# Stéganographie par CNN et approche jeu à 3 joueurs

Mehdi YEDROUDJ[1], Frédéric COMBY[1], Marc CHAUMONT[1,2]

[1]UNIVERSITE MONTPELLIER, UMR5506-LIRMM, F-34095 Montpellier Cedex 5, France

[2]UNIVERSITE DE NIMES, F-30021 Nîmes Cedex 1, France

yedroudj@lirmm.fr, comby@lirmm.fr, chaumont@lirmm.fr

**Résumé –** La stéganographie d'images vise à intégrer en toute sécurité des informations secrètes dans des images. Actuellement, les algorithmes d'insertion adaptative tels que S-UNIWARD ou Mi-POD sont parmi les plus sûrs et les plus utilisés en stéganographie. L'arrivée de l'apprentissage profond et des GANs a fait apparaître de nouvelles techniques. Parmi celles-ci, on retrouve l'approche du jeu à 3 joueurs (J3J) où trois réseaux s'affrontent. Nous proposons ici un système stéganographique basé sur le J3J comme une alternative rigoureuse à deux publications récentes (1) et (2). Notre approche donne de meilleurs résultats que les travaux précédents, et ouvre la voie à des recherches futures sur le sujet.

**Abstract –** Image steganography's main goal is to hide secret data into images. For now, S-UNIWARD or Mi-Pod are among the best adaptive insertion algorithms. The advent of deep learning and GANs has revealed new techniques, among which is the 3 player game where 3 networks fight against each other. In this paper, we introduce a 3 player game steganographical system as a rigorous alternative to two recent publications (1) et (2). Our approach provides better results and offers great perspectives.

## 1 Introduction

En stéganographie, les algorithmes récents d'insertion prennent en compte le contenu du support afin de mieux cacher le message secret : ils sont dit adaptatifs (3), (4). Cependant du point de vue de la théorie du jeu, ils sont qualifiés de naïf (5) (6). En effet, pour un algorithme d'insertion, l'évolution de la stratégie de stéganalyse n'est pas prise en compte. Il est plus intéressant de proposer une stéganographie adaptative optimale, aussi appelée *stéganographie adaptative stratégique* (6). Avec un tel algorithme, la probabilité de modification de chaque pixel est réglée pour assurer l'équilibre de Nash dans le jeu du chat et de la souris entre stéganographes (Alice/Bob) et stéganalyste (Eve).

Deux articles récents (1; 2) ont proposé des algorithmes d'insertion stratégique utilisant 3 réseaux de neurones (CNN) mis à jour de manière itérative. Chacun des 3 CNN représente un des acteurs de la stéganalyse : l'**Agent-Alice**, l'**Agent-Bob** et l'**Agent-Eve**. Ces articles abordent les concepts du jeu à 3 joueurs mais contiennent des erreurs.

- Pas d'utilisation explicite d'une clé secrète partagée au cours de l'insertion/extraction. L'insertion du message s'effectue donc toujours avec la même clé. Ceci n'est pas recommandé en stéganographie car cela conduit à une très haute détectabilité (7) ;
- Absence de discrétisation des images générées : L'image fournie à Bob devrait contenir des pixels dont les valeurs sont entières dans $\{1, ..., 255\}$. Dans (1; 2), l'**Agent-Alice** génère des images à valeurs réelles, qui ne peuvent pas être stockées dans un format d'image standard, ou dans un format adhoc qui serait fortement suspect ;
- Le fonctionnement est limité à de petites images dû à l'utilisation des couches entièrement connectées (FC), qui génèrent des coûts en mémoire et en calcul élevés ;
- L'algorithme de stéganalyse utilisé : ATS (8), date de 2015 et est conçu pour traiter un problème de *cover-source mismatch*. Ce n'est donc pas le scénario le plus approprié pour évaluer la sécurité empirique d'un algorithme d'insertion.

