



# LA STÉGANALYSE PAR

## MATRICE D'ADJACENCE DE PIXELS SOUSTRAITS (SUBTRACTIVE PIXEL ADJACENCY MATRIX OU SPAM)

### ET CLASSIFICATEUR SVM

ELOFIR ZAKARIA-CYBERSECURITY

# SOMMAIRE

- **Contexte et Définition de la méthode**
- **Schéma général ( processus de la méthode de stéganalyse)**
- **Fonctionnement de la méthode**  
**(Explication des différents blocs, les techniques et outils utilisés par la méthode)**
- **Algorithme**
- **Application sur notre Dataset**
- **Conclusion**
- **Bibliographie**

## CONTEXTE ET DÉFINITION DE LA MÉTHODE

La stéganalyse est le domaine qui se consacre à l'identification et à l'analyse des techniques de dissimulation d'informations. Dans le contexte de la stéganalyse d'images, l'objectif est de développer des méthodes capables de déceler la présence de contenus dissimulés au sein des images.

La stéganalyse aveugle est une approche visant à repérer des messages dissimulés dans des médias, même sans connaissance de l'algorithme de stéganographie utilisé. Les images numériques sont fréquemment exploitées à des fins de stéganographie, avec des méthodes spatiales telles que la substitution de certains bits dans l'image qui connaissent une popularité.

La stéganalyse peut être catégorisée en tant que ciblée ou aveugle, cette dernière étant conçue pour des scénarios où l'algorithme de stéganographie demeure inconnu.

La méthode de stéganalyse aveugle proposée dans ce travail exploite la matrice d'adjacence de pixels soustractifs pour l'extraction de caractéristiques. Le choix de cet algorithme est motivé par sa capacité à capturer les dépendances spatiales et les changements de valeurs de pixels, le rendant adapté à l'identification des altérations causées par des techniques de stéganographie.

Après, on exploite ces caractéristiques avec l'usage d'un classificateur SVM.

La SVM est une méthode de classification binaire basée sur la création d'un hyperplan qui sépare les données en deux classes 'clean/stego' pour notre cas.

# PROCESSUS DE LA MÉTHODE DE STÉGANALYSE

## Schéma général:

### Définition du Problème :

Il y a une collection d'images originales et d'images stéganographiques, et l'objectif est de classer les images fournies en deux catégories : originales (ou couvertures) et images stéganographiques.

### Dataset :

IStego100K : Ensemble de données de stéganalyse d'images à grande échelle, mélangé avec divers algorithmes stéganographiques, taux d'encapsulation et facteurs de qualité. Notre dataset contient 8104 images clean/stego.

### Extraction de Caractéristiques :

Les caractéristiques des images sont extraites pour sélectionner et conserver des informations pertinentes. Les caractéristiques sont extraites à l'aide de la matrice d'adjacence de pixels soustractifs (SPAM), qui extrait 686 caractéristiques d'une image.

### SPAM (Matrice d'Adjacence de Pixels Soustractifs) :

L'algorithme est principalement divisé en trois étapes :

i. Le modèle calcule les différences entre les pixels dans huit directions.

ii. Pour modéliser les dépendances entre les pixels le long de huit directions, une chaîne de Markov est utilisée entre les paires de différences.

iii. La moyenne des quatre matrices horizontales et verticales et la moyenne des quatre matrices diagonales sont utilisées comme caractéristiques. Les caractéristiques sont formées par les matrices de probabilités de transition de Markov moyennes échantillonnées.

**Classification :** La prochaine étape de la stéganalyse aveugle consiste à classifier les images. La SVM est choisie pour la classification en raison de son efficacité et de sa flexibilité.

**Évaluation:** évaluer la précision des différents kernel du svm.

Collect the dataset of images(cover and stego)

Extract features using SPAM

Train the SVM using training data

Classify the images (testing of data)

## FONCTIONNEMENT DE LA MÉTHODE

La stéganalyse par matrice d'adjacence de pixels soustraits (Subtractive Pixel Adjacency Matrix ou SPAM) est une méthode qui vise à détecter la présence de stéganographie dans des images.

