer paso, se tiene en cuenta la estadística suficiente para estimar los parámetros deseados, peso, media y varianza para cada *GMM*. En el siguiente paso, esta nueva estadística de los datos de entrenamiento es usada para actualizar la estadística previa de la mezcla y crear parámetros adaptados para la mezcla (Reynolds, Gaussian Mixture Models, 2015).

En este punto se manejan dos hipótesis h_0(el audio de prueba X pertenece al modelo Y) o h_a(el audio de prueba X no pertenece al modelo Y), la forma de hacer una decisión es con el índice de probabilidad dado por la forma:

$$\frac{p(X|h_0)}{p(X|h_a)} \begin{cases} > \theta \ se \ acepta \ h_0 \\ < \theta \ se \ acepta \ h_a \end{cases}$$

Donde θ es el umbral del sistema y p(X|H) es la función de densidad de probabilidad para la hipótesis dada, donde esta función de probabilidad en un sistema *GMM* se define como una combinación lineal de densidades gaussianas (Bimbot, et al., 2004).

2.2.2 Matriz de variabilidad total TV (Total Variability).

La matriz *TV (Total Variability)* es un modelo propio de los sistemas *ivectors*. En este modelo se busca representar tanto la variabilidad del hablante como la de las sesiones en un subespacio de

baja dimensión, en conjunto con el *UBM* representa el modelo del hablante en forma de supervector, de la siguiente manera.

$$S = \mu + T\omega$$

Donde S y μ representan los modelos del hablante y del *UBM* respectivamente, T define a matriz *TV* y ω es la variable latente también llamada *ivector*[*]

La variable ω puede ser definida por medio de una distribución posterior condicionada a las, estadísticas de *Baum-Welch*, esta distribución es una distribución Gaussiana y la media de esta distribución corresponde al *ivector*, estas estadísticas son extraídas usando el *UBM*. Teniendo una secuencia de N franjas $\{X_1, X_2, \ldots, X_N\}$ y una *UBM* $\{\Omega\}$ compuesta por M componentes de mezclas, se obtienen las estadísticas por

$$N_c = \sum_{t=1}^{N} P(c|X_t, \Omega)$$

$$F_c = \sum_{t=1}^{N} P(c|X_t, \Omega) X_t$$

Donde N_c y F_c son las ecuaciones de momento 0 y 1 respectivamente. Además, c es el índice de Gaussianas y $P(c|X_t, \Omega)$ representa la probabilidad posterior de que la franja X_t es generada por la componente de la mezcla c (Dehak, Kenny, Dehak, Dumouchel, & Ouellet, 2011).

Ya que contiene las variabilidades antes mencionadas y como es usada tanto en la representación de todos los modelos y las extracciones, se considera que un *ivector* contiene la suficiente información para diferenciar una extracción de otra (Domínguez, Zazo, & González Rodríguez, 2012).

[*] Los vectores en el subespacio de baja dimensión son llamados *ivectors*

2.2.3 Extracción i-vector.

Esta etapa requiere de un primer paso que es el entrenamiento del *UBM* descrito en el capítulo 2.2.1. seguido de esta etapa se procede a entrenar la matriz *TV* por medio de una variante de algoritmo *EM* para variables aleatorias ocultas. El proceso que siempre se ha utilizado es el descrito en el capítulo 2.2.2, después de obtener las matrices de Variabilidad Total se procede a extraer *I-Vector* véase (Kenny P. , 2012, June).

2.2.4 Modelo PLDA.

El modelo *PLDA (Probabilistic Linear Discriminant Analysis)* fue implementado al principio para reconocimientos faciales (Domínguez, Zazo, & González Rodríguez, 2012). Actualmente también es usado en los sistemas *ivectors*. Es un modelo discriminativo basado en la probabilidad, donde la variabilidad del hablante y de las sesiones, son modeladas en subespacios separados descomponiendo los *I-vectors* de la siguiente manera:

$$\omega_{ij} = \mu + Fh_i + Gw_{ij} + \epsilon_{ij}$$

Donde ω_{ij} denota el j-ésimo *ivector* del i-ésimo individuo, μ es la media total de los *ivectors*, F es la matriz de variabilidad entre individuos y h_i es una variable latente de identidad. La matriz G contiene la variabilidad dentro del individuo y el termino w_{ij} denota la posición en este subespacio, ϵ_{ij} representa un término de ruido estocástico. Los dos primeros términos del modelo permanecen constantes para un modelo de hablante dado, y los dos últimos términos son los que explican por qué 2 audios de un mismo hablante no son idénticos (Prince, Li, Fu, Mohammed, & Elder, 2012).

El uso de las matrices F y G en el modelo. Donde se puede apreciar que, para una cantidad de datos dada, durante el entrenamiento del *PLDA* las matrices F y G se adaptan iterativamente para modelar las varianzas. Donde F busca maximizar las distancias entre las clases y G minimizarlas.

En un sistema *ivector-PLDA* dado dos *ivectors* ω_e (para inscripción) y ω_t (para prueba) la comparación viene dada por:

$$S(\omega_e, \omega_t) = \frac{p(\omega_e, \omega_t|H_0)}{p(\omega_e|H_a)p(\omega_t|H_a)}$$

La hipótesis H_0 indica que los *ivectors* son iguales. Esto quiere decir que tienen la misma variable latente de identidad descrita anteriormente como h. Por otro lado H_a indica que vienen de diferentes modelos (h diferente). Se asume que las variables h y w del modelo *PLDA* tienen una distribución normal estándar (Rajana, Afanasyev, Hautamäki, & Kinnunen, 2014)

2.3 Criterios de evaluación de sistemas de Verificación del Hablante

Estos sistemas no están libres de errores. Se presentan dos tipos de errores importantes que son el *FAR* (error tipo II) que corresponde al porcentaje donde impostores son aceptados, y *FRR* (error tipo I) que corresponde al porcentaje donde los genuinos son rechazados. Medir estos errores sirve para conocer el desempeño del sistema. La herramienta que muestra estos errores es la curva *DET (Detection Error Tradeoff)*. Un ejemplo de la curva DET se muestra en figura 4, en la cual se hace un barrido del umbral y se van calculando los errores. En donde, en el eje de las abscisas se ubica *FAR* y en el eje de las ordenadas *FRR*. Esta curva es ampliamente usada en la evaluación de sistemas de verificación del hablante y está incluida dentro de los estándares del

NIST (National Institute of Standards and Technology) (Martin, Doddinggton, Kamm, Ordowski, & Przybocki, 1997).

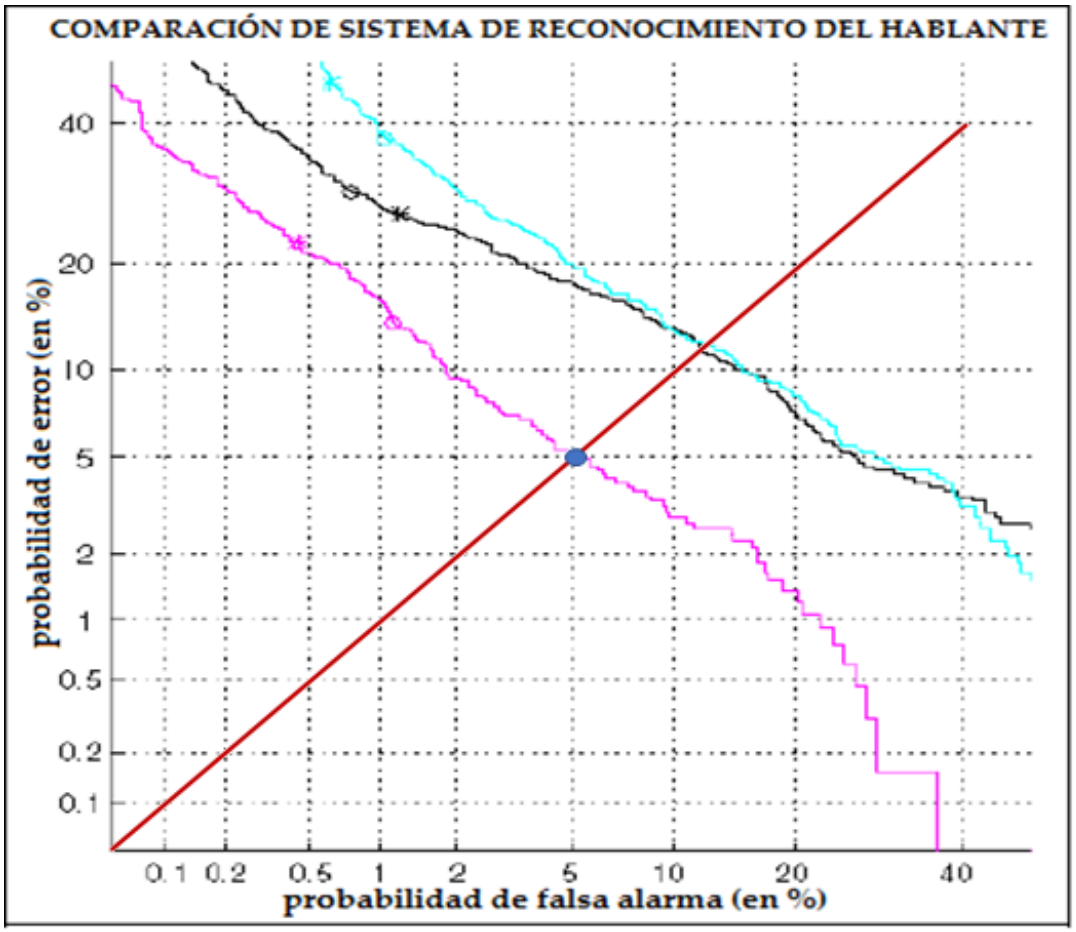


Figura 4. Curva *DET*. Adaptado de (Martin, Doddinggton, Kamm, Ordowski, & Przybocki, 1997)

Otra medida implementada en las evaluaciones del *NIST* es el *EER*, el cual está representado por el punto de color azul en la gráfica descrita anteriormente, donde los dos errores son iguales (línea de 45 grados), como se observa el sistema con el más bajo *EER* es el que corresponde al color fucsia con un 5%.

Adicionalmente, el *NIST* usa una función de detección de costo C_{Det} para medir el desempeño de los sistemas. Esta función asigna pesos a los errores antes mencionados de la siguiente manera:

$$C_{Det} = C_{miss} * P_{FRR} * P_{targ} + C_{FA} * P_{FAR} * (1 - P_{targ})$$

Donde, P_{targ} es la probabilidad a priori de que ocurra un evento de hablante objetivo en el sistema, C_{miss} es el costo de falso rechazo, C_{FA} es el costo de una falsa aceptación. P_{FRR} y P_{FAR} dependen del umbral. Hallando el umbral que hace que esta función sea mínima es lo que se conoce como mínimo *DCF*. Por otro lado, los valores de los pesos y la probabilidad son propuestos por el *NIST* en sus evaluaciones de reconocimiento del hablante las cuales realizan desde 1996. Los valores más usados en esta tarea son los propuestos en el año 2008 y 2010 (Martin, Doddinggton, Kamm, Ordowski, & Przybocki, 1997).

2.4 Engaños y Ataques Usados en Sistemas de Verificación del Hablante

Los sistemas de reconocimiento del hablante tienden a ser afectados por engaños y ataque de usuarios maliciosos, que buscan engañar al sistema y poder lograr modificar el resultado de verificación para así permitir acceso a quienes legítimamente no deberían tenerlo. Estas técnicas son conocidas como *SPOOFING*[*]. Actualmente hay diferentes tipos de *SPOOFING* los cuales haremos una breve introducción (Sánchez de la Fuente, 2016).

Suplantación: Se refiere a los ataques realizados con una voz humana alterada o dicho de otra manera una imitación de la voz original, para realizar este ataque no es necesario el uso

[*] Así es llamado el uso de técnicas de suplantación de identidad generalmente para uso malicioso.

de ningún dispositivo. Este ataque es considerado una técnica más efectiva en oyentes humanos que una amenaza real en sistemas *ASV* (Sánchez de la Fuente, 2016) y (Evans, Kinnunen, & Yamagishi, 2013, August).

Grabación: Esta consiste en que el atacante debe obtener o realizar grabaciones del hablante original y así poder reproducirlas ante el sistema para poder conseguir acceso, para realizar este tipo de ataque no se necesita un conocimiento muy avanzado en tecnología (Sánchez de la Fuente, 2016) y (Evans, Kinnunen, & Yamagishi, 2013, August).

Síntesis del habla: Por medio de esta técnica se puede convertir un texto de entrada en elementos lingüísticas y a partir de estos elementos se puede generar una voz. Los métodos antiguos requieren una gran cantidad de datos de diferentes hablantes con transcripciones preparadas cuidadosamente para poder crear los modelos, por otra parte, los sintetizadores del estado del arte basados en modelos ocultos de *Markov* pueden aprender modelos de relativamente pocos hablantes (Sánchez de la Fuente, 2016) y (Evans, Kinnunen, & Yamagishi, 2013, August).

Conversión de voz: Este tipo de ataque busca tomar la voz de una persona dada y hacer que sea similar a la de otro locutor, la diferencia con el ataque de síntesis es que la entrada de esta técnica son voces y no textos (Sánchez de la Fuente, 2016) y (Evans, Kinnunen, & Yamagishi, 2013, August).

3 Herramientas del Sistema

3.1 Base de Datos Experimentales

La base de datos usada para evaluar los sistemas es *ASVspoof* 2015. Se trata de una base de datos pública disponible en (Wu, Kinnunen, Evans, & Yamagishi, Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof 2015) Database, 2015). Se encuentra bajo licencia del tipo CC-BY* el cual permite al usuario copiar, modificar, remezclar, entre otras acciones al archivo original.

Tabla 1 *Número de hablantes y oraciones*

Subconjuntos	N.º Hablantes		N.º Oraciones		Tipos de falsificaciones
	Hombres	Mujeres	Genuinos	Falsificaciones	
Entrenamiento	10	15	3750	12625	S1-S5
Desarrollo	15	20	3497	49875	S1-S5
Evaluación	20	26	9404	≈200000	Suplantación, S1-S10

Nota: (Wu, et al., ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge, 2015).

Esta base de datos está compuesta por 106 hablantes, los cuales se dividen en 3 subconjuntos como se aprecia en la tabla 1. No hay solapamiento a lo largo de los 3 subconjuntos, es decir no se repiten hablantes. Las oraciones marcadas como genuinas son

* Esta licencia permite a otros distribuir, mezclar, retocar, y crear a partir de una obra, incluso con fines comerciales, siempre y cuando se de crédito por la creación original. Esta es la más flexible de las licencias ofrecidas. Se recomienda para la máxima difusión y utilización de los materiales licenciados

grabadas sin ninguna modificación y sin efecto de ruido significante de canal o de fondo. El habla falsificada es producto de modificar el genuino usando algoritmos *SS (Speech Synthesis)* y *Synthesis)* y *VC (Voice Conversion)*. El formato de los archivos es *WAV* con una tasa de muestreo de 16 [kHz] y guardado en un formato *PCM* de entero con signo de 16 bits (Wu, et al., ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge, 2015).

Tabla 2 *Algoritmos de ataques implementados en la base de datos*

Subconjunto	N.º Oraciones			*Vocoder**	Algoritmo de falsificación
	Entrenamiento	Desarrollo	Evaluación		
Genuino	3750	3497	9404	Ninguno	Ninguno
S1	2525	9975	18400	*STRAIGHT*	*VC*
S2	2525	9975	18400	*STRAIGHT*	*VC*
S3	2525	9975	18400	*STRAIGHT*	*SS*
S4	2525	9975	18400	*STRAIGHT*	*SS*
S5	2525	9975	18400	*MLSA*	*VC*
S6	0	0	18400	*STRAIGHT*	*VC*
S7	0	0	18400	*STRAIGHT*	*VC*
S8	0	0	18400	*STRAIGHT*	*VC*
S9	0	0	18400	*STRAIGHT*	*VC*
S10	0	0	18400	Ninguno	*SS*

Nota: * *voice coder* es un analizador y sintetizador de voz. Adaptado de (Wu, et al., ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge, 2015).

Una breve explicación de los tipos de falsificaciones mostrados en la tabla 2:

S1 es una selección de franjas donde el habla convertido es generado por selección la selección de franjas del habla del objetivo.

S2 es uno de los algoritmos más simples que ajusta solo el primer coeficiente cepstral para mover la pendiente del espectro de la fuente al objetivo.

S3 implementado con sistemas de síntesis basados en modelos de Markov y usa técnicas de adaptación del hablante y solo 20 pronunciaciones de adaptación, de manera similar S4 es generado, pero con 40 pronunciaciones.

S5 es un algoritmo de conversión de voz implementado con el sistema *Festvox*.

S6 algoritmo basado en modelos de mezclas gaussianas de densidad conjunta y generación de parámetro de máxima verosimilitud considerando la varianza global.

S7 similar al S6, pero usando pares del espectro lineal en lugar de coeficientes cepstrales de mel para la representación del espectro.

S8 es generado usando tensores.

S9 usa un método Kernel de Cuadrados Mínimos Parciales para implementar una función de transformada no lineal.

S10 es un algoritmo implementado con *MARY Text-To-Speech System*.

3.2 Herramientas de Software de Verificación del Hablante

Las herramientas seleccionadas para realizar la comparación son de código abierto, las cuales son: *ALIZE, SIDEKIT, SPEAR y MSR IDENTITY TOOLBOX*.

La mayoría de las herramientas se caracterizan por trabajar con vectores de soporte (SVM) para la calibración y fusión de puntajes, decisión y evaluación. (Khoury, El Shafey, & Marcel,

2014, May) y (Larcher, Lee, & Meignier, An extensible speaker identification sidekit in Python, 2016, March)

3.2.1 Sidekit.

Es una herramienta escrita puramente en *Python* distribuida bajo licencia *LGPL*[*], contiene una lista de componentes incluidas en el estado del arte y permite una rápida realización de un prototipo. Para cada paso de extracción de características se debe realizar una normalización antes y después, la cual posee métodos como *CMS, CMVN* y *feature warping* en una ventana deslizante, detección de actividad del habla esta dispone de métodos basados en la energía; *SNR;* percentiles o la lectura de etiquetas de una fuente externa, modelamiento, puntuación y visualización ofrece un rango de algoritmos estándar (Larcher, Lee, & Meignier, An extensible speaker identification sidekit in Python, 2016, March).