Dans cet article nous proposons un système de stéganographie basé sur l'approche J3J. La section 2 aborde le concept général de la stéganographie avec un J3J. La section 3 présente une architecture traitant les problèmes non résolus dans (1; 2). La section 4 présente les premiers résultats expérimentaux obtenus et nous concluons sur l'apport de cette approche dans la section 5.

## 2 Concept du jeu à 3 joueurs (J3J)

Les notation suivantes seront utilisées pour ce document :

- **a** pour un vecteur ou une matrice ;
- $a$ pour un scalaire ;
- "$\times$" pour séparer les dimensions d'un vecteur multidimensionnel ;

### 2.1 Concept général

Cette partie de l'article présente l'idée général du jeu à 3 joueurs et décrit le rôle de chaque agent. Ceci est illustré sur la Fig. 1 ou chaque agent du J3J est représenté par un CNN : **Agent-Alice**, **Agent-Bob**, et **Agent-Eve**. On appellera $\boldsymbol{z} \in \{0, ..., 255\}^{w \times h}$ une image non encore étiquetée. $l$ sera l'éti-

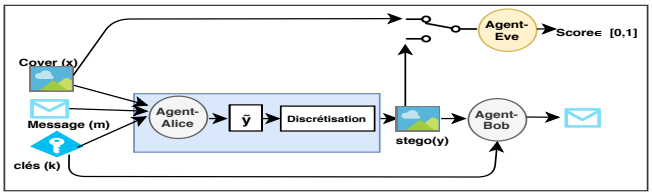


FIGURE 1 – L'architecture générale du *jeu à 3 joueurs*.

quette de l'image avec $l \in \{0,1\}$, $l = 0$ (resp. 1) si $\boldsymbol{z}$ est une image non modifiée "*cover*" (resp. modifiée "*stego*").

Soit $\mathbf{x} \in \{0,...,255\}^{w \times h}$ une image *cover* composée de $w \times h$ pixels et $\mathbf{y} \in \{0,...,255\}^{w \times h}$ une image *stego* générée par l'**Agent-Alice**. Soit **m** un message binaire secret de $m$ bits que l'**Agent-Alice** veut envoyer à l'**Agent-Bob** et **m'** le message binaire extrait par le destinataire. Soit **k** la clé partagée entre ces mêmes agents où **k** est un vecteur binaire de taille $k$.

Le système prend en entrée une image source **x**, un message secret **m** et une clé **k**. Ces entrées sont d'abord introduites dans le réseau **Agent-Alice** qui génère une image *stego* $\tilde{\mathbf{y}}$ dont les valeurs des pixels sont des réels ($\tilde{\mathbf{y}} \in \mathbb{R}^{w \times h}$). Ensuite, le module de discrétisation reçoit $\tilde{\mathbf{y}}$ et génère **y** une image *stego* avec des valeurs entières. **y** est ensuite donné à la fois à l'**Agent-Bob** et à l'**Agent-Eve**. L'**Agent-Bob** essaie de récupérer le message secret **m** à partir de **y** en utilisant la clé secrète partagée **k**. Quant à l'**Agent-Eve**, elle reçoit une image **z** et renvoie un score de l'appartenance de **z** aux classes *cover* ou *stego*.

## 2.2 Fonction de perte de chacun des trois agents

La fonction de perte (*loss*) définit le critère à optimiser par les réseaux de neurones.

La **Fonction de perte de l'Agent-Eve** consiste à minimiser la distance entre l'étiquette $l$ d'une image **z** et la prédiction de l'**Agent-Eve** :

$$\mathcal{L}_{Eve} = dist(l - \textbf{Agent-Eve}(\mathbf{z})). \tag{1}$$

La **Fonction de perte de l'Agent-Bob** consiste à minimiser une distance entre le message reconstruit par **Agent-Bob m'** et le message dissimulé par l'**Agent-Alice m**. Il s'agit généralement une distance $L2$.