Cette approche utilise des caractéristiques extraites des différences entre les pixels de l'image pour identifier des modèles qui pourraient indiquer la dissimulation d'informations.

Le processus SPAM consiste à calculer des matrices de co-occurrence pour les différences de pixels dans **différentes directions**. Ces matrices capturent des informations statistiques sur les relations entre les valeurs de pixels voisines. En utilisant un modèle markovien d'ordre 1, la méthode SPAM crée des caractéristiques qui sont ensuite utilisées pour la stéganalyse.

Ensuite, ces caractéristiques SPAM sont utilisées comme entrées pour un classificateur SVM (Support Vector Machine).

Le SVM est un modèle d'apprentissage automatique supervisé qui cherche à trouver un hyperplan optimal pour séparer différentes classes de données. Dans le contexte de la stéganalyse, le SVM est formé sur un ensemble de données qui comprend à la fois des images non dissimulées (propres) et des images dissimulées (stéganographiées).

Une fois formé, le SVM peut être utilisé pour classer de nouvelles images comme propres ou stéganographiées en fonction des caractéristiques SPAM extraites.

Cette approche combine donc l'extraction de caractéristiques SPAM, qui capture des propriétés locales des images, avec la puissance de classification du SVM pour détecter la dissimulation d'informations dans des images.

## FONCTIONNEMENT DE LA MÉTHODE

Etapes de la méthode SPAM pour extraction des caractéristiques de nos images :

- Calcul de la matrice de différence (D) :

$$D_{i,j \rightarrow} = I_{i,j} - I_{i,j+1}, \quad i \in \{1, \dots, m\}, j \in \{1, \dots, n-1\}$$

- $D_{i,j \rightarrow}$  : Représente la différence entre les valeurs de pixels en position  $(i, j)$  et son voisin immédiat à droite  $(i, j+1)$  dans la direction horizontale.
- $I_{i,j}$  : Désigne l'intensité du pixel en position  $(i, j)$  dans l'image.

- Caractéristiques SPAM du premier ordre (F1st) :

$$M_{u,v \rightarrow} = \Pr(D_{i,j+1 \rightarrow} = u \mid D_{i,j \rightarrow} = v), \quad u, v \in \{-T, \dots, T\}$$

- $M_{u,v \rightarrow}$  : Représente la probabilité de transition d'une valeur de différence de pixel  $v$  à  $u$  dans la direction horizontale.
- $\Pr(D_{i,j+1 \rightarrow} = u \mid D_{i,j \rightarrow} = v)$  : La probabilité conditionnelle d'observer  $u$  en position  $(i, j+1)$  sachant que  $v$  a été observé en position  $(i, j)$ .
- $u, v \in \{-T, \dots, T\}$  : La plage des valeurs possibles de différence de pixels.

- Caractéristiques SPAM du second ordre (F2nd) :

$$\Pr(D_{i,j+2 \rightarrow} = u \mid D_{i,j+1 \rightarrow} = v, D_{i,j \rightarrow} = w), \quad u, v, w \in \{-T, \dots, T\}$$

- $M_{u,v,w \rightarrow}$  : Représente la probabilité de transition d'une valeur de différence de pixel  $w$  à  $u$  dans la direction horizontale, en considérant les deux différences précédentes  $v$  et  $w$ .
- $\Pr(D_{i,j+2 \rightarrow} = u \mid D_{i,j+1 \rightarrow} = v, D_{i,j \rightarrow} = w)$  : La probabilité conditionnelle d'observer  $u$  en position  $(i, j+2)$  sachant que  $v$  a été observé en position  $(i, j+1)$  et  $w$  a été observé en position  $(i, j)$ .
- $u, v, w \in \{-T, \dots, T\}$  : La plage des valeurs possibles de différence de pixels.

## FONCTIONNEMENT DE LA MÉTHODE

- **Matrices moyennes :**

Pour diminuer la dimensionnalité des caractéristiques, calculer la moyenne des matrices horizontales et verticales séparément, puis des matrices diagonales.

$$F^{1,\dots,k} = \frac{1}{4}(M_{\rightarrow} + M_{\leftarrow} + M_{\downarrow} + M_{\uparrow}),$$

où  $k = (2T + 1)^2$  pour les caractéristiques de premier ordre

Ces formules mettent l'accent sur la modélisation probabiliste des différences de pixels et de leurs transitions, capturant des propriétés statistiques adaptées à la stéganalyse. L'étape de moyennage contribue à réduire la dimensionnalité et prend en compte la symétrie présente dans les images naturelles.