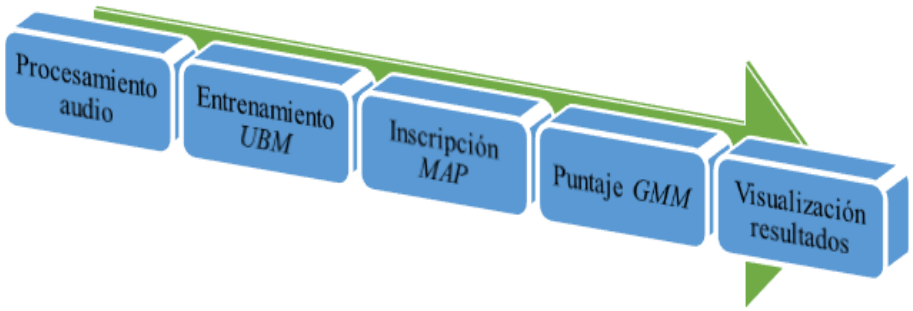


Figura 5. Sistema *GMM-UBM Sidekit*

[*] Esta licencia permite copiar, modificar y la integración con aplicaciones privadas

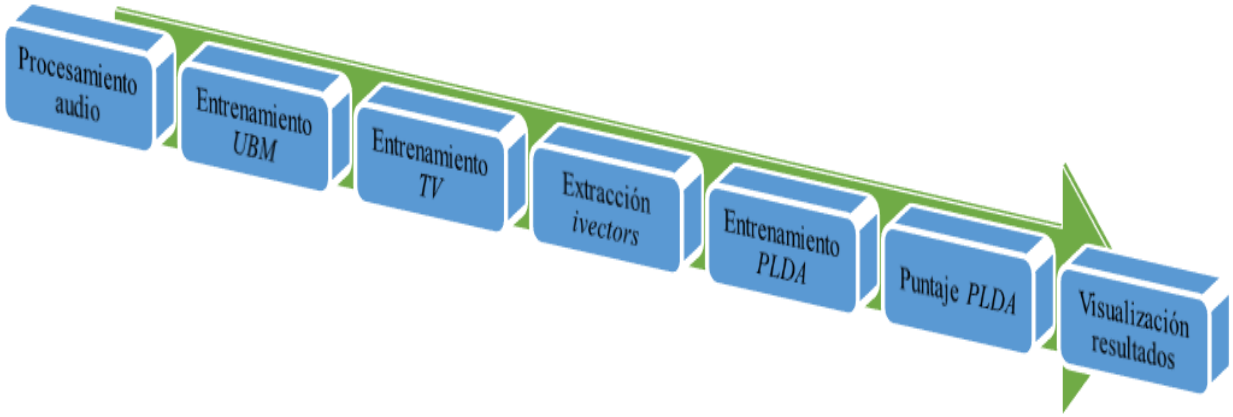
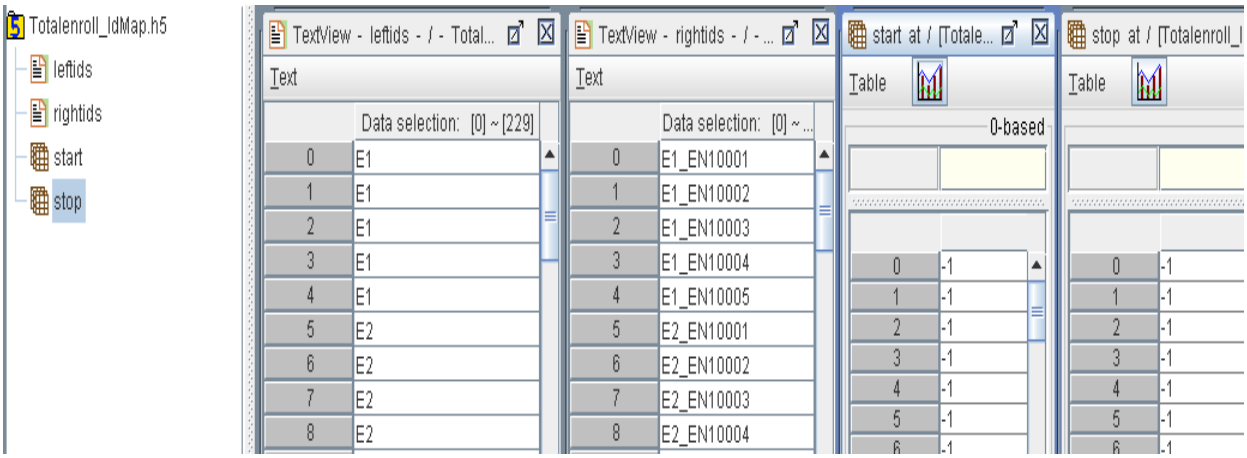


Figura 6. Sistema *ivector-PLDA Sidekit*

Para implementar los procesos de las figuras 5 y 6 la herramienta debe manejar las listas de archivos en formato *HDF5*. Para esto dispone de una clase para crear archivos conocidos como *IdMap, Ndx, Key* y *Score*. El archivo *IdMap* es usado en el sistema *GMM-UBM* para generar estadísticas de los archivos y realizar la inscripción de los hablantes por medio del entrenamiento *MAP*, en el sistema *ivector-PLDA* se usa este archivo para listar los archivos que se usaran para entrenar la matriz *TV* el modelo *PLDA* además de listar los archivos de los



modelos y de prueba para extraerle los *ivectors*

Figura 7. Estructura archivo *IdMap Sidekit*

La organización de este tipo de archivo (figura 7) es de 4 columnas. La columna *leftids* lleva el nombre de los modelos y a la columna *rightids* se asigna el nombre del archivo. En

estas listas se permiten identificadores duplicados. En las últimas columnas va asignado un valor numérico para identificar qué segmento del archivo será utilizado, en este caso tiene un valor de -1 en ambas columnas pues esto significa que se hará uso del archivo completo, las 4 columnas deben tener el mismo tamaño.

Otro tipo de archivo manejado en esta herramienta son los *Ndx* (figura 7). Estos archivos son necesarios para realizar la comparación en ambos sistemas. Internamente se compone de dos columnas y una matriz, la primera columna es *modelset*, en la cual se asignan los nombres de los modelos inscritos en el sistema contrariamente a lo archivos *IdMap*. En este formato el identificador debe ser único al igual que en la columna *segset* que corresponde al audio de prueba.

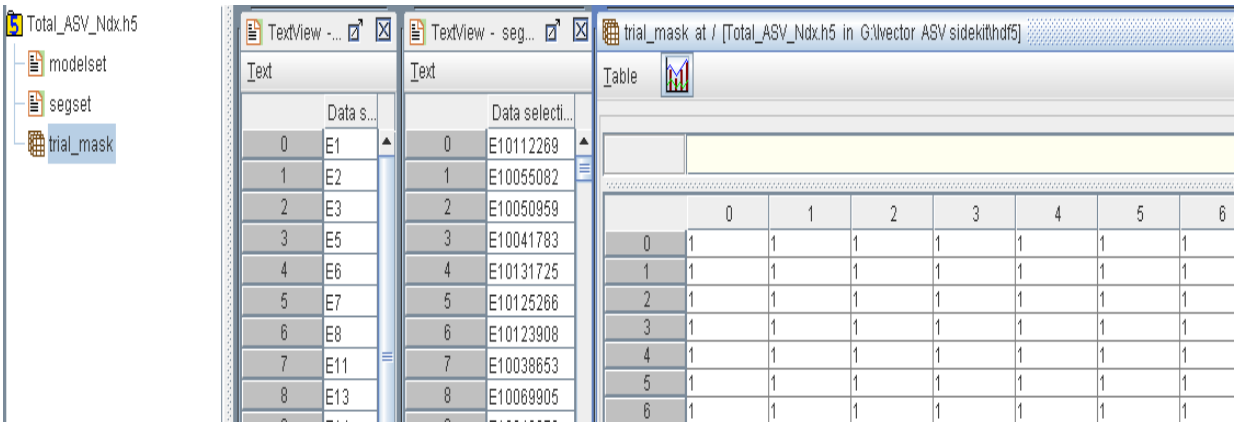


Figura 8. Estructura archivo *Ndx Sidekit*

El archivo *trial_mask* es una matriz donde se asignan 1's o 0's que son de tipo booleanos para definir qué combinaciones se llevaran a cabo en las puntuaciones y sus dimensiones son dependientes de las columnas descritas anteriormente. Donde el tamaño de *modelset* define el número de filas y *segset* define el número de columnas. Un ejemplo de cómo el sistema usa esta matriz sería el siguiente, si en la $trial\_mask_{1,1}$ apareciera un 1, esto significa que se realizara la comparación entre el primer modelo en *modelset* y el primer archivo del *segset*

Otro archivo importante son los archivos *Key* los cuales tienen la misma estructura que los *Nd.*, Sin embargo, la diferencia está en la matriz donde se asignan las posibles combinaciones donde anteriormente era para identificar si se realizaba o no una comparación mientras acá se manejan tres posibles valores 1's, -1's o 0's. Ahora bien, al igual que en los *Ndx*, la posición de estos valores relaciona el modelo y el archivo de prueba donde un 1 significa que el audio pertenece al modelo, en cambio el -1 indica que el audio es de un impostor y por último el 0 solo ignora esa combinación.

Los archivos de tipo *Score* junto con los *Key* son usados por el sistema para realizar la visualización de resultados, este archivo duplica la información del *Ndx* añadiendo una matriz *Scoremat* la cual contiene los resultados de las pruebas realizadas.

3.2.2 Spear.

Esta herramienta está escrita en una mezcla de *Python* y C++ distribuida bajo la licencia *GPLv3*[*], está construida sobre Bob, la cual es una librería de procesamiento de señales y entrenamiento de máquinas. Al igual que *sidekit* posee herramientas para implementar el procesamiento del audio tales como normalización de características; eliminación de sonidos basado en la energía y la modulación de la energía en los 4 [Hz]. Estás técnicas son usadas sobre el archivo de audio directamente siendo esto un preprocesamiento para luego proceder a la extracción de características. Así mismo posee varias técnicas de modelamientos incluidas en el estado del arte como *GMM, JFA, ISV* e *ivectors*. También contiene herramientas para la visualización de

[*] Esta licencia asegura que el software y modificaciones hechas utilizando software bajo esa licencia seguirá siendo libre

resultados y medidas de evaluación como *EER*, curva *DET*, mínimo *DCF* entre otras (Khoury, El Shafey, & Marcel, 2014, May).

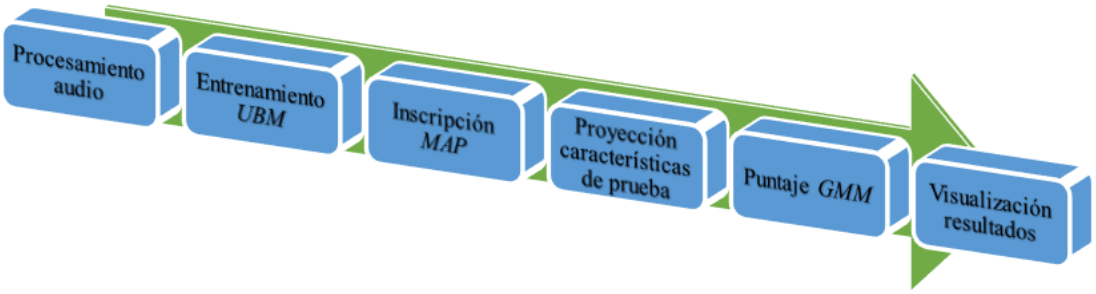


Figura 9. Sistema *GMM-UBM Spear*

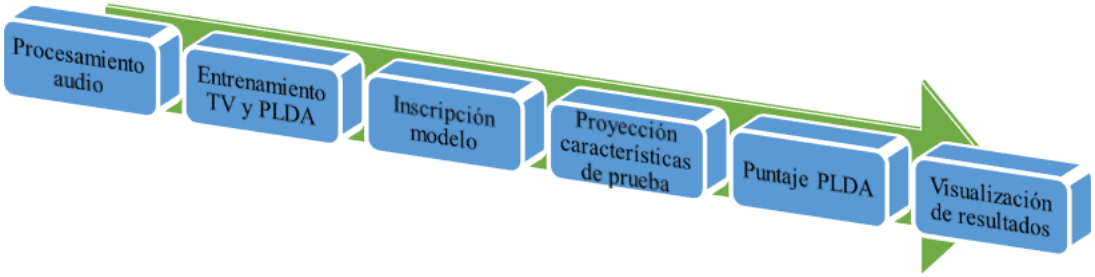


Figura 10. Sistema *ivector-PLDA Spear*

A diferencia de las otras herramientas que maneja un formato de listas para cargar los datos, esta revisa la forma como se le ingresan. Es decir, en el sistema *GMM-UBM (*figura 8) el paso de entrenamiento del *UBM* y la inscripción, al igual que en el sistema *ivector-PLDA* en el paso de entrenamiento, las características deben ingresarse en forma de lista de matrices. Estas matrices deben ser de 2 dimensiones y sus valores de punto flotante de 64 bits.

En ambos sistemas, durante el paso de la proyección de características y en el sistema GMM-UBM durante el paso de puntaje, los datos deben ser matrices de la forma como se acaban de describir. Adicionalmente en el sistema *ivector* durante el paso de inscripción, los datos deben ingresar en forma de lista de vectores de punto flotante de 64 bits, y en el paso de puntajes esa misma estructura de vectores debe conservarse.

3.2.3 Alize.

ALIZE es una herramienta escrita en C++ distribuida bajo licencia *LPGL* ampliamente usada, fue iniciada en el 2004 por la universidad de Aviñón. La librería *ALIZE* es un motor estadístico basado en el conocido modelamiento de mezclas gaussianas. Además, incluye una librería de alto nivel llamada *LIA_RAL* que contiene herramientas tales como *JFA*, *SVM* y *PLDA*. Todo esto es implementado por ejecutables (Larcher, et al., 2013, August). Por otra parte, esta herramienta, a diferencia de *SIDEKIT* o *SPEAR*, no contiene métodos para la extracción de características, aunque es capaz de leer características del tipo *HTK* y *SPro* las cuales se pueden normalizar mediante *CMVN* o *feature mapping*. Además, permite eliminar silencios por medio de la energía usando dos técnicas *meanSAD* y *weightSAD* (Bonomo Laynez, 2012). Tampoco cuenta con herramientas para la visualización de resultados o cálculo de otras medidas de desempeño anteriormente mencionadas.

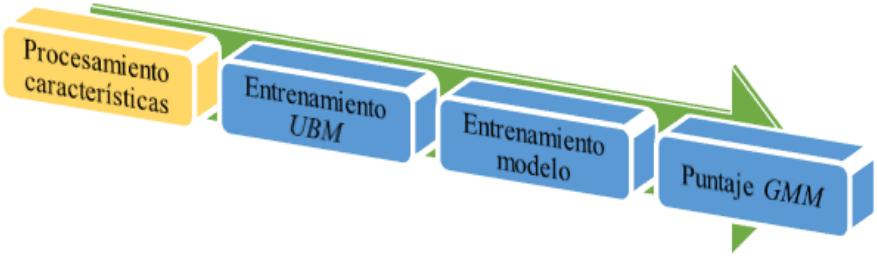


Figura 11. Sistema *GMM-UBM Alize*

En la implementación del sistema en la figura 11, se necesitan los ejecutables *NormFeat* y *EnergyDetector* para la etapa de procesamiento de características, *TrainWorld* en la etapa de entrenamiento del *UBM*. Por consiguiente, se debe realizar la respectiva compilación de las librerías para disponer de la adecuada configuración de estos y otros ejecutables disponibles

en la herramienta. Estos 3 ejecutables manejan una lista para la operación de los datos donde se coloca archivo por línea.

```
E1 E1_EN10001 E1_EN10002 E1_EN10003 E1_EN10004 E1_EN10005
E2 E2_EN10001 E2_EN10002 E2_EN10003 E2_EN10004 E2_EN10005
E3 E3_EN10001 E3_EN10002 E3_EN10003 E3_EN10004 E3_EN10005
E5 E5_EN10001 E5_EN10002 E5_EN10003 E5_EN10004 E5_EN10005
E6 E6_EN10001 E6_EN10002 E6_EN10003 E6_EN10004 E6_EN10005
E7 E7_EN10001 E7_EN10002 E7_EN10003 E7_EN10004 E7_EN10005
E8 E8_EN10001 E8_EN10002 E8_EN10003 E8_EN10004 E8_EN10005
```

Figura 12. Estructura *Ndx Alize*

TrainTarget se utiliza para inscribir los modelos de los hablantes y *ComputeTest* en la etapa de calcular puntajes. Estos ejecutables por otra parte manejan archivos *Ndx* figura 12. Donde el primer elemento de cada línea es el modelo para inscribir, y los restantes son los archivos de audio requeridos para dicho modelo para el caso de *TrainTarget*. En el caso de *ComputeTest* el primer elemento es el nombre del audio de prueba y los restantes son los modelos con los cuales se quiere comparar dicho audio.