$$\mathcal{L}_{Bob} = dist(\mathbf{m}, \mathbf{m'}). \tag{2}$$

La **Fonction de perte de l'Agent-Alice** est la somme pondérée de trois termes : $\mathcal{L}_{bob}$, $\mathcal{L}_{Eve}$, et *dist*(**x**,**y**) qui est la distance calculée entre **x** et **y**. Les coefficients $\lambda_A$, $\lambda_B$, $\lambda_E \in [0,1]$ et leur somme est égale à 1. Les termes de pertes sont également normalisés afin d'avoir des amplitudes similaires.

$$\mathcal{L}_{Alice} = \lambda_A \cdot dist(\mathbf{x},\mathbf{y}) + \lambda_B \cdot \mathcal{L}_{Bob} - \lambda_E \cdot \mathcal{L}_{Eve}. \tag{3}$$

## 2.3 Processus d'apprentissage :

Le système stéganographique est entraîné pendant $M_i$ itérations. Pour chaque itération, un lot réduit d'images sources, de messages secrets et de clés alimentent le système. L'entraînement se fait alternativement : dans un premier temps les acteurs **Agent-Alice** et **Agent-Bob** sont entraînés conjointement pendant *j* itérations, tandis que l'**Agent-Eve** est figé. Puis l'**Agent-Eve** est entraîné pendant *j'* itérations, tandis que les réseaux **Agent-Alice** et **Agent-Bob** sont figés.

# 3 Architecture de notre système stéganographique

Nous proposons un nouvelle approche de *stéganographie stratégique adaptative*, basée sur l'idée du J3J. Cette approche s'affranchit des défauts expliqués précédemment. Pour ceci, nous avons : **1)** intégré une clef secrète dans le système stéganographique, **2)** géré le problème de discrétisation, **3)** garanti une solution évolutive grâce à une architecture constituée uniquement de couches de convolutions. Notre approche possède également deux propriétés intéressantes : tout d'abord, elle est "adaptative au débit binaire" (il est inutile d'entraîner à nouveau le système dès que le débit binaire est modifié comme il est géré à l'intérieur de réseau). Ensuite, nous choisissons un stéganalyste fort : "Yedroudj-Net" (9) [1] pour l'**Agent-Eve** assurant ainsi potentiellement une meilleure sécurité.

L'architecture proposée est illustrée sur la Fig. 2. L'**Agent-Alice** utilise la clé **k** ainsi qu'un générateur de nombres pseudo-aléatoires (PRNG) pour étaler le message secret **m** dans une matrice notée $\mathbf{s}^{(\mathbf{m})} \in \{0,1\}^{\mathrm{w} \times \mathrm{h}}$ de même taille que **x**. Puis le résultat de la couche *SRM-F* est concaténé avec $\mathbf{s}^{(\mathbf{m})}$ et envoyé à *conv_Stack0* qui est composé d'un ensemble de couches de convolutions. La sortie est une carte de modifications $\boldsymbol{n} \in \{-1,0,1\}^{w \times h}$. Cette carte est ensuite ajoutée à l'image source **x** pour générer l'image *stego* **y**. la fonction de perte de l'**Agent-Alice** est définie par l'équation 4 :

$$\mathcal{L}_{Alice} = \lambda_A \cdot (dist(\mathbf{x},\mathbf{y}) - \beta) + \lambda_B \cdot \mathcal{L}_{Bob} - \lambda_E \cdot \mathcal{L}_{Eve} \tag{4}$$

où $\lambda_A, \lambda_B, \lambda_E \in [0,1]$. Notez que $\beta$, le taux de modification, contrôle la discrétion du réseau de l'**Agent-Alice**, c'est-à-dire le nombre de pixels que l'**Agent-Alice** est autorisé à modifier.

L'architecture de l'**Agent-Bob** intègre un U-Net (10) (voir Fig. 2). Son but est de reconstruire l'image source **x** depuis l'image *stego* **y**. La carte de modifications **n** est donc le résultat de la soustraction de l'image reconstruite **x'** et de l'image *stego* **y**. Ensuite, nous convoluons **x'** avec les noyaux SRM (11) en utilisant la couche *SRM-F* (voir Fig. 2). La sortie est ensuite concaténée avec les cartes de modification obtenues et envoyée à *conv_Stack1*.