## ALGORITHME

Les deux parties majeures de l'algorithme sont l'algorithme de l'extraction des caractéristiques par matrice d'adjacence de pixels soustraits (Subtractive Pixel Adjacency Matrix ou SPAM) et l'algorithme pour implémenter le classificateur SVM basé sur ces dernières.

Le code pour extraire les caractéristiques d'une image en utilisant l'algorithme de la Matrice d'Adjacence de Pixels Soustraits (SPAM). L'algorithme est appliqué dans les directions horizontale et diagonale pour capturer les propriétés statistiques des différences de pixels. Ces caractéristiques sont ensuite concaténées pour former un vecteur de caractéristiques pour chaque image.

Les deux fonctions principales de cet algorithme d'extraction sont :

- **Fonction GetM3 :**

- Cette fonction prend trois tableaux (L, C et R), représentant les colonnes de gauche, du centre et de droite d'un tableau 2D, en entrée.
- Elle calcule les cooccurrences des différences de pixels dans une plage spécifiée (-T à T).
- Le résultat est un tableau 3D M représentant la matrice de cooccurrence.
- La matrice est ensuite aplatie et normalisée.

- **Fonction spam\_extract\_2 :**

- Cette fonction prend une image représentée en tant que tableau NumPy (X) et un paramètre T.
- Elle extrait des caractéristiques en utilisant l'algorithme SPAM dans les directions horizontale et diagonale.
- L'image est divisée en trois canaux de couleur (RGB), et les différences de pixels sont calculées dans les directions spécifiées.
- La fonction GetM3 est utilisée pour calculer les matrices de cooccurrence.
- Enfin, les matrices résultantes sont moyennées pour créer deux vecteurs de caractéristiques (F1 et F2), qui sont concaténés pour former le vecteur de caractéristiques final.

# ALGORITHME

Aperçu:

```

def spam_extract_2(X, T):
    # horizontal left-right
    X = np.concatenate((X[:, :, 0], X[:, :, 1], X[:, :, 2]), axis=1)
    D = X[:, :-1] - X[:, 1:]
    L = D[:, 2:]
    C = D[:, 1:-1]
    R = D[:, :-2]
    Mh1 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # horizontal right-left
    D = -D;
    L = D[:, :-2]
    C = D[:, 1:-1]
    R = D[:, 2:]
    Mh2 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # vertical bottom top
    D = X[:-1, :] - X[1:, :]
    L = D[2:, :]
    C = D[1:-1, :]
    R = D[:-2, :]
    Mv1 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # vertical top bottom
    D = -D
    L = D[:-2, :]
    C = D[1:-1, :]
    R = D[2:, :]
    Mv2 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # diagonal left-right
    D = X[:-1, :-1] - X[1:, 1:]
    L = D[2:, 2:]
    C = D[1:-1, 1:-1]
    R = D[:-2, :-2]
    Md1 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # diagonal right-left
    D = -D
    L = D[:-2, :-2]
    C = D[1:-1, 1:-1]
    R = D[2:, 2:]
    Md2 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # minor diagonal left-right
    D = X[1:, :-1] - X[:-1, 1:]
    L = D[:-2, 2:]
    C = D[1:-1, 1:-1]
    R = D[2:, :-2]
    Mm1 = GetM3(L.copy(), C.copy(), R.copy(), T)

    # minor diagonal right-left
    D = -D
    L = D[2:, :-2]
    C = D[1:-1, 1:-1]
    R = D[:-2, 2:]
    Mm2 = GetM3(L.copy(), C.copy(), R.copy(), T)

    F1 = (Mh1+Mh2+Mv1+Mv2)/4
    F2 = (Md1+Md2+Mm1+Mm2)/4
    F = np.concatenate((F1, F2), axis=0)
    return F