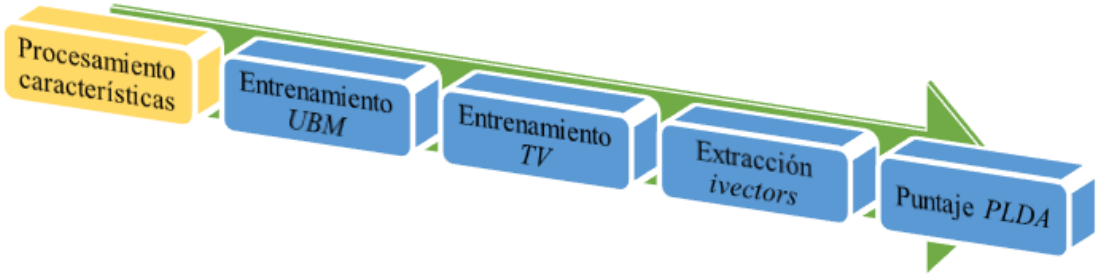


Figura 13. Sistema *ivector-PLDA Alize*

En el sistema de la figura 13 las dos primeras etapas se mantienen constantes, el ejecutable *TotalVariability* sirve para entrenar la matriz *TV*. Este ejecutable recibe la misma lista que se usa para entrenar el *UBM*, pero cambiando la extensión (.lst por .ndx), *IvExtractor* es el encargado de extraer los *ivectors*, para esto las listas deben ser *Ndx*. El mismo archivo usado en la adaptación *MAP* para inscribir los modelos es usado también en esta etapa, para el caso de los audios de prueba la estructura es de solo dos columnas donde la primera columna es el nombre

como se guardará el archivo *ivector* y la segunda columna es el nombre del archivo al cual se le extraerá el *ivector*.

Para realizar la comparación de *ivectors* se usa *IvTest*, el cual puede realizar normalización de *ivectors* en la marcha del proceso sin necesidad de guardar en el disco, también puede realizar el entrenamiento del modelo *PLDA* para lo cual se necesita un archivo *Ndx* donde cada línea del archivo es una clase y en cada línea van los audios correspondientes a dicha clase.

3.2.4 MSR Identity.

Es una caja de herramientas o *toolbox* escrita para *MATLAB* el cual maneja su propio lenguaje de programación y distribuida bajo la licencia *MSR-LA*[*]. Esta caja dispone de herramientas para la normalización de características por medio de *CMVN* y una variante de esta con una ventana deslizante, entrenamientos de *UBM; TV; PLDA* e inscripción de hablantes por medio de *MAP*, gráfico de curva *DET*, cálculo del *EER* y mínimo de *DCF*.

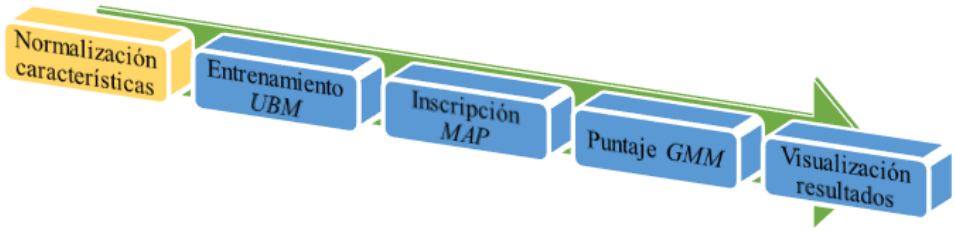


Figura 14. Sistema *GMM-UBM MSR Identity*

[*] Permite el uso, modificación y trabajo derivado para cualquier propósito no comercial, pero no permite la distribución del software ni de ningún trabajo derivado

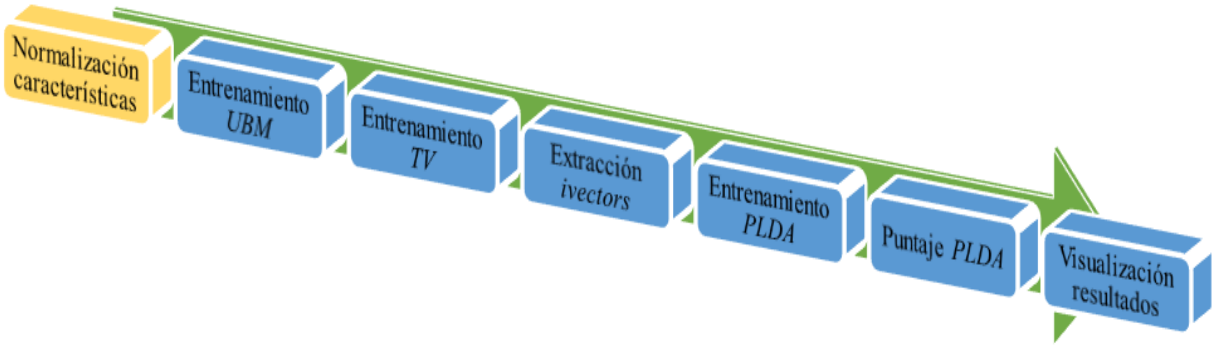


Figura 15. Sistema *ivector-PLDA MSR Identity*

Además de esto provee 2 tipos de tutoriales, uno llamado demo artificial en el cual los datos generados para los sistemas son creados, y en el otro son los sistemas base para *GMM-UBM* e *ivector-PLDA*. Para realizar el proceso en la figura 14, el programa necesita de 3 archivos en formas de listas uno para entrenar el *UBM*, inscribir los hablantes y realizar el puntaje como se observa en a), b) y c) de la figura 16 respectivamente.

```
T2_1000001.htk      E1 E1_EN10001      E1 E10040441 target      T2_1000001.htk  T2
T2_1000002.htk      E1 E1_EN10002      E1 E10030199 target      T2_1000002.htk  T2
T2_1000003.htk      E1 E1_EN10003      E1 E10079432 target      T2_1000003.htk  T2
T2_1000004.htk      E1 E1_EN10004      E1 EAF102001 impostor    T2_1000004.htk  T2
T2_1000005.htk      E1 E1_EN10005      E1 EAF102002 impostor    T2_1000005.htk  T2
```

a)ubm.lst b)speaker_model_maps.lst c)trials.lst d)ubm_with_inds.lst

Figura 16. Listas sistema *ivector-PLDA MSR Identity*

La estructura de las listas es la siguiente. En la segunda lista hay 2 columnas en la cual la primera hace referencia al modelo del hablante, y la segunda al archivo de características correspondiente para realizar la inscripción. En la tercera lista existen 3 columnas, la primera contiene el nombre del modelo que se va a comparar con el audio de prueba que está en la segunda columna, y en la última columna se especifica si el audio de prueba corresponde a dicho modelo de manera obligatoria con la palabra *target* o con 1, de manera contraria si el audio no corresponde al modelo se debe etiquetar con una palabra diferente a *target* o con un 0.

El sistema *ivector-PLDA* en la figura 15 hace uso de la primera lista tanto para entrenar el *UBM* como la matriz *TV*. Además, necesita una lista adicional (figura 16 d), la estructura de esta lista es la misma de la primera, adicionalmente tiene una segunda columna en la cual se especifica el modelo del audio de entrenamiento.

4 Implementación del Sistema

Ya que se busca la comparación de las herramientas de *software* nombrados en el capítulo anterior, se busca configurarlos de tal manera que tengas las mismas condiciones de partida, es decir, han de trabajar con los mismos datos y características de representación de entrada del sistema. Las pruebas fueron realizadas en un computador de marca *DELL* con procesador *Intel® Core™* i7-6700, sistema operativo Linux de 64 *bits, UBUNTU 16.04 LTS y* 7.7 [GiB] de memoria *RAM*.

4.1 Configuración del Experimento

Al momento de realizar pruebas se necesitan unos parámetros definidos, los cuales en todas las herramientas utilizadas deben ser las mismas. Para estimar la configuración de las herramientas de *software* abierto se comparan configuraciones en otros trabajos consultados ver tabla 3.

Se realiza una tabla en la que se relacionan los parámetros utilizados en los mencionados experimentos (Tabla 4). A partir de la observación de la misma se opta por usar 19 coeficientes MFCC junto con el logaritmo de la energía adicionalmente sus respectivos Δ y $\Delta\Delta$ (Bonomo Laynez, 2012; Alam, Kenny, Gupta, & Stafylakis, 2016, June; Wu, et al., 2016).

Igualmente se define el número de mezclas gaussianas en 512 en el entrenamiento del modelo *UBM*. De manera más general se manejan valores entre 512-2048 componentes para el *UBM*, 400-600 para la dimensión de *TV* y un número de 10 iteraciones para sus entrenamientos (Nautsch & Darmstadt, 2014). El modelo *PLDA* es ampliamente usado y ha demostrado una mejoría sobre otros modelos (Kenny P. , 2010, June). Por otra parte, la herramienta *MSR Identity* restringe en este aspecto ya que no presenta otro u otros métodos de discriminación en *ivector*.

Al momento de definir los parámetros anteriores se tuvo en cuenta los resultados que nos arrojaron la tabla 4 que recoge información de 6 trabajos que se encuentran en la tabla 3, los cuales hacen estudios referentes a la comunidad de reconocimiento del hablante lo que permitió identificar cuáles son los parámetros más utilizados y que arrojan buenos resultados en el estudio de la verificación del hablante.

Tabla 3 *Trabajos previos consultados*

Artículos	**Referencia**
A	*Speaker Verification using i-vectors* (Nautsch & Darmstadt, 2014)
B	*Joint Speaker Verification and Anti-Spoofing in the i-Vector Space* (Sizov, Khoury, Kinnunen, Wu, & Marcel, 2015)
C	*Introducing I-Vectors for Joint Anti-spoofing and Speaker Verification* (Khoury, Kinnunen, Sizov, Wu, & Marcel, 2014)
D	*ASVspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge* (Wu, et al., 2017)
E	*Factor Analysis Methods for Joint Speaker Verification and Spoof Detection* (Dhanush, et al., 2017, March)

F	*Anti-Spoofing for Text-Independent Speaker Verification: An Initial Database, Comparison of Countermeasures, and Human Performance* (Wu, et al., 2016)

Tabla 4 *Parámetros de configuración*

Articulo	**Mezclas**	**Iteraciones**	**Dimensión TV**	**Características**
A	512-2048	1,2,5,10	400-600	39-60
B	2048	10	600	60
C	512	10	400	60
D	128,256,512	10	200 - 600	63,60,50
E	512, 1024	5,10	200 - 400	42
F	512	5, 10	100 - 400	60

Adicionalmente, como se ilustró en la tabla 1 la base de datos contiene una gran cantidad de archivos de audios alcanzando un tamaño mayor a las 20 [GB]. Por esta razón se fraccionó en un conjunto más pequeño de la siguiente manera. Para el paso de los entrenamientos se secciono en dos partes la primera contiene todos los elementos genuinos del subconjunto de entrenamiento y en la segunda contiene una mezcla de todos los genuinos junto con una parte de los 5 ataques implementados, es decir se seleccionaron aleatoriamente 30 audios por cada tipo de ataque en cada uno de los modelos dando esto un total de 3750 audios de ataques. En la parte de comparación la base de datos dispone de una cantidad mayor a 200000 audios en consecuencia se tomó una muestra de estos audios de 20 audios por tipo de ataque y por genuino de cada modelo es decir un total de 11040 audios.

4.2 Procesamiento de audio

Como se explicó anteriormente, los *MFCC* intentan imitar el sistema auditivo que tiene una mayor resolución a bajas frecuencias, además de ser robustos al ruido. Estas son razones por la que estas características son ampliamente usadas en la tarea del reconocimiento del hablante. Además, estas características vienen acompañadas de sus coeficientes Δ y $\Delta\Delta$ también conocidos como coeficientes de velocidad y aceleración. Estos coeficientes dan información de la evolución temporal de los fonemas asimismo como de la transición entre fonemas y en consecuencia tiene en información en cuanto a la variabilidad del hablante a la hora de hablar (Bonomo Laynez, 2012).

Adicionalmente, teniendo en cuenta que *MSR Identity* solo tiene función para leer archivos restringida al formato *HTK*, se decidió usar este formato para el almacenamiento de las características. Por otro lado, *SIDEKIT* maneja los datos en formato *HDF5*, pero tiene la capacidad de escribir las características en el formato seleccionado y también permite realizar el procesamiento del audio. De esta manera todos los sistemas trabajarán con las mismas características.

5 Resultados

Los resultados obtenidos con las herramientas se mostrarán en la sección de apéndices *Sidekit* (Apéndice A), *Alize* (Apéndice B), *Spear* (Apéndice C) y *MSR Identity* (Apéndice D). En donde se referirá al primer conjunto de entrenamiento como humanos en la parte del título de las gráficas, por otro lado, se referirá a la segunda parte como todos. También, en las gráficas se observan unas etiquetas nombradas como humano impostor, y desde *S1* hasta *S10* las cuales hacen referencia a los diferentes tipos de ataques encontradas en la base de datos explicados brevemente en el capítulo anterior.

En las gráficas además de evidenciar el *EER*, también añade la información del mínimo *DCF* para los valores dados en las evaluaciones del *NIST* de los años 2008 y 2010 junto con sus respectivos puntos en las gráficas si están dentro de los rangos de sus ejes, a diferencia de *MSR Identity* que no posee esta característica de graficar dichos puntos simplemente devuelve los valores.

Como se explicó en el capítulo anterior la herramienta *Alize* no contiene herramienta para medir el desempeño del sistema o visualizar los resultados esta tarea se realizó con ayuda de la librería *bob.measure* la cual hace parte de la librería base de *Spear*.

5.1 Clasificación de las herramientas cualitativamente

En la siguiente tabla se clasificarán las herramientas según un punto de vista más subjetivo teniendo en cuenta por ejemplo la cantidad de información disponible, configuración, su instalación o el lenguaje de programación para hacer modificaciones dentro del programa.

La forma de clasificación se dará por medio de un valor comprendido entre 0 a 5, que refiere de lo más complicado hacia lo más asequible o fácil de trabajar. Además, la última columna (Procesa *.wav*) que se refiere a cuál de las herramientas es capaz de procesar los archivos de audio en formato *.wav* sin necesidad de una herramienta externa, será clasificada con un '0' para indicar la carencia de esta etapa o '5' para el caso contrario. En la última columna de la tabla 5 queda registrado el promedio de cada fila, el cual esta denotado con 'T'

Tabla 5 *Comparación Cualitativa de las Herramientas*

Herramienta	**Información**	**Instalación**	**Configuración**	**Programación**	**Procesa .wav**	**T**
SIDEKIT	4	4	5	5	5	4,6
MSR IDENTITY	4	5	5	5	0	3,8
ALIZE	3	3	4	2	0	2,4
SPEAR	3	3	5	3	5	3,8

En la parte de instalación de las herramientas *MSR Identity* no presenta mayor complicación pues solo hay que agregar la dirección donde se encuentren los *scripts* para que *MATLAB* pueda identificarlos. Por otra parte, *Sidekit* se puede instalar de dos formas, la primera es de la forma recomendada en su página que requiere un paquete (Anaconda o Miniconda) el cual instala paquetes de *Python* en su propio entorno con su propio interprete de *Python*. La segunda forma es realizar una instalación en el sistema el cual se realiza con el comando *pip* que conlleva a problemas de compatibilidad entre paquetes pues *Sidekit* hace uso de otras librerías como por ejemplo *matplotlib*. *matplotlib* es la encarga de manejar los gráficos de los resultados, de la

misma manera sucede con *Spear* con la diferencia que su programa base contiene no solo librerías para procesamiento de audio sino también de imágenes, por esto se debe tener en cuenta que librerías instalar. Para la instalación de *Alize* se debe realizar la previa compilación y disponer de ciertas herramientas instalas con antelación las cuales están especificadas en su página.

5.2 Porcentaje EER

En la tabla 6 se recopilan los datos obtenidos de las gráficas de los sistemas *GMM-UBM* para las 4 herramientas donde se aprecia como los diferentes tipos de ataque degradan la eficiencia de los mismas. También se puede apreciar que la herramienta *MSR Identity* presenta en promedio un *EER* bajo en comparación con las demás herramientas. Asimismo, en la tabla 7 se observa el mismo comportamiento anteriormente descrito y se puede apreciar una mejora en todos los sistemas al incluir en el entrenamiento información de ciertos ataques, la diferencia entre estas tablas es el tipo de datos usados en la etapa de entrenamientos del *UBM* donde la primera parte se usaron únicamente audios de humanos y en la segunda parte se añadieron audios con los ataques *S1* al *S5* mencionados en el capítulo acerca de la base de datos.

Tabla 6 *Promedio EER del sistema GMM-UBM conjunto de entrenamiento Humanos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	16,7	9,78	18,7	10,3	13,8
S1	30,2	18,2	28	20,2	24,1
S10	43,1	40	43,9	40,5	41,8
S2	19,3	10,2	19,9	11,6	15,2

S3	42,8	40,3	49,7	41,9	43,6
S4	45,6	44,6	49,7	45,4	46,3
S5	38,7	32,7	44,2	35,5	37,7
S6	41,3	33,2	45,5	36,4	39,1
S7	28,6	22,4	35,4	21,9	27,07
S8	34,7	31	44,4	32,2	35,5
S9	32,8	26,7	39,5	28	31,7
Total general	33,9	28,09	38,08	29,4	32,4

Tabla 7 *Promedio EER del sistema GMM-UBM conjunto de entrenamiento Todos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	17,4	10,4	15,5	11,4	13,6
S1	24,7	13,8	20,6	16,5	18,9
S10	42,7	40,3	42,8	38,7	41,1
S2	17,2	9,13	15,6	10,7	13,1
S3	36,2	31,6	45,9	33,8	36,8
S4	38	33,9	48,9	37,8	39,6
S5	34	27,2	37,7	29,6	32,1
S6	35,7	28,7	40,3	30,3	33,7
S7	26,6	18,9	27,3	19	22,9
S8	30,8	26,1	36,8	27,8	30,3
S9	28,7	20,3	31,6	21,7	25,5
Total general	30,1	23,6	33	25,2	28,01

El otro sistema de interés es el *ivector* para el cual los valores de los *EER* con el conjunto de entrenamiento mezclado se encuentran en la tabla 8 y tabla 9 en ellas se aprecia que el *Spear* muestra un resultado bajo dentro de las cuatro herramientas.