Si nous considérons *conv_Stack1* et *conv_Stack0* comme des fonctions, alors nous voulons que la fonction *conv_Stack0* soit proche d'une fonction réciproque de *conv_Stack1*.

La fonction de pertes de l'**Agent-Bob** est définie par :

$$\mathcal{L}_{Bob} = 1/2(\mathcal{L}_{rec} + \mathcal{L}_m) \tag{5}$$

1. Pour plus de détails voir : `http://www.lirmm.fr/~chaumont/Yedroudj-Net.html`

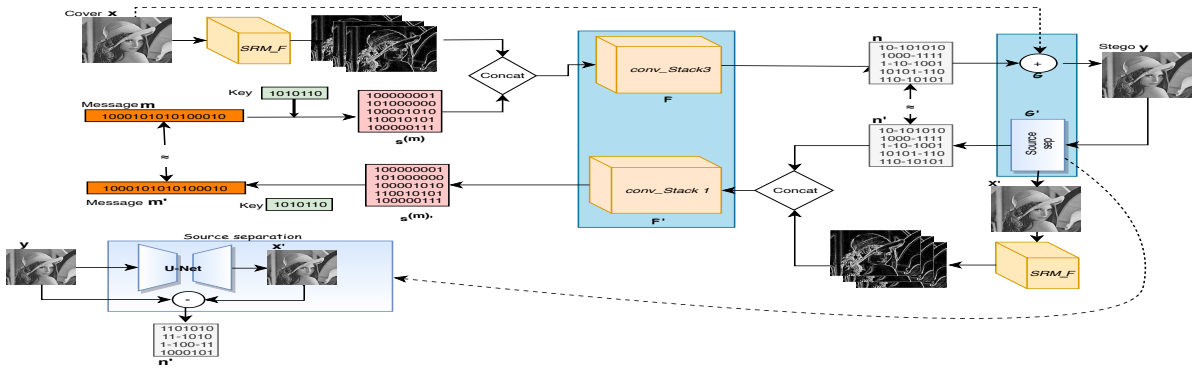


FIGURE 2 – L'architecture de deux réseau utilisés pour l'insertion et l'extraction.

avec : $\mathcal{L}_{rec} = EQM(\mathbf{x'}, \mathbf{x})$ et $\mathcal{L}_m = EQM(\mathbf{m}, \mathbf{m'})$, où $EQM$ est l'erreur moyenne quadratique.

# 4 Expérimentations

## 4.1 Base de données et logiciels

Les expériences sont réalisées sur deux bases de données : BOSSBase 1.01 (12) et BOWS2 (13), chacune contenant 10000 images en niveaux de gris 8 bits de taille $512 \times 512$. Les images ont été réduites à $256 \times 256$ pixels pour faire face aux limitations de la plate-forme de calcul GPU, du temps de calcul, du temps de l'entraînement de J3J. L'architecture proposée est implémentée avec TensorFlow V 1.6. Nous comparerons nos résultats avec deux algorithmes d'insertion S-UNIWARD (3), et WOW (4).Toutes nos expériences ont été réalisées sur la plate-forme GPU NVIDIA Titan X.

## 4.2 Entraînement, validation et test

La phase d'entraînement nécessite un modèle pré-entraîné de l'**Agent-Eve**. L'algorithme S-Uniward permet de générer 10000 images *stego*s à partir de la base BOSSBase. Un réseau "Yedroudj-Net" est entraîné sur 4000 paires *cover*/*stego*. Une fois ce modèle appris, il est transféré à l'**Agent-Eve**.

Commence alors l'apprentissage des 3 joueurs sur la base BOSSBase. Les messages sont générés à l'aide d'un PRNG pour une charge utile choisie. Nous utilisons l'optimiseur adaptatif Adam pour l'entraînement de notre système où la taille du *mini-batch* est fixée à 4. Les valeurs de $\lambda_A$, $\lambda_B$ et $\lambda_E$ sont respectivement fixées empiriquement à $0,2$, $0,4$ et $0,4$.