```

```

def GetM3(L,C,R,T):
    # marginalization into borders
    L = np.clip(L, -T, T).flatten('F')
    C = np.clip(C, -T, T).flatten('F')
    R = np.clip(R, -T, T).flatten('F')

    # get cooccurrences [-T...T]
    M = np.zeros((2*T+1, 2*T+1, 2*T+1))
    for i in range(-T, T+1, 1):
        C2 = C[L==i]
        R2 = R[L==i]
        for j in range(-T, T+1, 1):
            R3 = R2[C2==j]
            for k in range(-T, T+1, 1):
                M[i+T, j+T, k+T] = np.sum(R3==k)

    # normalization
    M = M.flatten('F')
    M /= np.sum(M)
    return M

```

## ALGORITHME

Partie de classification svm.

### Entraînement du modèle SVM :

- Les caractéristiques extraites sont normalisées.
- Un modèle SVM est entraîné avec différents noyaux (linéaire, polynômial, RBF, sigmoïdal) sur les données d'entraînement normalisées.
- La précision du modèle est évaluée sur les données de test.

```
x = new_df['Spam_Features'].tolist()
y = new_df['Label']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

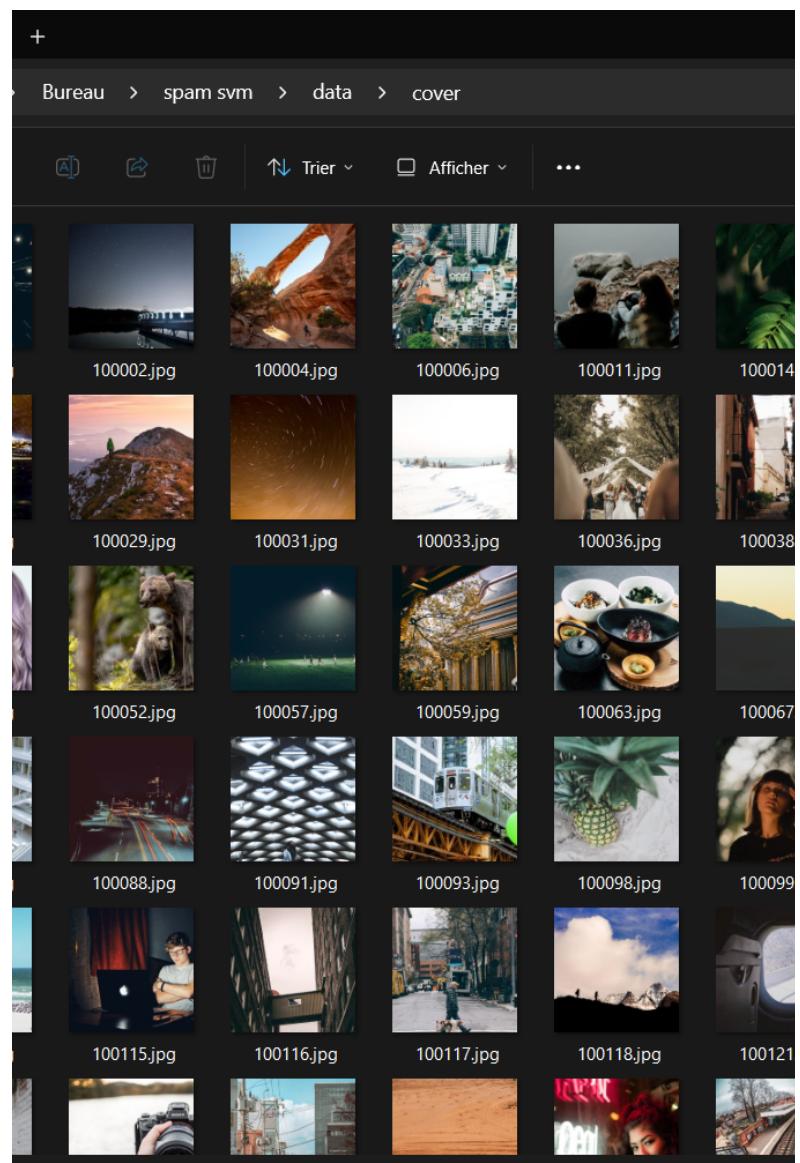
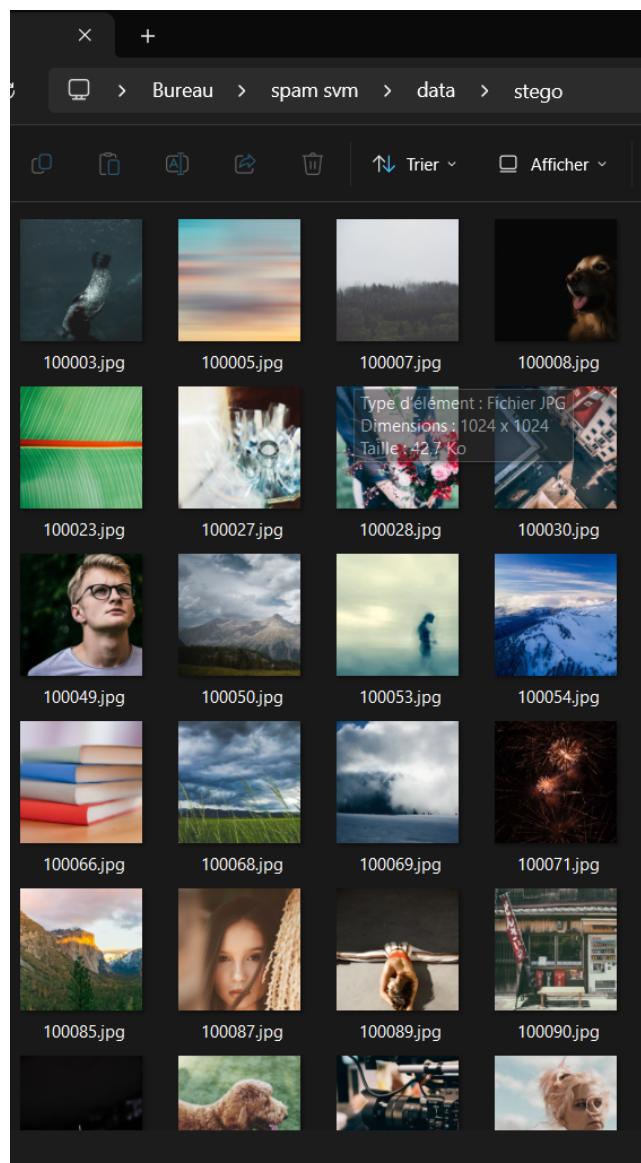
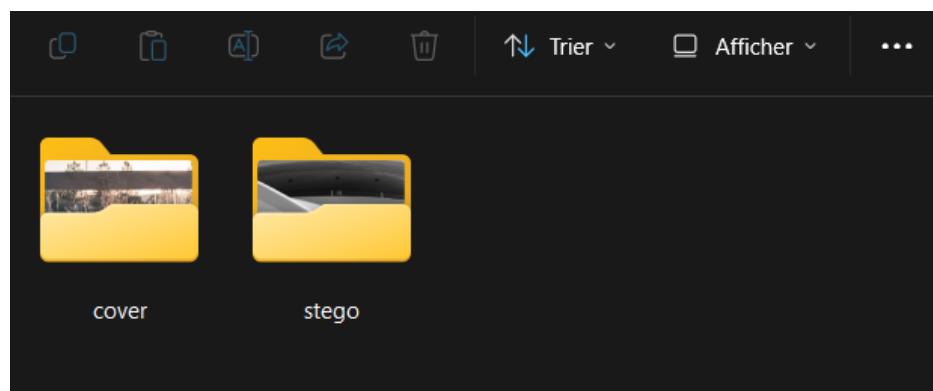
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernels:
    svm_classifier = SVC(kernel=kernel)
    svm_classifier.fit(x_train_scaled, y_train)
    y_pred = svm_classifier.predict(x_test_scaled)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy with {kernel} kernel: {accuracy:.2%}")
```