Tabla 8 *Promedio EER del sistema ivector-PLDA conjunto de entrenamiento Humanos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	44,8	22,9	17,5	14,6	24,9
S1	43,5	39,3	34,3	36,5	38,4
S10	47,8	48,5	45,9	46,1	47
S2	45,5	26,8	24,2	18,1	28,6
S3	46,6	52,8	43,3	40	45,6
S4	48,1	57	45,5	40,5	47,7
S5	48,7	46,1	40,7	38,4	43,4
S6	48	45,1	42,8	37,1	43,2
S7	46,3	35,7	33,3	32,6	36,9
S8	47,1	40,9	39	37,3	41
S9	47	41,4	39	36,2	40,9
Total general	46,6	41,5	36,8	34,3	39,8

Tabla 9 *Promedio EER del sistema ivector-PLDA conjunto de entrenamiento Todos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	45,5	22,2	18	14,9	25,1
S1	43,5	43,6	37,8	44,5	42,3

S10	45	49,2	46,4	47,6	47
S2	45,2	26,7	23,4	18,7	28,5
S3	44	54	49,1	43,5	47,6
S4	44,5	57,7	49,6	46,3	49,5
S5	45,7	48,3	45,1	38,3	44,3
S6	46,6	48	46,3	35,9	44,2
S7	44,5	38,7	32,6	31,8	36,9
S8	41,9	46,3	41,8	38	42
S9	42,7	45,5	41,2	38,9	42
Total general	44,4	43,6	39,2	36,2	40,8

5.3 Mínimo del DCF

Otro criterio de evaluación expuesto en el capítulo 2 es el mínimo *DCF* donde se piensa en una aplicación más al mundo real, pues se le da prioridad a disminuir un error de falsa aceptación. Para los dos conjuntos de entrenamientos se presentan los resultados para los valores de C_{miss}, P_{targ}, C_{FA} de 10, 0.01 y 1 respectivamente, estas se recopilan en las tablas 10 y 11. En este escenario la herramienta que muestra un mejor resultado en promedio es *Spear*.

Tabla 10 *Promedio mínimo DCF 2008 conjunto de entrenamiento Todos en sistema GMM-UBM*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	6,151	4,387	6,209	4,375	5,2805

S1	7,395	4,877	6,751	5,182	6,05125
S10	9,154	9,966	9,105	9,154	9,34475
S2	6,278	4,308	6,681	3,972	5,30975
S3	9,132	9,639	9,174	9,124	9,26725
S4	9,035	9,772	9,174	9,134	9,27875
S5	9,154	9,107	9,174	8,991	9,1065
S6	9,144	9,214	9,174	8,753	9,07125
S7	8,252	7,302	8,675	7,002	7,80775
S8	9,062	9,01	9,172	9,085	9,08225
S9	8,549	8,148	9,074	7,931	8,4255
Total general	8,3005	7,793	8,396	7,518	8,0023

Tabla 11 *Promedio mínimo DCF 2008 conjunto de entrenamiento Humanos en sistema GMM-UBM*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	5,934	3,605	7,172	4,238	5,237
S1	7,974	6,567	8,174	6,521	7,309
S10	9,134	9,967	9,164	9,095	9,34
S2	6,718	3,761	7,898	4,519	5,724
S3	9,154	9,946	9,174	9,263	9,384
S4	9,075	9,967	9,164	9,134	9,335
S5	8,984	9,55	9,174	9,154	9,215
S6	9,095	9,648	9,164	9,085	9,248

S7	9,372	7,958	9,154	7,741	8,556
S8	8,845	9,542	9,164	8,987	9,134
S9	8,714	8,919	9,055	8,433	8,780
Total general	8,454	8,13	8,768	7,833	8,296

Tabla 12 *Promedio mínimo DCF 2010 conjunto de entrenamiento Humano en sistema GMM-UBM*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	0,07511	0,0563	0,09826	0,05924	0,07222
S1	0,09348	0,097	0,9054	0,08217	0,29451
S10	0,09957	0,0997	0,09989	0,09913	0,09957
S2	0,08272	0,0788	0,09315	0,0712	0,08146
S3	0,09978	0,0995	0,1	0,02085	0,08003
S4	0,09891	0,0997	0,09989	0,09957	0,09951
S5	0,09793	0,0988	0,1	0,09978	0,09912
S6	0,09913	0,0988	0,09989	0,09902	0,09921
S7	0,09663	0,0888	0,09978	0,09457	0,09494
S8	0,09641	0,0955	0,09989	0,09848	0,09757
S9	0,09652	0,0912	0,0987	0,09859	0,09625
Total general	0,0941	0,0912	0,172	0,0838	0,1104

Tabla 13 *Promedio mínimo DCF 2010 conjunto de entrenamiento Todos en sistema GMM-UBM*

	Alize	MSR Identity	Sidekit	Spear	Total general
Humano	0,08391	0,0677	0,08989	0,07011	0,07790
S1	0,08228	0,0677	0,09239	0,07217	0,07863
S10	0,09978	0,0998	0,09924	0,09978	0,09965
S2	0,0937	0,0797	0,08076	0,07207	0,08155
S3	0,09967	0,0986	0,1	0,09946	0,09943
S4	0,09848	0,0977	0,1	0,09957	0,09893
S5	0,09978	0,0986	0,1	0,2076	0,12649
S6	0,09967	0,0972	0,1	0,09957	0,09911
S7	0,09489	0,0937	0,09598	0,08728	0,09296
S8	0,09967	0,091	0,1	0,09902	0,09742
S9	0,095	0,0865	0,09935	0,08978	0,09265
Total general	0,0951	0,0889	0,0961	0,0996	0,0949

Otros valores propuestos por el *NIST* para el mínimo *DCF* son C_{miss}, P_{targ}, C_{FA} de 1, 0.01 y 1 respectivamente, los resultados de esta medida se encuentran en las tablas 12 y 13. Donde la herramienta *Spear* muestra mejor resultado cuando es entrenado solamente con humanos y por otro lado la herramienta *MSR Identity* al entrenarse tanto con información de humanos como con ataques presenta el valor más bajo de las cuatro herramientas.

Tabla 14 *Promedio mínimo DCF 2008 conjunto de entrenamiento Todos en sistema ivector-PLDA*

	Alize	MSR Identity	Sidekit	Spear	Total general
Humano	9,095	9,137	7,833	7,109	8,293
S1	9,154	9,935	8,925	9,124	9,284
S10	9,154	10	9,055	9,164	9,343
S2	9,114	9,778	8,493	7,689	8,768
S3	9,164	9,989	9,134	9,095	9,345
S4	9,362	9,989	9,174	9,154	9,419
S5	9,164	9,978	9,095	9,124	9,340
S6	9,144	10	9,174	9,134	9,363
S7	9,144	9,705	8,995	8,975	9,204
S8	9,263	10	9,174	9,114	9,387
S9	9,154	10	9,173	9,164	9,372
Total general	9,173	9,864	8,929	8,804	9,193

Tabla 15 *Promedio mínimo DCF 2008 conjunto de entrenamiento Humanos en sistema ivector-PLDA*

	Alize	MSR Identity	Sidekit	Spear	Total general
Humano	9,164	9,935	7,979	7,408	8,621
S1	9,164	9,902	8,895	9,134	9,273
S10	9,164	9,935	9,164	9,164	9,356

S2	9,243	9,376	8,765	7,914	8,824
S3	9,114	9,935	9,085	9,095	9,307
S4	9,114	9,967	9,095	9,263	9,359
S5	9,154	10	9,095	9,114	9,340
S6	9,529	10	9,085	9,213	9,456
S7	9,164	9,767	8,984	8,885	9,2
S8	9,362	9,967	8,995	9,085	9,352
S9	9,461	9,967	9,044	9,144	9,404
Total general	9,239	9,886	8,926	8,856	9,227

Tabla 16 *Promedio mínimo DCF 2010 conjunto de entrenamiento Todos en sistema ivector-PLDA*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	0,09913	0,0953	0,1	0,08859	0,0957
S1	0,09978	0,0993	0,09728	0,09946	0,0989
S10	0,09978	0,1	0,0987	0,09989	0,0995
S2	0,09935	0,0992	0,09522	0,09326	0,0967
S3	0,09989	0,0999	0,09957	0,09913	0,0996
S4	0,3171	0,0999	0,1	0,09978	0,1541
S5	0,09989	0,0998	0,09913	0,09946	0,0995

S6	0,09967	0,1	0,1	0,09957	0,0998
S7	0,09967	0,0976	0,09804	0,09783	0,0982
S8	0,2085	0,1	0,1	0,09935	0,1269
S9	0,09978	0,1	0,1	0,09989	0,0999
Total general	0,1293	0,0991	0,0989	0,0978	0,1063

Tabla 17 *Promedio mínimo DCF 2010 conjunto de entrenamiento Humanos en sistema ivector-PLDA*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Humano	0,09989	0,0974	0,1	0,09489	0,09804
S1	0,09989	0,099	0,09696	0,09957	0,09885
S10	0,09989	0,0993	0,09989	0,09989	0,09974
S2	0,2083	0,0997	0,09967	0,09685	0,12613
S3	0,09935	0,0993	0,09902	0,09913	0,0992
S4	0,09935	0,0997	0,09913	0,2085	0,12667
S5	0,09978	0,1	0,09913	0,09935	0,09956
S6	0,05339	0,1	0,09902	0,2079	0,11507
S7	0,09989	0,0993	0,09913	0,09685	0,09879
S8	0,3171	0,0997	0,09804	0,09902	0,15346
S9	0,4257	0,0997	0,0988	0,09967	0,18096
Total general	0,1547	0,0993	0,09898	0,11832	0,11786

5.4 Tiempos de ejecución

En la siguiente tabla se realizó una cuantificación con respecto al tiempo de ejecución, para el cual se tomaron los tiempos de los pasos más relevantes que mostraban un mayor impacto al momento de ejecución de las pruebas.

Tabla 18 *Tiempos de pasos con el conjunto de entrenamiento Todos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Enroll	00:00:09	00:00:11	00:00:07	00:07:41	00:08:08
Extracción iv	00:49:35	00:21:10	01:54:23	01:57:46	05:02:54
PLDA	00:06:52	00:00:29	00:00:02	00:00:06	00:07:29
Puntajes GMM	00:20:53	00:04:46	00:15:52	00:08:28	00:49:59
Puntajes iv	00:00:06	00:00:06	00:00:04	00:01:27	00:01:43
TV	04:55:00	01:03:04	08:13:49	01:31:51	15:43:44
UBM	00:56:27	00:06:56	00:37:23	01:00:41	02:41:27
Total general	07:09:02	01:36:42	11:01:40	04:48:00	24:35:24

Al momento de ejecutar el proceso de *Enroll* se puede apreciar que la herramienta más eficiente en este paso es *Sidekit*, y la que muestra el tiempo más alto es la herramienta *Spear* al mostrar un tiempo de 7 minutos.

Para el siguiente proceso Extracción iv para el cual la herramienta que muestra un notorio resultado es *MSR Identity*, y un resultado bastante alto se observa en la herramienta *Spear*

En el entrenamiento del modelo *PLDA* la herramienta *Sidekit* arrojó el mejor resultado, al contrario de *Alize* que realiza la tarea en minutos.

En el instante de la ejecución de puntajes *GMM* el mejor tiempo lo realiza *MSR Identity* con un tiempo de 4 minutos 46 segundos y el peor a *Alize* con un tiempo de 20 minutos 53 segundos.

A pesar de que es un proceso de muy poco tiempo la herramienta *Spear* fue la que mayor tiempo le costó realizar esta tarea y *Sidekit* la que menor tiempo gastó al momento de ejecutar el proceso de puntajes iv.

Para los procesos *UBM* y extracción de la matriz *TV* es muy notoria la eficiencia de *MSR Identity*, para *UBM* el peor resultado se lo lleva *Alize*, al igual que *Sidekit* para *TV*.

Tabla 19 *Tiempos de pasos con el conjunto de entrenamiento Humanos*

	Alize	**MSR Identity**	**Sidekit**	**Spear**	**Total general**
Enroll	00:00:04	00:00:31	00:00:07	00:01:42	00:02:24
Extracción iv	00:36:25	00:18:42	01:42:16	02:00:19	04:37:42
PLDA	00:04:36	00:00:18	00:00:01	00:00:03	00:04:58
Puntajes GMM	00:02:13	00:05:22	00:04:51	00:10:03	00:22:29
Puntajes iv	00:00:27	00:00:19	00:00:04	00:01:38	00:02:28
TV	02:26:46	00:24:48	03:41:51	00:35:30	07:08:55
UBM	00:11:39	00:03:42	00:17:27	00:33:28	01:06:16
Total general	03:22:10	00:53:42	05:46:37	03:22:43	13:25:12

Para la tabla 19 se obtuvieron resultados similares a los de la tabla 18, con la diferencia de que los tiempos totales de cada herramienta bajan a valores cercanos a la mitad pues en esta prueba fueron usados la mitad de los archivos en la parte de entrenamiento. La discrepancia más notoria

se ve en la fila de 'puntajes *GMM*', ya que en la tabla 18 el que tuvo el tiempo más bajo fue la herramienta *MSR Identity* en cambio en la tabla 19 lo obtuvo *Alize*, el cual en la tabla anterior fue el resultado más alto.

6 Discusión

En este capítulo se trae a colación el tema de 'porque el *EER* calculado por las herramientas al momento de obtener los resultados no fue el esperado', aunque para nosotros fue extraño este resultado bastante alto, mostrado en las tablas 6, 7, 8 y 9 del capítulo de Resultados en la parte de Porcentaje *EER*. Ya que las herramientas fueron calibradas previamente a las pruebas con la base de datos *ASVSpoof* 2015, realizando varias pruebas con la base de datos del *MIT* hasta obtener los resultados mostrados en la mayoría de artículos. Esta calibración se realizó con la base de datos del *MIT* ya que es mucho más pequeña que *ASVSpoof* 2015 y las pruebas tomarían menos tiempo.

Se cree que este problema en parte se debe a que no se utilizó toda la base de datos *ASVSpoof* 2015 explicado en el capítulo 4.2 (configuración del experimento), ya que el sistema tiene menos archivos en los cuales verificar y comparar y por lo tanto los resultados no son los esperados.

El crédito mayor de este problema se lo lleva lo referente al ruido, ya que es uno de los factores que es aleatorio e impredecible en todo sistema y cambia su comportamiento de forma inesperada. El ruido afecta todos los estados de los sistemas como lo deja claro (García Perea, 2014). Es un factor que afecta notoriamente y se ve reflejado en los puntajes EER

como se puede observar en la figura 17 donde la línea azul representa una condición completamente limpia y la línea roja la condición con ruido.

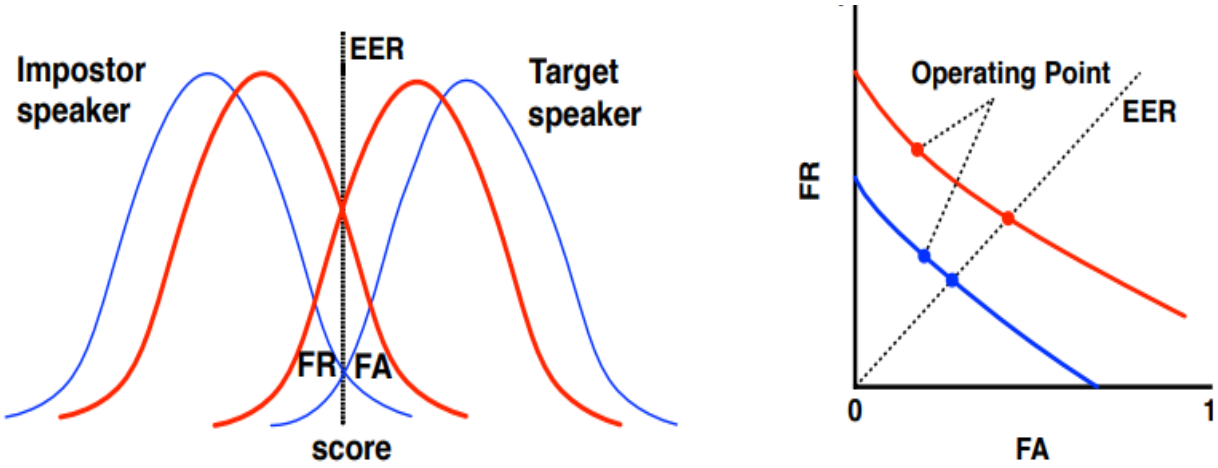


Figura 17. Efectos del ruido en los sistemas de verificación del hablante Muestra. Adaptado de (García Perea, 2014)

Finalmente, para tener una mejor aseveración de gran parte de los resultados cuantitativos se crearon las tablas 20 y 21.

Tabla 20 *Promedio total para la prueba conjunto de entrenamiento Humanos*

Herramienta	**Porcentaje Eer**		**Mínimo del DCF 2008**		**Mínimo del DCF 2010**	
	GMM	*I-vector*	*GMM*	*I-vector*	*GMM*	*I-vector*
Sidekit	38,08	36,8	8,768	8,926	0,172	0,099
MSR Identity	**28,09**	41,6	8,130	9,886	0,091	0,099
Alize	33,9	46,5	8,454	9,239	0,094	0,129
Spear	29,4	**34,3**	**7,833**	**8,856**	**0,084**	**0,098**

Tabla 21 *Promedio total para la prueba conjunto de entrenamiento Todos*

Herramienta	**Porcentaje Eer**		**Mínimo del DCF 2008**		**Mínimo del DCF 2010**	
	GMM	*I-vector*	*GMM*	*I-vector*	*GMM*	*I-vector*
Sidekit	33	39,2	8,396	8,929	0,096	0,099
MSR Identity	**23.6**	43,6	7,793	9,864	**0,089**	0,099
Alize	30,1	44,4	8,301	9,173	0,095	0,129
Spear	25,2	**36,2**	**7,518**	**8,804**	0,096	**0,098**

En la evidencia mostrada anteriormente se ve claramente que la herramienta que presenta un mejor desempeño durante la mayoría de pruebas es *Spear*, superada por *MSR Identity* en el sistema *GMM* respecto a la métrica del *EER*, aunque por muy poco, por otra parte, en el sistema *ivector* la herramienta *Sidekit* maneja un *EER* cercano al *Spear*.