Pendant l'entraînement, nous fixons le nombre maximum d'itérations *M_i* à 1 million d'itérations. L'entraînement est alterné entre les trois agents : l'**Agent-Alice** et l'**Agent-Bob** sont regroupés et effectués pour 50 itérations (*j*=50) contre 1 itération pour l'**Agent-Eve** (*j'*=1).

Lorsque les pertes des trois réseaux tendent à être stables, nous figeons l'apprentissage et nous intégrons le module de discrétisation dans l'**Agent-Alice** (voir Fig. 1) [2].

Les performances de notre méthode sont évaluées par deux critères :

— la probabilité d'erreur ($P_e$) d'un stéganalyseur donné, avec $P_e$ la moyenne du taux de fausses alarmes et du taux de détections manquées ;

— l'erreur de transmission utilisant le taux d'erreur binaire (TEB : nombre de bits que Bob peut récupérer sans erreur).

À ce jour, aucune des architectures de la littérature pour le J3J n'est capable de cacher un message qui pourrait être récupéré sans erreur. De ce fait, le lecteur doit envisager l'utilisation d'un code correcteur.

Pour la phase de test, l'**Agent-Alice** commence par générer 10,000 images *stego* à partir d'images *cover* de BOWS2 dans lesquelles un message aléatoire est inséré à l'aide d'une clé aléatoire. Les images générées sont utilisées pour évaluer la précision de récupération du message de Bob d'une part et la sécurité de notre système d'autre part.

## 4.3 Résultats

Notre architecture permet de contrôler la puissance de bruit introduit par l'**Agent-Alice** lors de l'insertion du message. La Fig. 3 illustre l'évolution du TEB et de la probabilité d'erreur ($P_e$) pour différentes valeurs de $\beta$ (Eq. 4) avec une charge utile fixe de 0,4 bpp. Le stéganalyseur utilisé est "Yedroudj-Net". Nous pouvons voir que, pour $\beta = 0,4$, l'**Agent-Eve** obtient une valeur de $P_e$ proche de 6% quand Bob a un $TEB = 3\%$. Pour $\beta = 0,2$, la sécurité de notre système s'améliore avec $P_e = 9\%$, bien que l'extraction du message devienne plus difficile pour Bob avec $TEB = 6\%$. Pour $\beta \leq 0,05$, les images générées contiennent un message plus difficilement détectable. Cependant, le TEB augmente considérablement. L'objectif de la stéganographie est d'avoir un $TEB = 0$ tout en maximisant la valeur de l'erreur de détection ($P_e$). Nous devons alors trouver la valeur optimale de $\beta$ impliquant une modification minimale du support permettant d'avoir une sécurité maximale ainsi qu'une extraction parfaite de message par Bob (avec utilisation d'un code correction erreur). Pour une charge utile de taille 0,4bpp, la valeur de $\beta$ provoquant une variation soudaine

2. l'activation du module de discrétisation au début de la phase d'entraînement empêche la convergence du système car la fonction arrondi n'est pas dérivable.

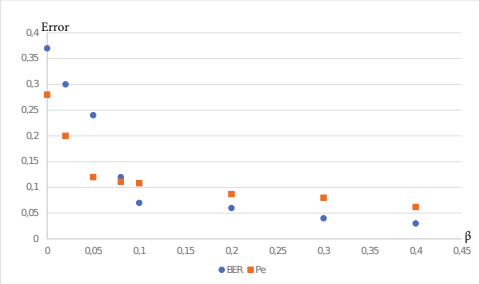


FIGURE 3 – TEB et $P_e$ pour une charge utile de 0,4 bpp en fonction du taux de modification $\beta$.

du TEB (c'est-à-dire un point d'inflexion) est autour de $0,1$. À cet endroit le TEB est de $0,06$.