# APPLICATION SUR NOTRE DATASET

Notre dataset contient 8104 images stegano et cover



## APPLICATION SUR NOTRE DATASET

L'entraînement du svm a pris presque 12 heures pour tous les kernels

```
Entrée [*]: import pandas as pd
from tqdm import tqdm

tqdm.pandas()
df['Spam_Features'] = df['Image_Path'].progress_apply(SPAM)
15%|██████████| 1185/8104 [41:24<3:49:33, 1.99s/it]
```

```
Entrée [*]: import pandas as pd
from tqdm import tqdm

tqdm.pandas()
df['Spam_Features'] = df['Image_Path'].progress_apply(SPAM)
33%|██████████| 2692/8104 [4:21:09<3:10:24, 2.11s/it]
```

```
import pandas as pd
from tqdm import tqdm

tqdm.pandas()
df['Spam_Features'] = df['Image_Path'].progress_apply(SPAM)
88%|██████████| 7124/8104 [7:01:11<36:21, 2.23s/it]
```

```
: import pandas as pd
from tqdm import tqdm

tqdm.pandas()
df['Spam_Features'] = df['Image_Path'].progress_apply(SPAM)
100%|██████████| 100/100 [12:54<00:00, 7.75s/it]
```

## APPLICATION SUR NOTRE DATASET

Les résultats d'accuracy indiquent la performance du modèle SVM avec différents noyaux sur les données de test.

- Accuracy with linear kernel: 85.00%
  - Le modèle atteint une précision de 85% lorsqu'un noyau linéaire est utilisé. Cela signifie que le modèle a correctement classé 85% des échantillons dans l'ensemble de test.
- Accuracy with poly kernel: 85.00%
  - Le modèle atteint également une précision de 85% avec un noyau polynomial. La performance est similaire à celle du noyau linéaire.
- Accuracy with rbf kernel: 80.00%
  - Lors de l'utilisation d'un noyau RBF (Radial Basis Function), la précision diminue à 80%. Cela peut indiquer que le modèle ne généralise pas aussi bien sur les données de test avec ce noyau spécifique.
- Accuracy with sigmoid kernel: 80.00%
  - De manière similaire au noyau RBF, le noyau sigmoïdal atteint une précision de 80%. Les noyaux sigmoïdaux sont moins couramment utilisés dans la pratique.

Il semble y avoir une certaine similitude dans les performances entre les noyaux linéaire et polynomial, tandis que les noyaux RBF et sigmoïdal donnent des résultats légèrement moins bons.

```
x = new_df['Spam_Features'].tolist()
y = new_df['Label']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernels:
    svm_classifier = SVC(kernel=kernel)
    svm_classifier.fit(x_train_scaled, y_train)
    y_pred = svm_classifier.predict(x_test_scaled)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy with {kernel} kernel: {accuracy:.2%}")
```

```
Accuracy with linear kernel: 85.00%
Accuracy with poly kernel: 85.00%
Accuracy with rbf kernel: 80.00%
Accuracy with sigmoid kernel: 80.00%
```

## **CONCLUSION**

**La stéganalyse est une tâche complexe en raison de la nature subtile des changements introduits dans les images stéganographiques.**

**La stéganalyse par matrice d'adjacence de pixels soustraits (Subtractive Pixel Adjacency Matrix ou SPAM) qui vise à détecter la présence de stéganographie dans des images utilise des caractéristiques extraites des différences entre les pixels de l'image pour identifier des modèles qui pourraient indiquer la dissimulation d'informations.**

**Notre méthode a fourni de bons résultats, mais elle peut être améliorée en optimisant les performances du SVM et en augmentant la taille de notre ensemble de données tout en diversifiant les méthodes de stéganographie utilisées pour les images stégos.**

## BIBLIOGRAPHIE

- **Blind Steganalysis for Digital Images using Support Vector Machine Method.**  
2016 International Symposium on Electronics and Smart Devices (ISESD) November 29-30, 2016
- **Steganalysis by Subtractive Pixel Adjacency Matrix**  
Tomáš Pevný and Patrick Bas and Jessica Fridrich, IEEE member
- **IStego100K: Large-scale Image Steganalysis Dataset, mixed with various steganographic algorithms, embedding rates, and quality factors.**