7 Trabajos Futuros

Con el transcurrir de los tiempos se ha visto como los sistemas de verificación del hablante han tomado un gran auge, debido a la necesidad de seguridad. Donde cualquier avance repercute en gran escala al crecimiento tecnológico. Se invita al lector a que deje salir su curiosidad para así hacer propia la búsqueda del mejoramiento y entendimiento de estas herramientas, ya que a pesar de encontrar bastante información aún hace falta mucho por investigar acerca de los engaños y ataques que van en contra de estos sistemas de verificación del hablante, por lo que es un área sin resolver y de reciente interés. Esto fue una gran aseveración que se obtuvo con el desarrollo de este trabajo de grado debido a que no se observó y no fue fácil encontrar información sobre el tema.

- ✓ En un trabajo futuro se puede buscar un mejoramiento en los sistemas basados en i-vector de todas las herramientas. Ya que no se obtuvieron los resultados más óptimos.
- ✓ Modificar las herramientas, en busca de un mejoramiento a tal punto de conseguir que sean más robustos ante *SPOOFING*, ya que no hay herramientas que detecten estos ataques o engaños.
- ✓ Sabiendo que *spear* muestra un buen desempeño en la mayoría de pruebas en estos tipos de sistemas (*GMM* e *IVECTOR*), él ofrece la posibilidad de ser implementada en redes neuronales el cual puede utilizarse para un estudio exhaustivo en este tema.

8 Conclusiones

1. Desde un punto de vista subjetivo enfocado más a romper la barrera entre un principiante y dichas herramientas, *Sidekit* es la herramienta más idónea pues no solo permite realizar una rápida implementación de un sistema, sino que también posee unos tutoriales en su página para familiarizarse con la herramienta, además es de fácil instalación y de fácil edición en el código de ser requerido.
2. De las tablas de registro de los tiempos queda constatado que la herramienta *MSR Identity* muestra una ventaja en cuanto a la velocidad de procesamiento, el cual es superior a las otras herramientas, ya que fue la que realizo la mayoría de procesos en menor tiempo, ya que esta herramienta realiza los procesos en paralelo.
3. Un aspecto importante de estas herramientas es el tipo de licencias que poseen ya que estas limitan su uso, como *MSR Identity* cuya licencia solo la restringe a usos no comerciales.
4. Según el criterio *EER* en el sistema *GMM,* la herramienta más robusta a los ataques de la base de datos mencionados en el capítulo 3.2 es *MSR Identity*. Por otra parte, en el sistema de *ivector* la herramienta más robusta a estos ataques resulta ser *SPEAR*.

Referencias Bibliográficas

Alam, M. J., Kenny, P., Gupta, V., & Stafylakis, T. (2016, June). Spoofing detection on the ASVspoof2015 challenge corpus employing deep neural networks. *In Proc. Speaker Lang. Recognit. Workshop Odyssey*, (pp. 270-276).

Bimbot, F., Bonastre, J. F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., . . . Reynolds, D. A. (2004). A Tutorial on Text-Independent Speaker Verification. *EURASIP Journal on Advances in Signal Processing*(4), 430-451.

Bonomo Laynez, D. (2012). Sistemas de Verificación Automática del Locutor. Sevilla.

Campbell Jr, J. P. (1997). Speaker recognition: a tutorial. *Proceedings of the IEEE, 85*(9), 1437-1462.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing, 19(4)*, 788-798.

Dhanush, B. K., Suparna, S., Aarthy, R., Likhita, C., Shashank, D., Harish, H., & Ganapathy, S. (2017, March). Factor analysis methods for joint speaker verification and spoof detection. *In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference On* (pp. 5385-5389). IEEE.

Domínguez, J. G., Zazo, R., & González Rodríguez, J. (2012). On the use of Total Variability and Probabilistic Linear Discriminant Analysis for Speaker Verification on Short Utterances. In *Advances in Speech and Language Technologies for Iberian Languages* (pp. 11-19). Berlin: Springer.

Evans, N. W., Kinnunen, T., & Yamagishi, J. (2013, August). Spoofing and countermeasures for automatic speaker verification. *Interspeech*, (págs. 925-929).

García Perea, L. P. (2014). *Métodos discriminativos para la optimización de modelos en la Verificación del Hablante*. (Doctoral dissertation, Universidad de Zaragoza).

Handbook of Biometrics. (s.f.). Springer.

Kenny, P. (2010, June). Bayesian speaker verification with heavy-tailed priors. *In Odyssey*, (p. 14).

Kenny, P. (2012, June). A small footprint i-vector extractor. *Odyssey*, *2012*, págs. 1-6.

Khoury, E., El Shafey, L., & Marcel, S. (2014, May). Spear: An open source toolbox for speaker recognition based on Bob. *In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (pp. 1655-1659). IEEE.

Khoury, E., Kinnunen, T., Sizov, A., Wu, Z., & Marcel, S. (2014). Introducing i-vectors for joint anti-spoofing and speaker verification. *In Fifteenth Annual Conference of the International Speech Communication Association*.

Larcher, A., Bonastre, J. F., Fauve, B., Lee, K. A., Lévy, C., Li, H., . . . Parfait, J. Y. (2013, August). ALIZE 3.0-open source toolkit for state-of-the-art speaker recognition. *Interspeech*, (pp. 2768-2772).

Larcher, A., Lee, K. A., & Meignier, S. (2016, March). An extensible speaker identification sidekit in Python. *In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (pp. 5095-5099). IEEE.

Martin, A., Doddinggton, G., Kamm, T., Ordowski, M., & Przybocki, M. (1997). *The DET curve in assessment of detection task performance*. Gaithersburg MD: National Institute of Standards and Technology.

Matějka, P., Glembeck, O., Castaldo, F., Alam, M., Plchot, O., Kenny, P., . . . Černocky, J. (2011). Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification.

Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on (pp. 4828-4831). IEEE.

Mendoza Marín, A. F., & Porras Plata, D. (2016). Desarollo de un Sistema de Verificación del Hablante Basado en Modelos de Mezclas Gaussianas.

Nautsch, A., & Darmstadt, H. (2014). Speaker Verification using i-vector. Hochschule, Darmstadt, Germany: University of Applied Science.

Prince, S. J., & Elder, J. H. (2007, October). Probabilistic Linear Discriminant Analysis for Inferences About Identity. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (pp. 1-8). IEEE.

Prince, S., Li, P., Fu, Y., Mohammed, U., & Elder, J. (2012). Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(1), 144-157.

Rajana, P., Afanasyev, A., Hautamäki, V., & Kinnunen, T. (2014). From single to multiple enrollment i-vectors: practical PLDA scoring variants for speaker verification. *Digital Signal Processing, 31*, 93-101.

Reynolds, D. (2015). Gaussian Mixture Models. In *Encyclopedia of biometrics* (pp. 827-832). Springer.

Reynolds, D. (2015). Universal Background Models. In *Encyclopedia of biometrics* (pp. 1547-1550). Springer.

Sánchez de la Fuente, J. (2016). Utilización de la fase armónica en la detección de voz sintética.

Sigüenza, B. G. (2008). BATVOX: sistema automático de reconocimiento de locutor. *Estudios de fonética experimental, 17*, pp. 303-316.

Sizov, A., Khoury, E., Kinnunen, T., Wu, Z., & Marcel, S. (2015). Joint Speaker Verification and Anti-Spoofing in the i-Vector Space. *IEEE Transactions on Information Forensics and Security, 10(4)*, 821-832.

Wu, Z., De Leon, P. L., Demiroglu, C., Khodabakhsh, A., King, S., Ling, Z. H., . . . Yamagishi, J. (2016). Anti-spoofing for text-independent speaker verification: An initial database, comparison of countermeasures, and human performance. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 24(4)*, 768-783.

Wu, Z., Kinnunen, T., Evans, N., & Yamagishi, J. (2015). Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof 2015) Database. University of Edinburgh. The Centre for Speech Technology Research (CSTR). Retrieved from http://datashare.is.ed.ac.uk/handle/10283/853

Wu, Z., Kinnunen, T., Evans, N., Yamagishi, J., Hanilçi, C., Sahidullah, M., & Sizov, A. (2015). ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. *In Sixteenth Annual Conference of the International Speech Communication Association.*

Wu, Z., Yamagishi, J., Kinnunen, T., Hanilci, C., Sahidullah, M., Sizov, A., . . . Todisco, M. (2017). ASVspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge. *IEEE Journal of Selected Topics in Signal Processing, 11(4)*, 588-604.

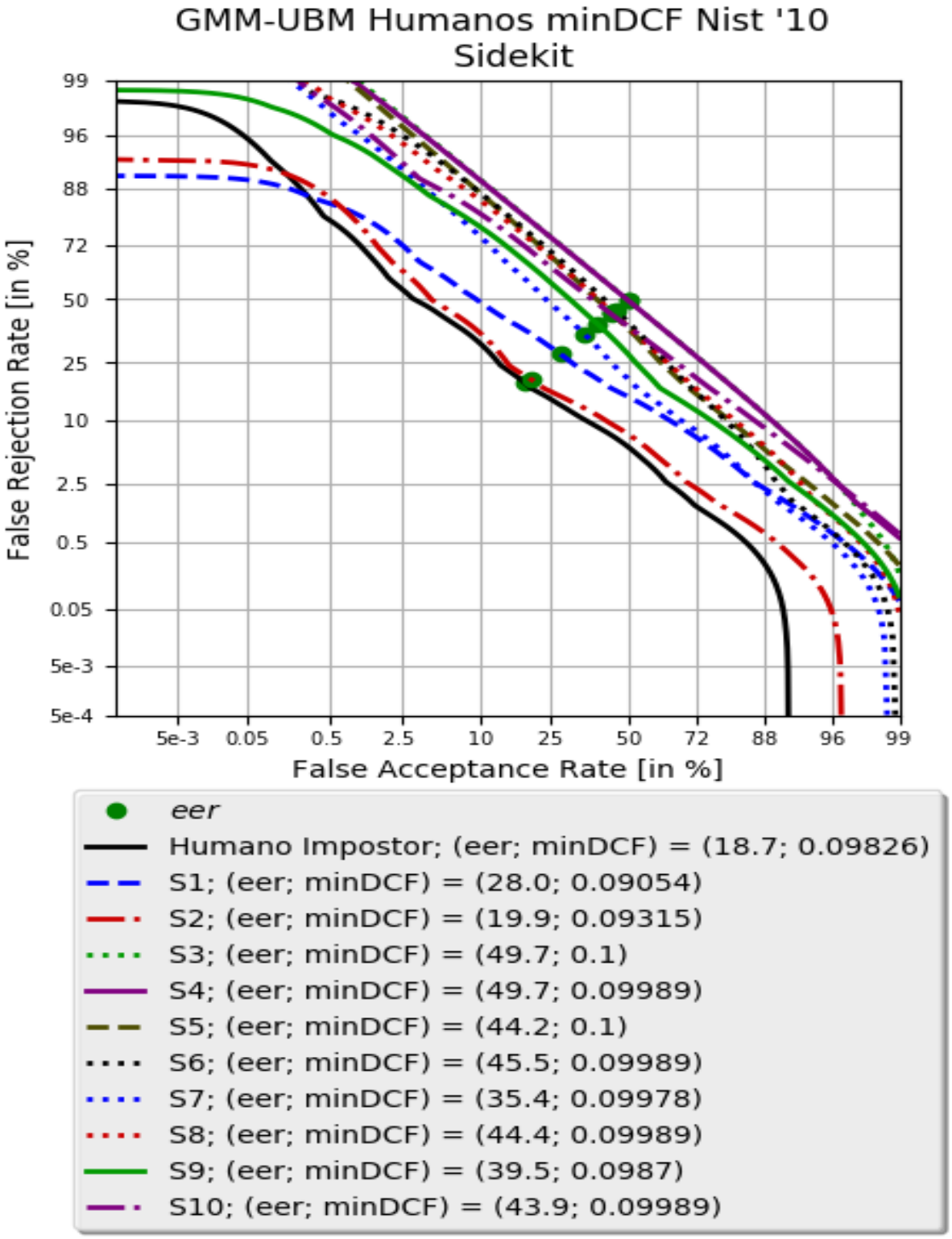
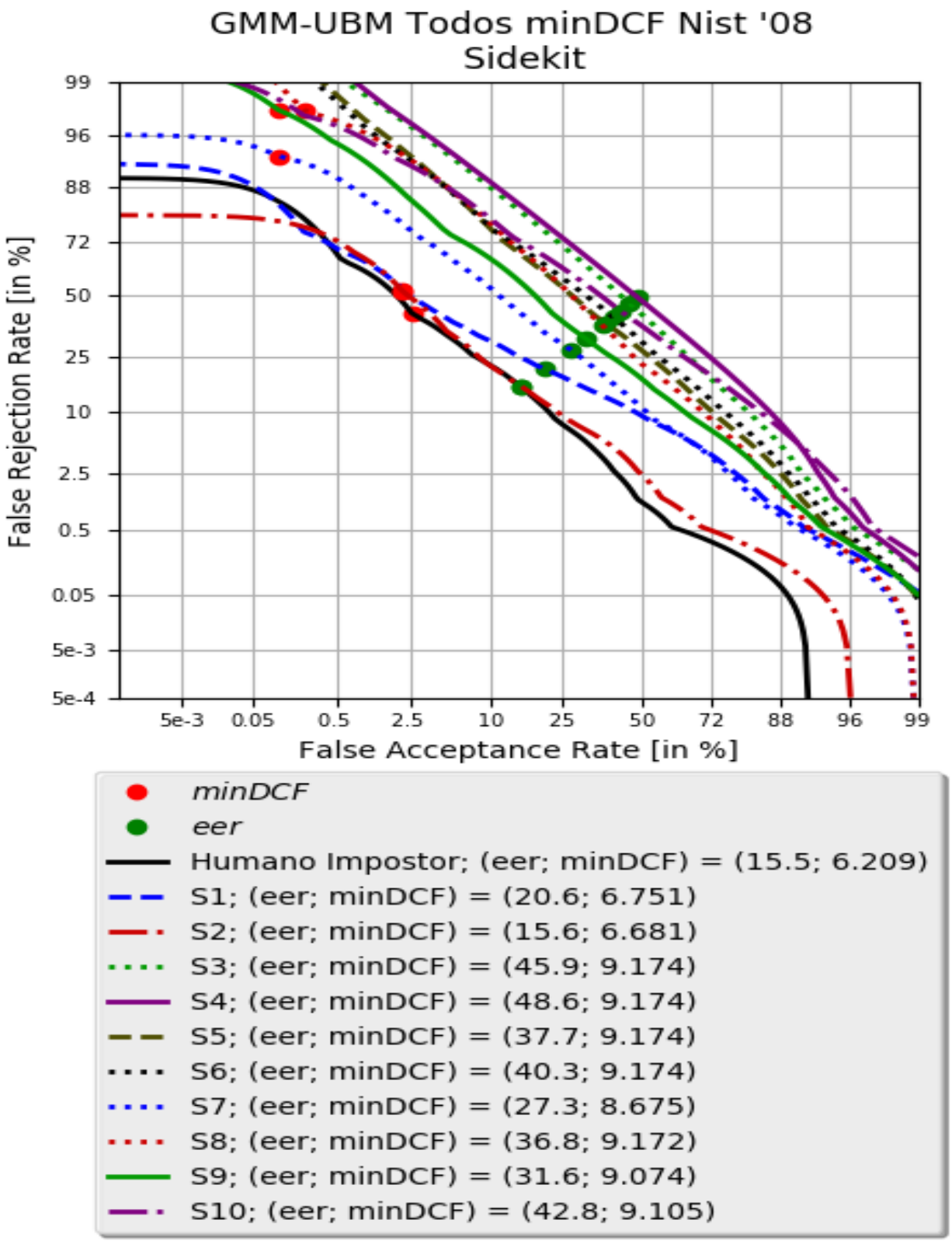


Figura 18. Sistema *GMM-UBM* Humanos *Sidekit* con valores para min *DCF* del *NIST* 2010



F*igura 19*. Sistema *GMM-UBM* Todos *Sidekit* con valores para min *DCF* del *NIST* 2008

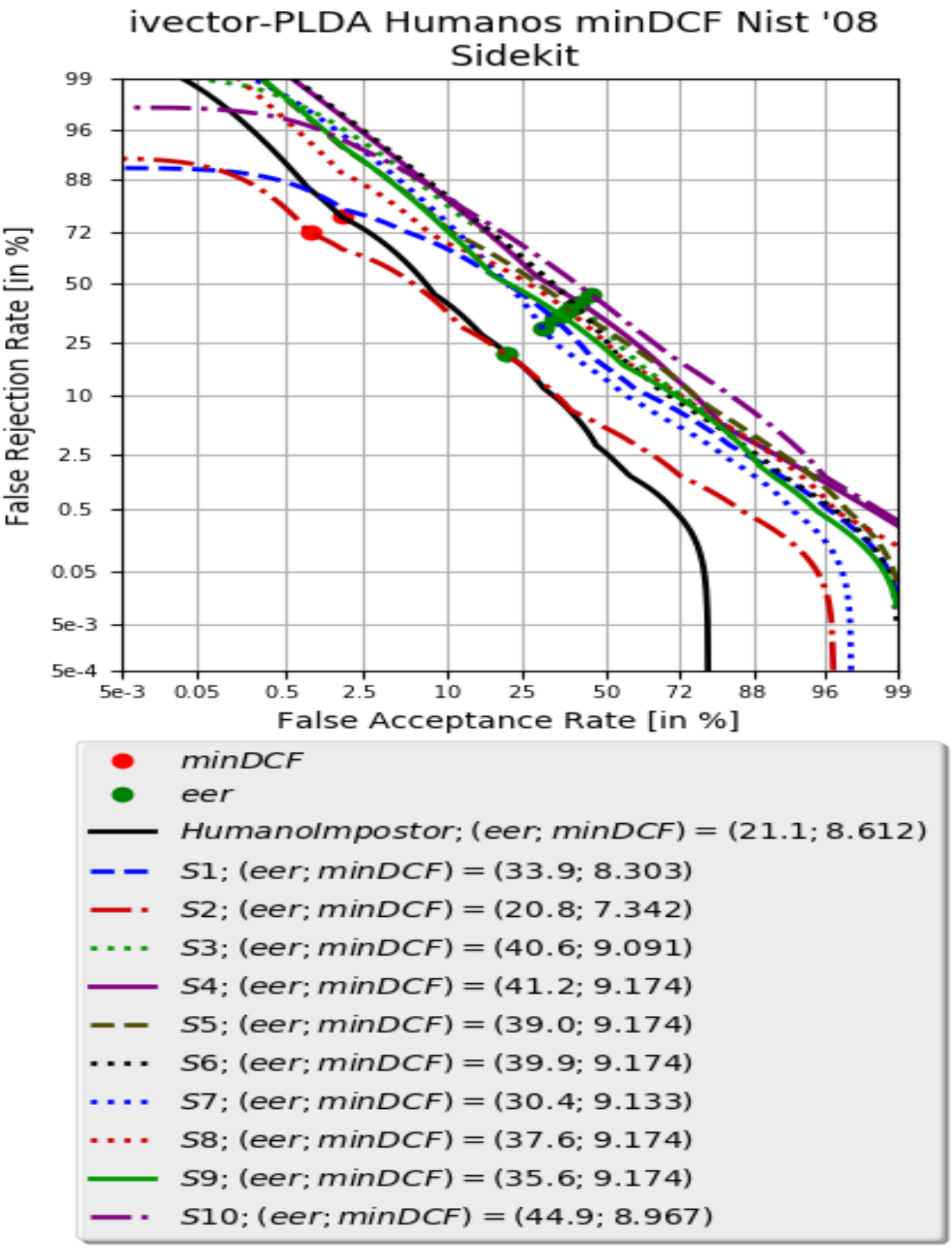


Figura 20. Sistema *ivector-PLDA* Humanos *Sidekit* con valores para min *DCF del NIST* 2008

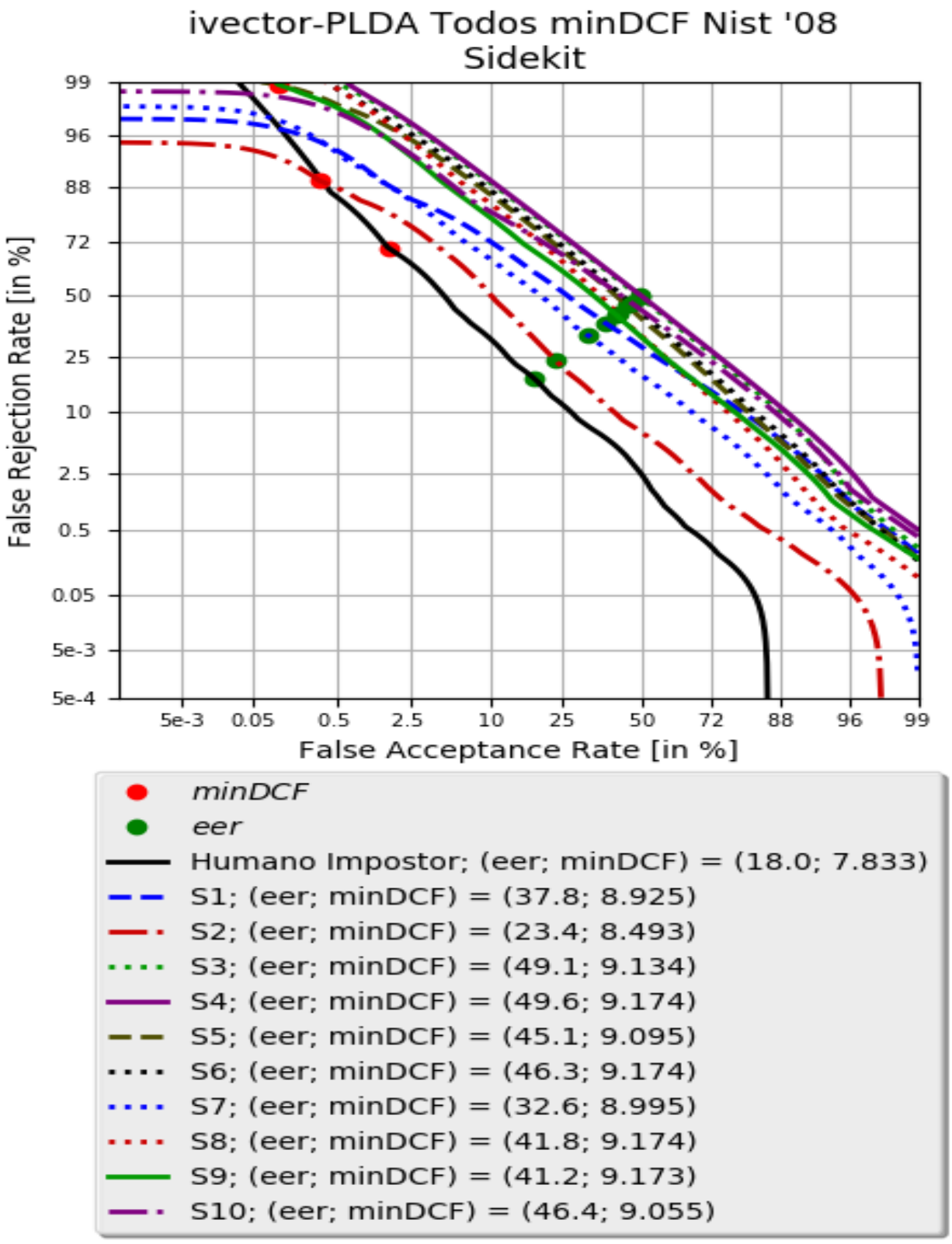
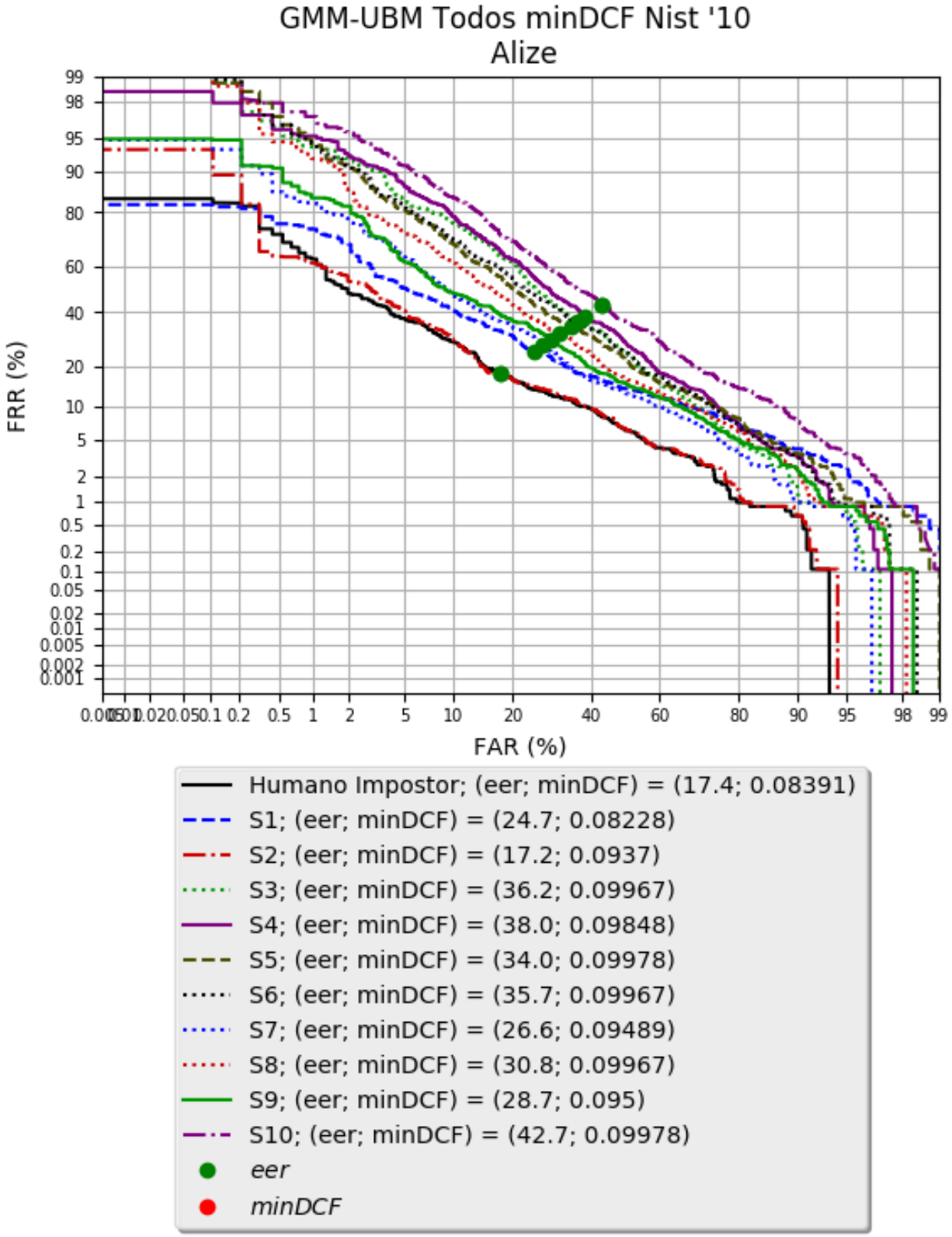


Figura 21. Sistema *ivector-PLDA* Todos *Sidekit* con valores para min *DCF* del *NIST* 2008

Apéndice B Pruebas con la herramienta ALIZE



F*igura 22*. Sistema *GMM-UBM* Todos *Alize* con valores para min *DCF* del *NIST* 2010

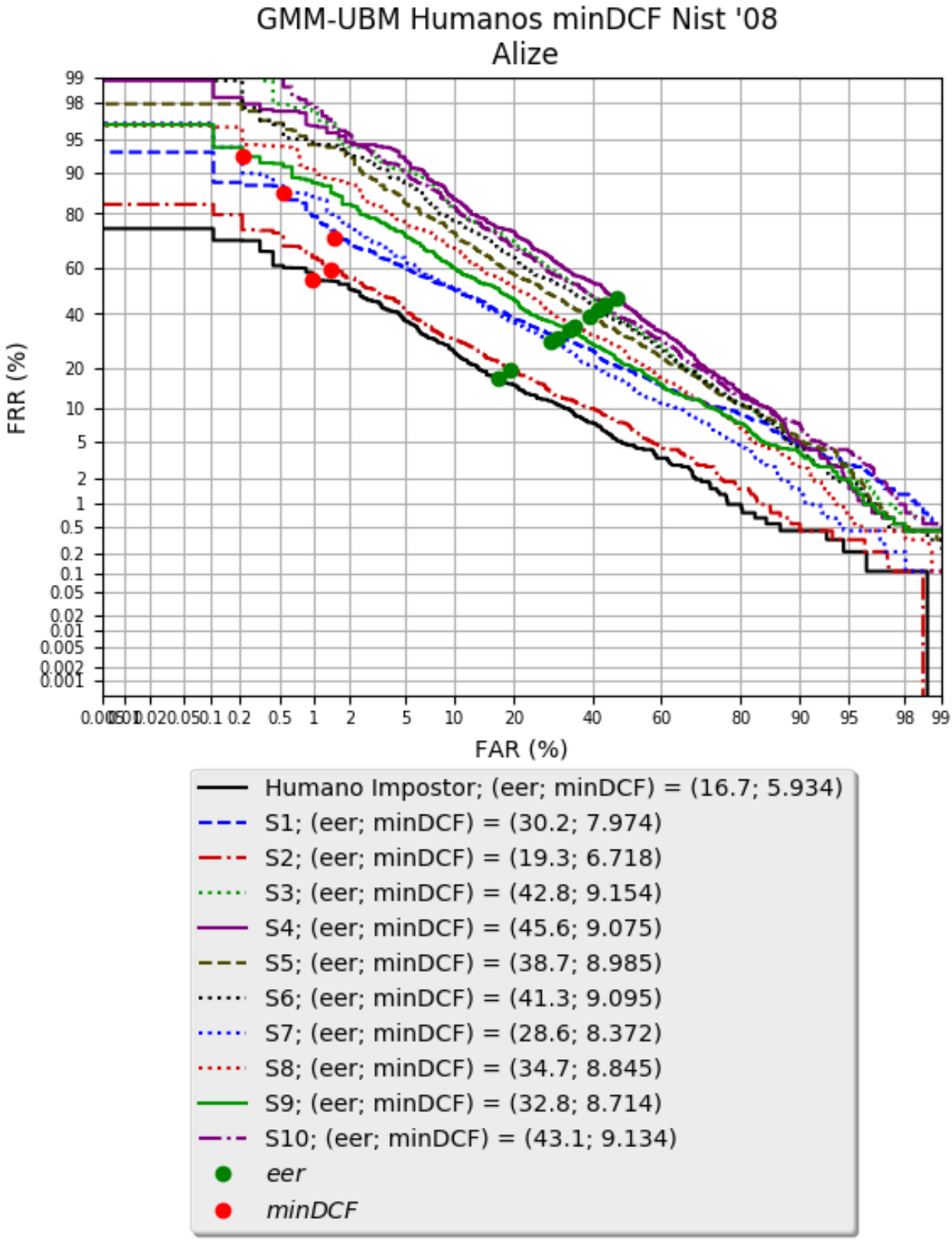


Figura 23. Sistema *GMM-UBM* Humanos *Alize* con valores para min *DCF* del *NIST* 2008

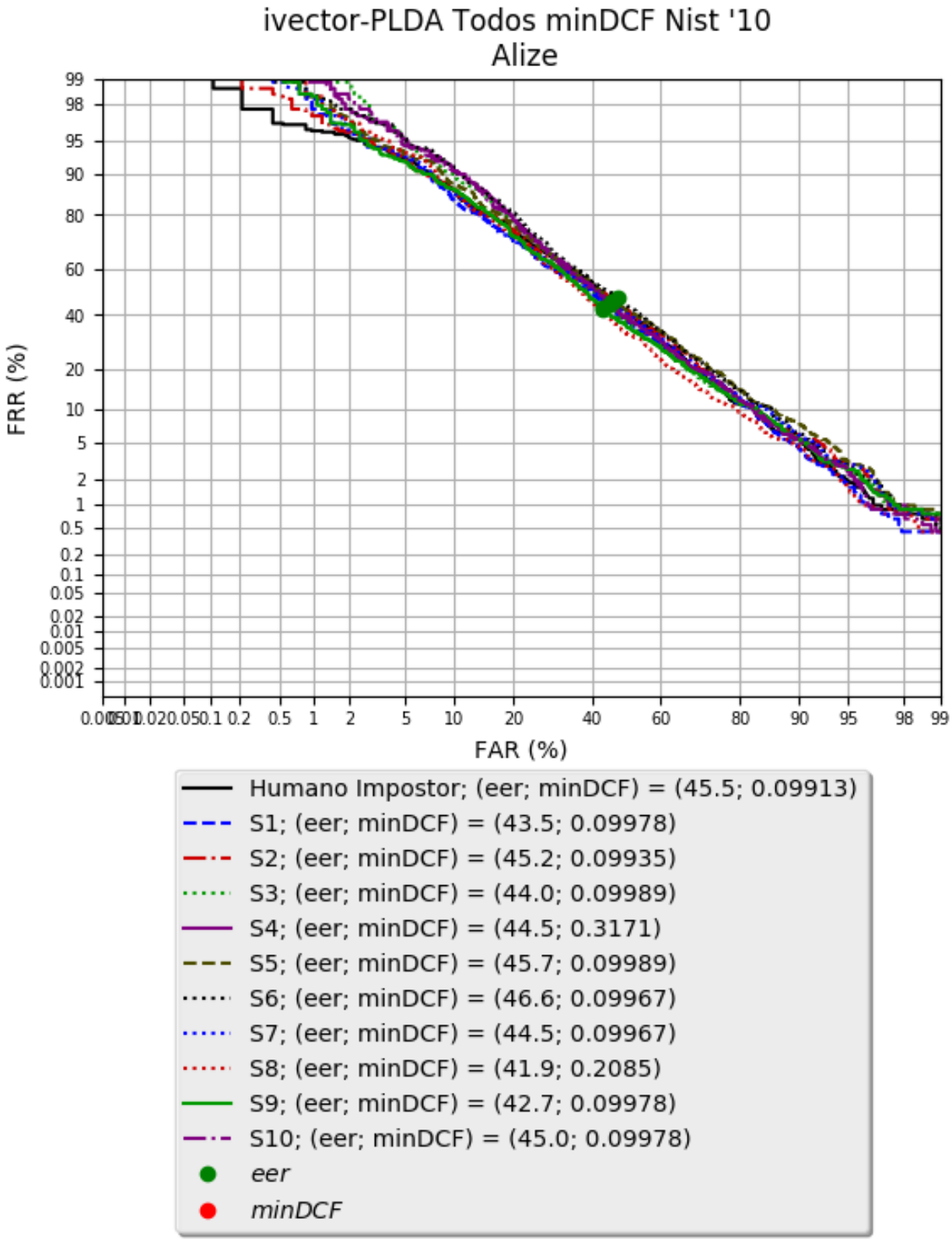


Figura 24. Sistema *ivector-PLDA* Todos *Alize* con valores para min *DCF* del *NIST* 2010

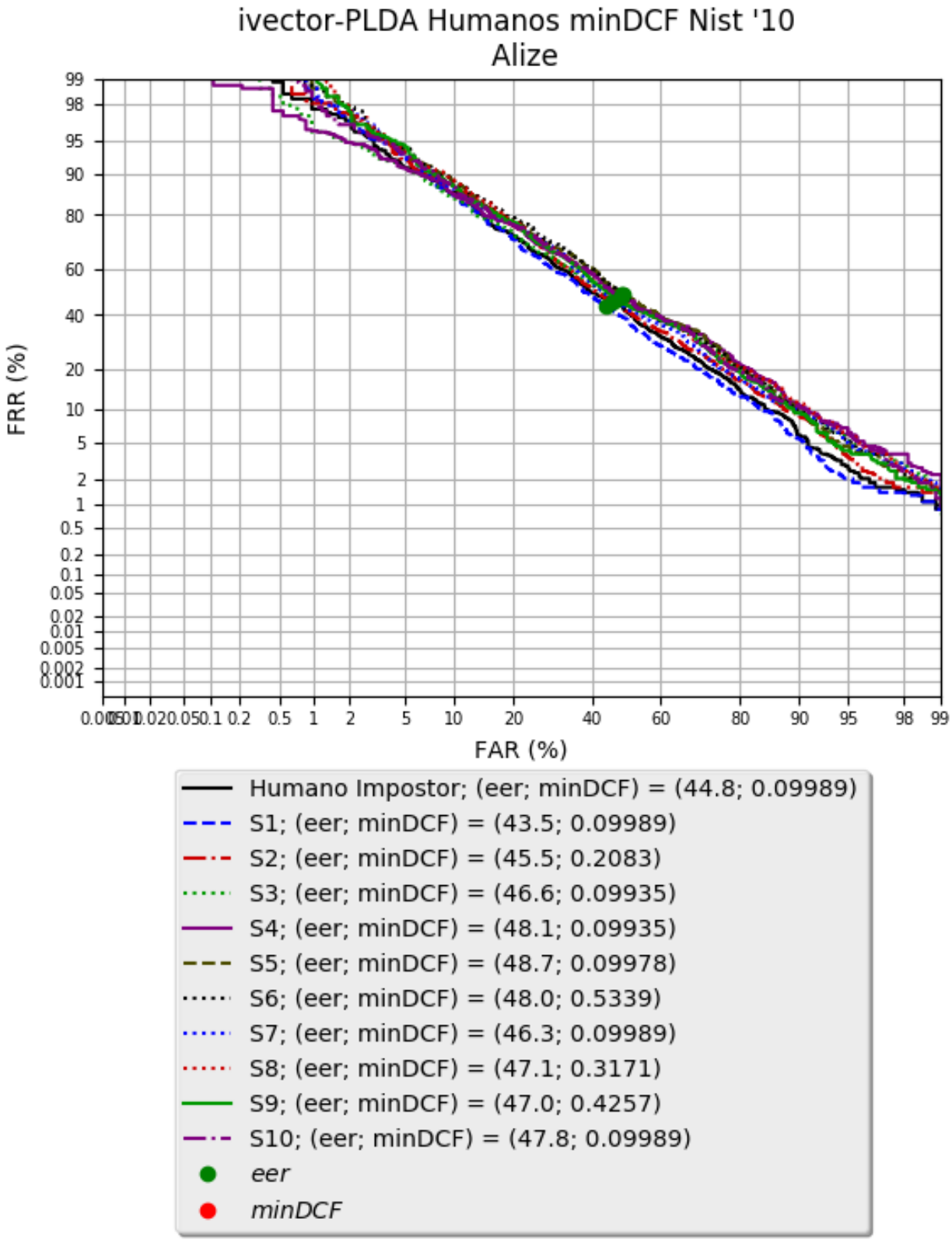


Figura 25. Sistema *ivector-PLDA* Humanos *Alize* con valores para min *DCF* del *NIST* 2010

Apéndice C Pruebas con la herramienta *SPEAR*

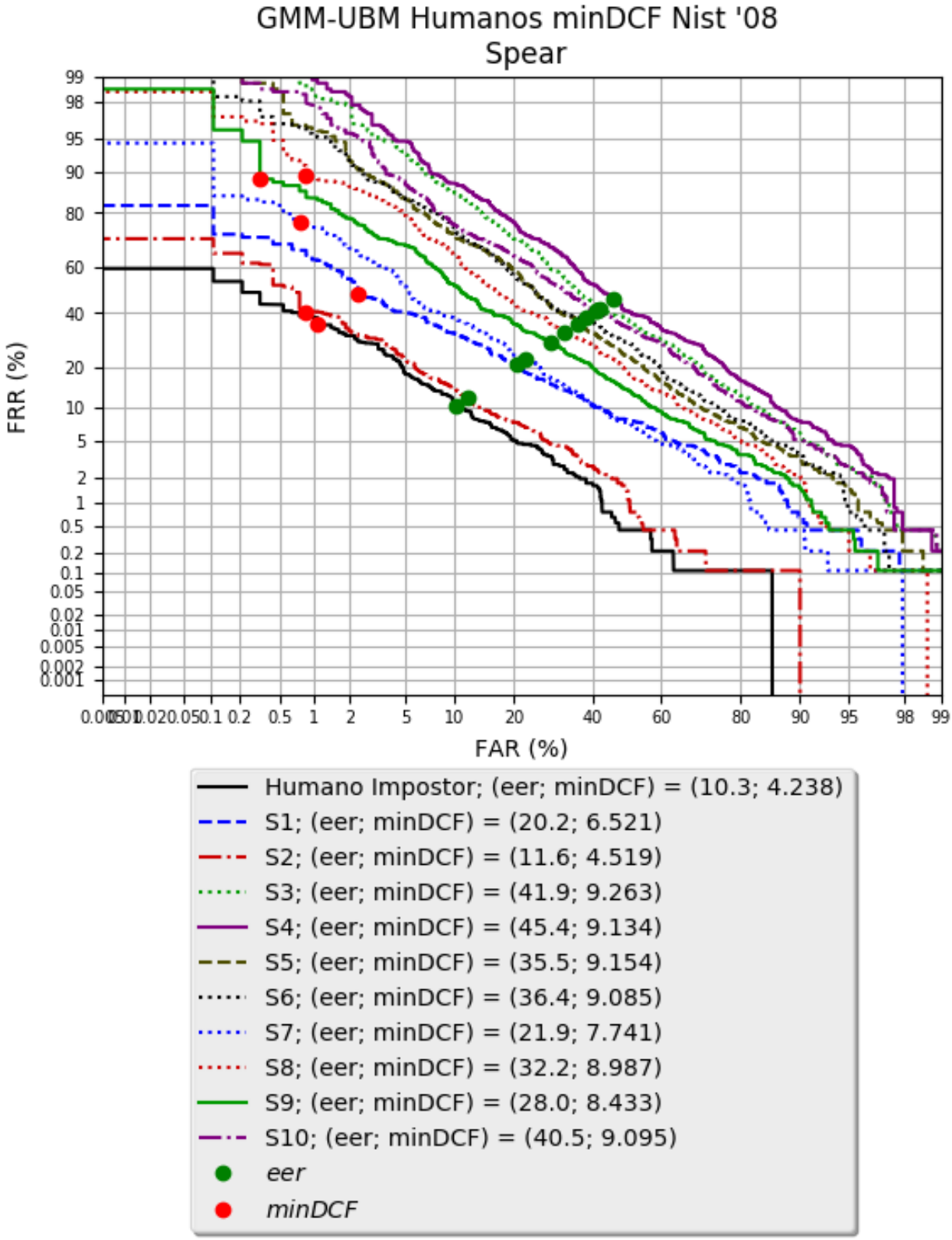


Figura 26. Sistema *GMM-UBM* Humanos *Spear* con valores para min *DCF* del *NIST* 2008

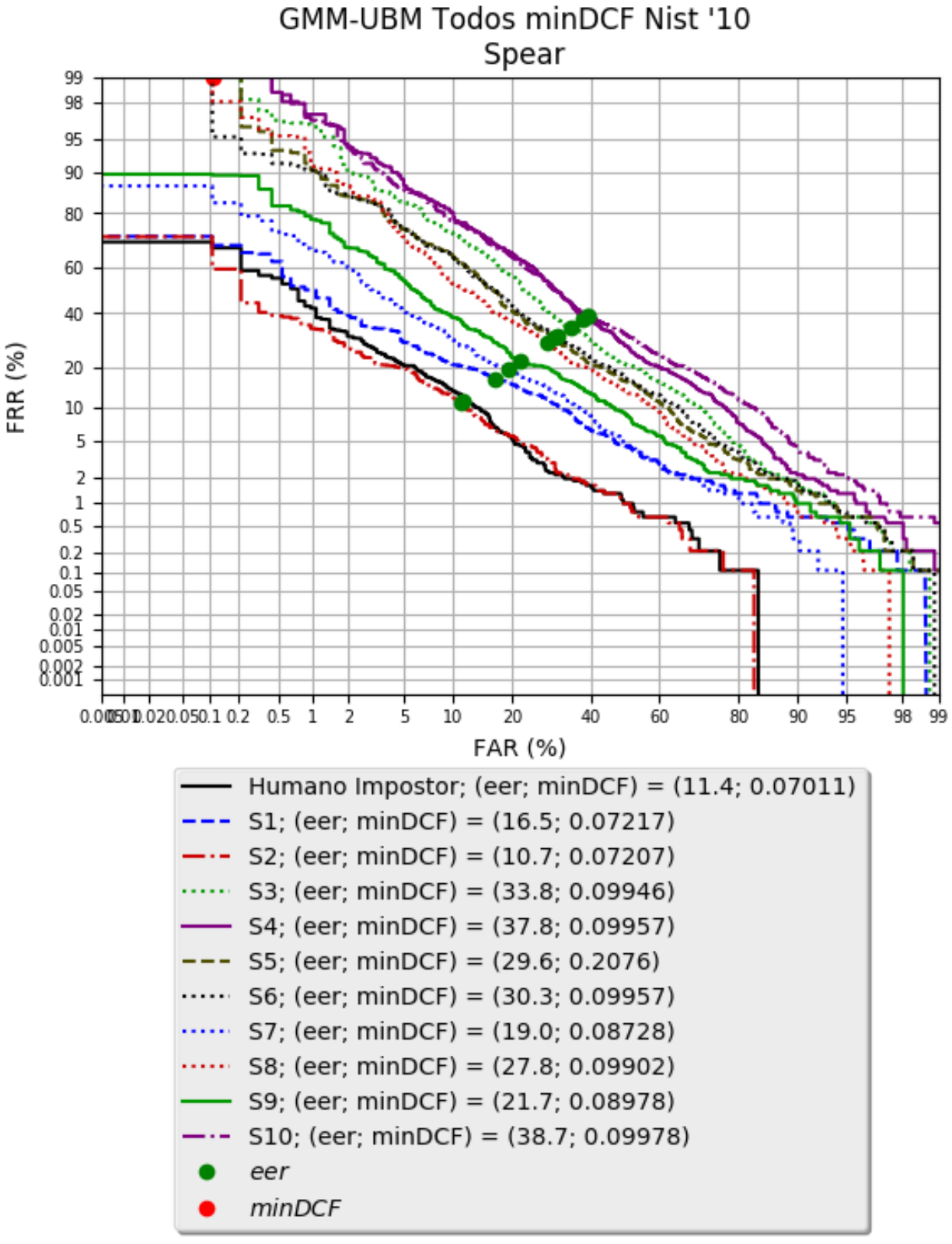
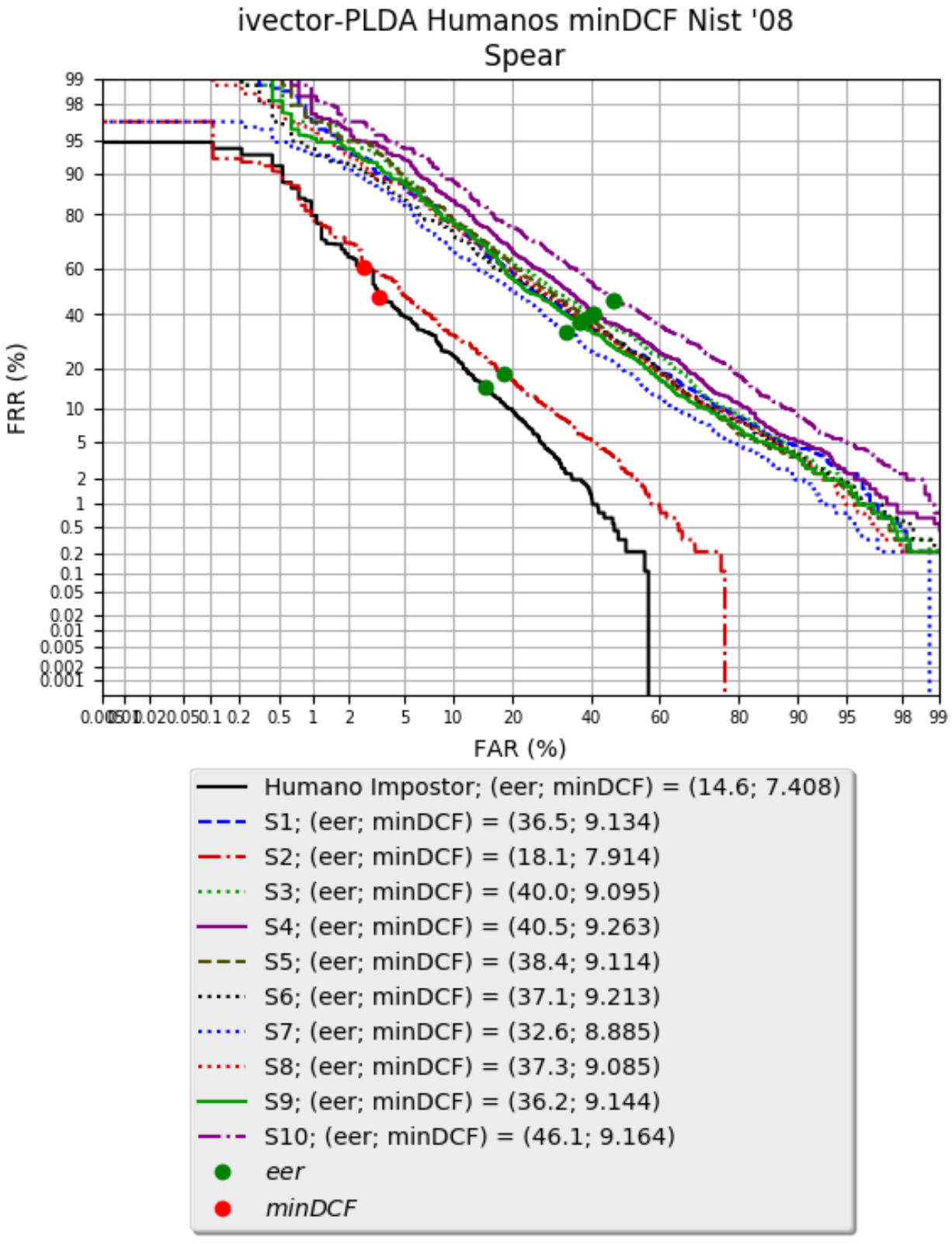


Figura 27. Sistema *GMM-UBM* Todos *Spear* con valores para min *DCF* del *NIST* 2010


ivector-PLDA Humanos minDCF Nist '08
Spear
FRR (%)
FAR (%)
99
98
95
90
80
60
40
20
10
5
2
1
0.5
0.2
0.1
0.05
0.02
0.01
0.005
0.002
0.001
0.1
0.2
0.5
1
2
5
10
20
40
60
80
90
95
98
99
Humano Impostor; (eer; minDCF) = (14.6; 7.408)
S1; (eer; minDCF) = (36.5; 9.134)
S2; (eer; minDCF) = (18.1; 7.914)
S3; (eer; minDCF) = (40.0; 9.095)
S4; (eer; minDCF) = (40.5; 9.263)
S5; (eer; minDCF) = (38.4; 9.114)
S6; (eer; minDCF) = (37.1; 9.213)
S7; (eer; minDCF) = (32.6; 8.885)
S8; (eer; minDCF) = (37.3; 9.085)
S9; (eer; minDCF) = (36.2; 9.144)
S10; (eer; minDCF) = (46.1; 9.164)
eer
minDCF

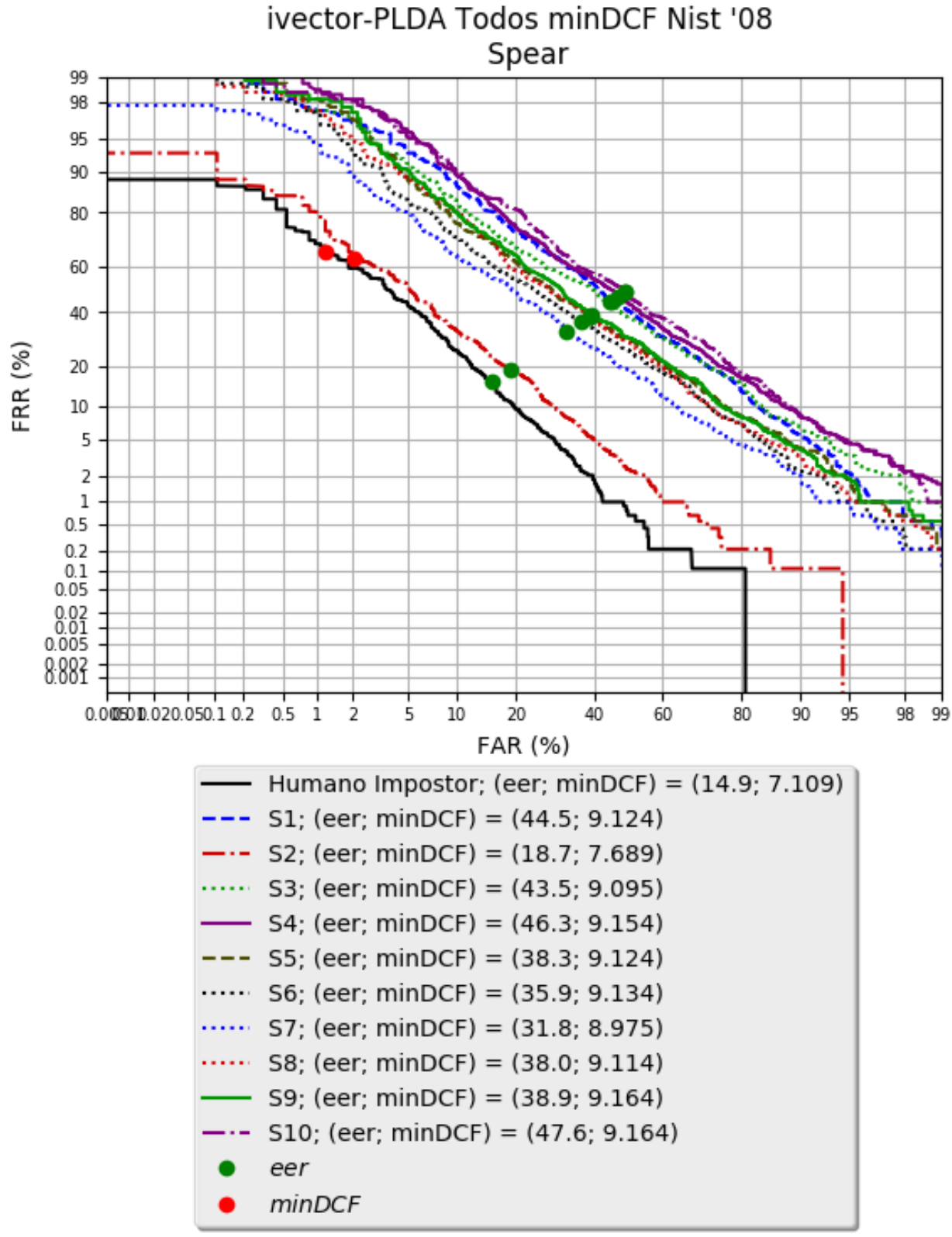


Figura 28. Sistema *ivector-PLDA* Todos *Spear* con valores para min *DCF* del *NIST* 2008

Apéndice D Pruebas con la herramienta *MSR IDENTITY TOOLBOX*

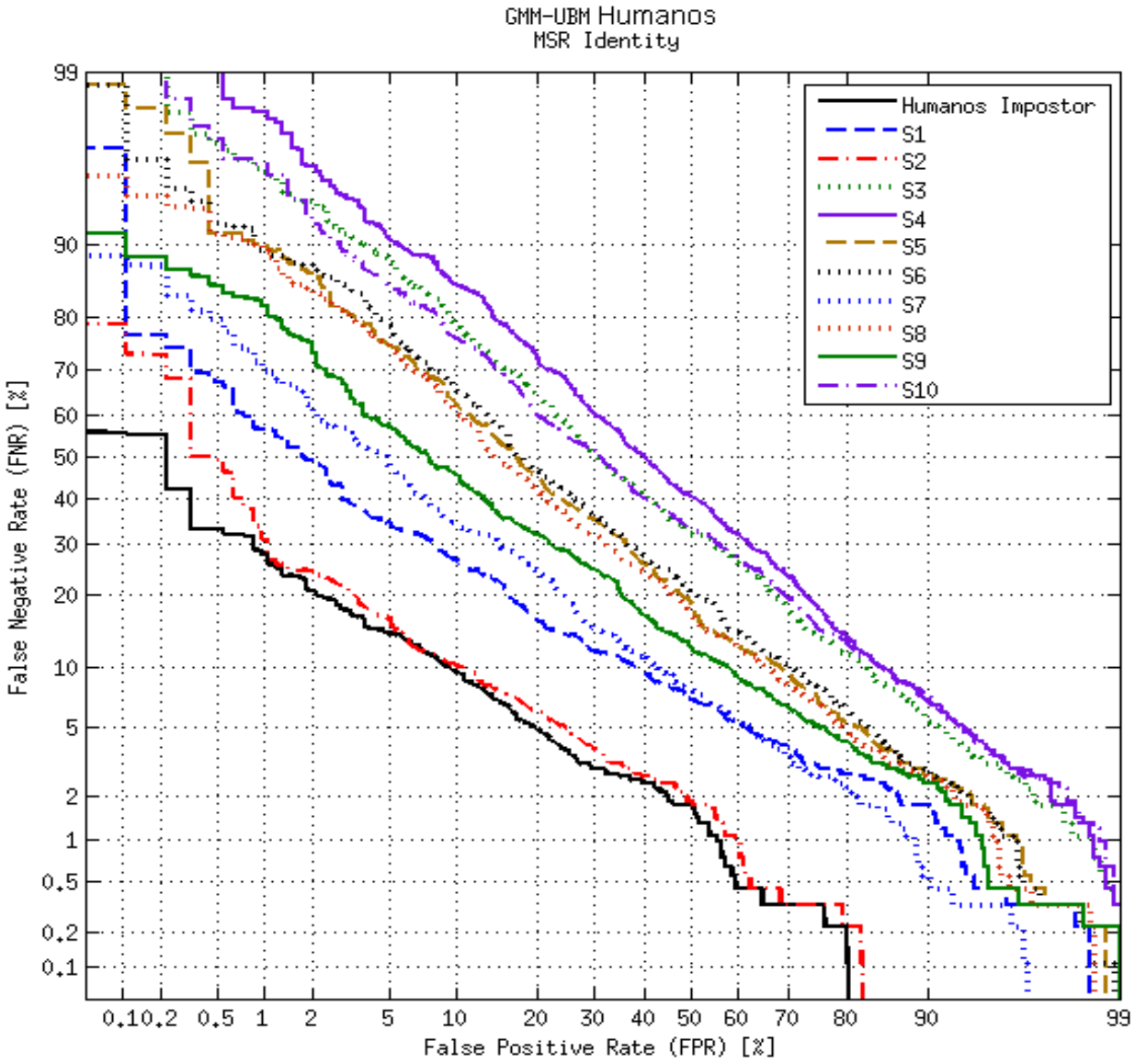


Figura 29. Sistema *GMM-UBM* Humanos *MSR Identity*

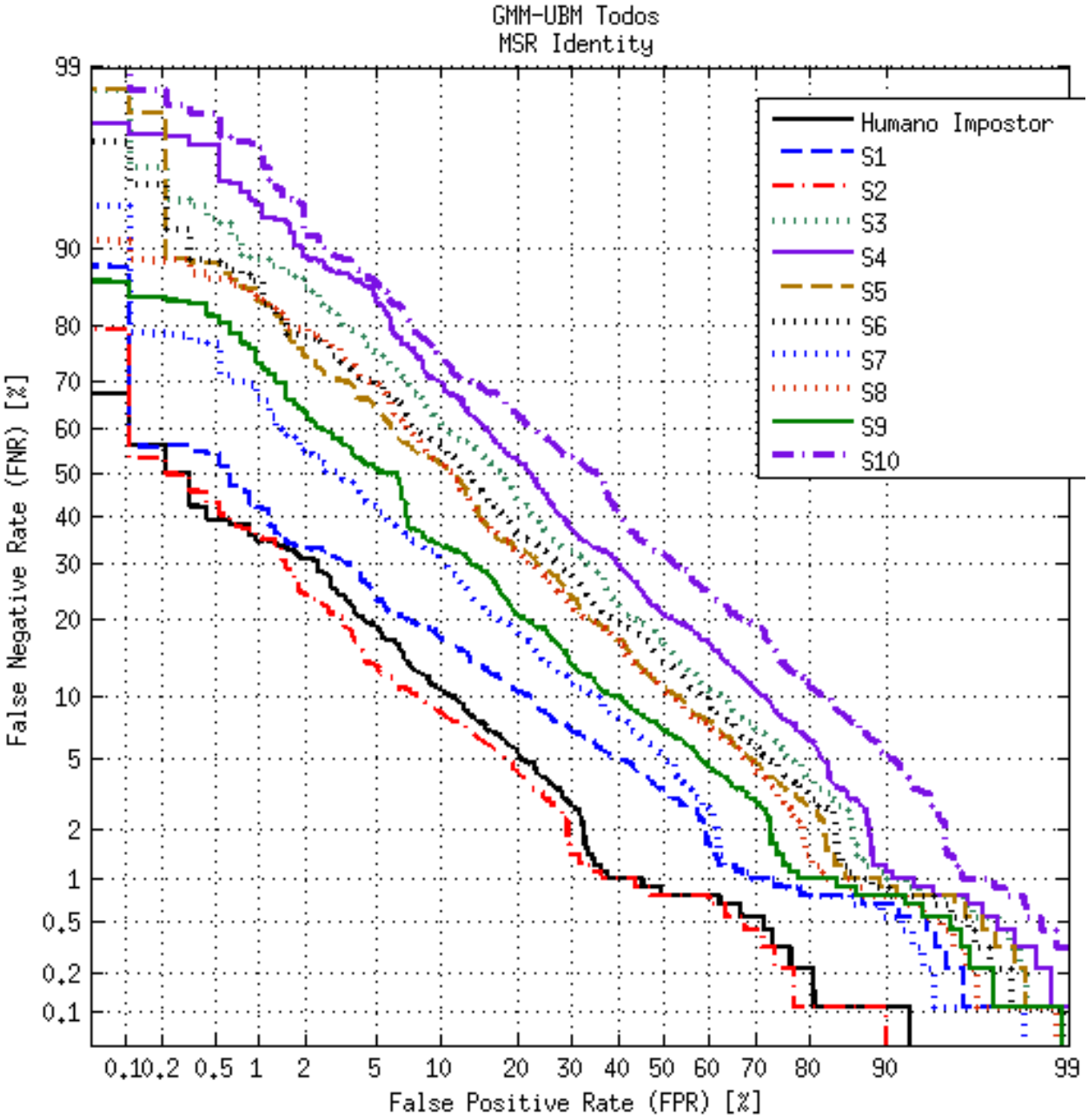


Figura 30. Sistema *GMM-UBM* Todos *MSR Identity*

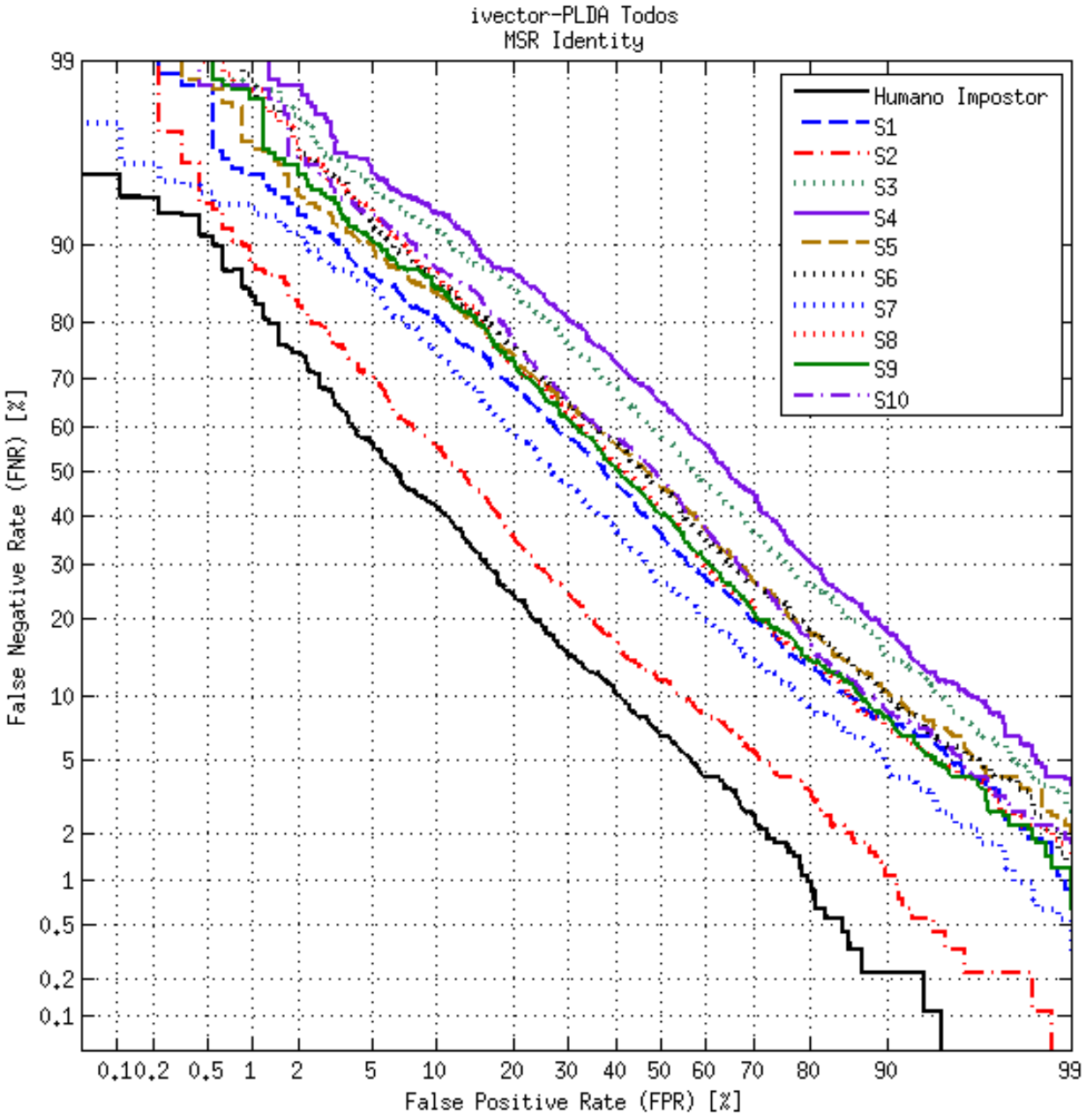


Figura 31. Sistema *ivector-PLDA* Todos *MSR Identity*

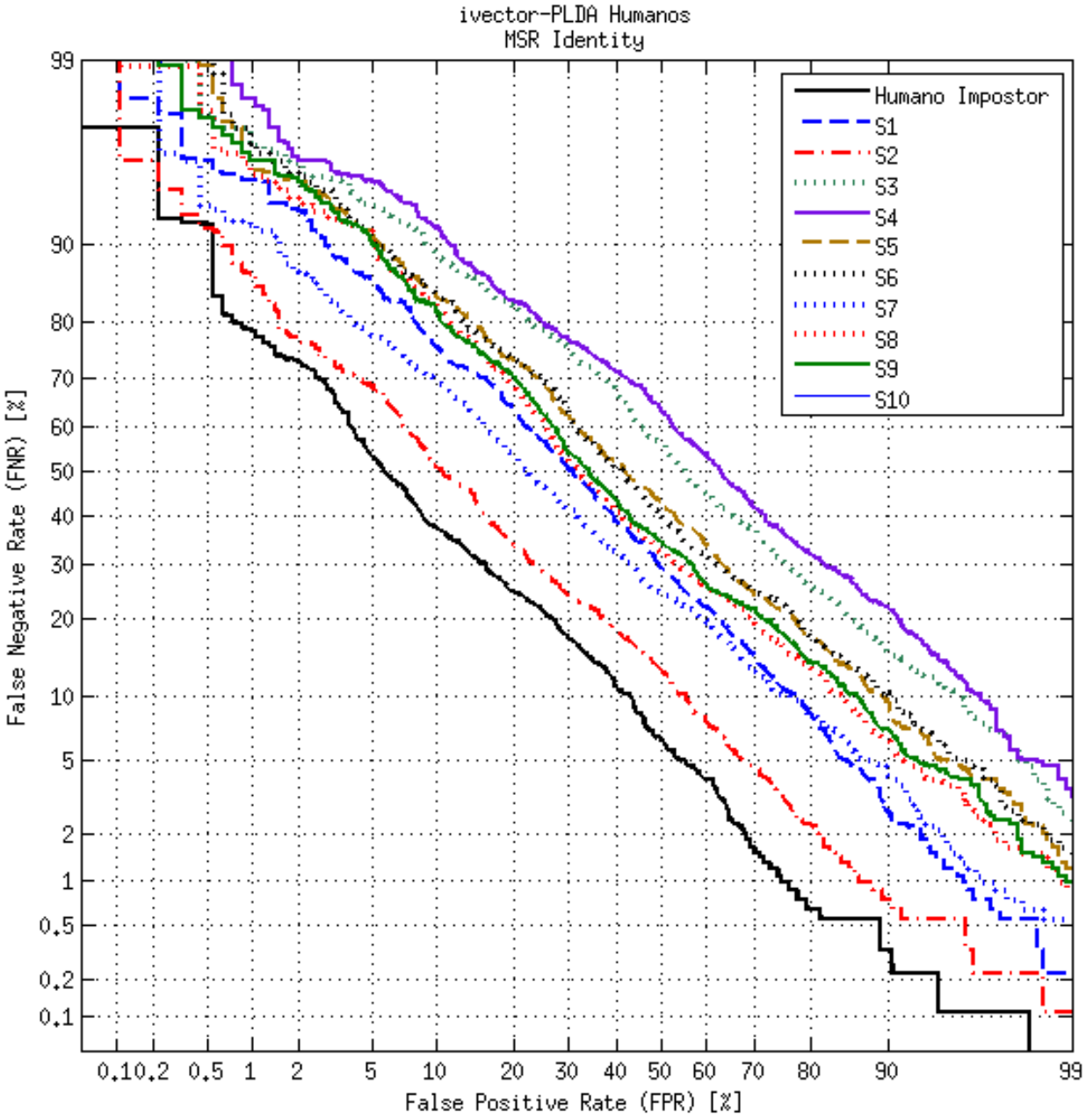


Figura 32. Sistema *ivector-PLDA* Humanos *MSR Identity*