Les erreurs peuvent être corrigées avec un code de Hamming avec les paramètres $\{15, 11, 3\}$. Dans ce cas, la taille réelle de la charge utile passe de $0,4$ à $(0,4 \cdot 11)/15 = 0,293$bpp $\approx \mathbf{0,3}$bpp. Pour cette charge utile réelle nous obtenons un $P_e$ d'environ $\mathbf{11}\%$. Pour comparer, nous exécutons une stéganalyse avec "Yedroudj-Net" contre deux algorithmes stéganographiques S-UNIWARD et WOW à une charge utile réelle de 0,3bpp en utilisant la base de données BOWS2, la probabilité d'erreur ($P_e$) est de 27,3% et 22,4% pour S-UNIWARD et WOW respectivement. Même si le niveau de sécurité de notre système actuellement inférieur à celui des algorithmes traditionnels tels que S-UNIWARD, WOW ces résultats d'un système J3J moderne sont encourageants. Par ailleurs, ce papier montre une réelle amélioration par rapport à HiDDEeN (2) et GSIVAT (1) et ouvre la voie à de nouvelles pistes de recherche.

## 5 Conclusion et perspectives

Nous avons présenté l'idée générale et comment utiliser correctement l'approche du jeu à 3 joueurs en stéganographie. Une architecture basée sur cette approche a été proposée. Au lieu de modifier directement l'image source, impliquant une puissance de bruit significatif, nous avons généré une carte de modifications avec des valeurs appartenant à {-1,0,1}. L' image *stego* est ensuite créée en ajoutant la carte de modifications à l'image source. Cette architecture est beaucoup plus sûre que les précédents travaux (1; 2) en stéganographie avec l'approche J3J.

Les résultats obtenus ne surpassent pas ceux d'algorithmes classiques, cependant ils vérifient les concepts de notre approche. Nous nous attendons à ce que ce travail débouche sur des pistes de recherche fructueuses. Dans le cadre de travaux futurs, nous pourrons étudier la possibilité de synchroniser davantage l'**Agent-Alice** et l'**Agent-Bob**, et ainsi d'améliorer les performances générales. Nous pourrons également essayer d'utiliser des fonctions de pertes plus subtiles afin d'aider les réseaux à converger vers une meilleure solution. De plus, trouver un moyen théorique de calculer le taux de changement $\beta$ pourrait accélérer le processus d'apprentissage. Les valeurs de $\lambda_A$, $\lambda_B$, $\lambda_E$ pourraient également être étudiées plus en détail.

## Références


[1] J. Hayes et G. Danezis. Generating steganographic images via adversarial training. Dans *Advances in Neural Information Processing Systems*, pages 1951–1960, dec 2017.

[2] J. Zhu, R. Kaplan, J. Johnson, et L. Fei-Fei. Hidden : Hiding data with deep networks. Dans *Proceedings of the 15th European Conference on Computer Vision*, pages 682–697, sep 2018.

[3] V. Holub, J. Fridrich, et T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, jan 2014.

[4] V. Holub et J. Fridrich. Designing Steganographic Distortion Using Directional Filters. Dans *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pages 234–239, dec 2012.

[5] P. Schöttle et R. Böhme. A game-theoretic approach to content-adaptive steganography. Dans *Proceedings of the 14th International Conference on Information Hiding*, pages 125–141, 2012.

[6] P. Schöttle et R. Böhme. Game theory and adaptive steganography. *IEEE Transactions on Information Forensics and Security*, 11(4) :760–773, apr 2016.

[7] L. Pibre, J. Pasquet, D. Ienco, et M. Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch. Dans *Proceedings of Media Watermarking, Security, and Forensics*, pages 1–11, feb 2016.

[8] D. Lerch-Hostalot et D. Megías. Unsupervised steganalysis based on artificial training sets. *Engineering Applications of Artificial Intelligence*, 50 :45–59, 2016.

[9] M.Yedroudj, F. Comby, et M. Chaumont. Yedroudj-Net : An efficient CNN for spatial steganalysis. Dans *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2092–2096, 2018.

[10] O. Ronneberger, P.Fischer, et T. Brox. U-net : Convolutional networks for biomedical image segmentation. pages 234–241. Springer, 2015.

[11] J. Fridrich et J. Kodovský. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3) :868–882, jun 2012.

[12] P. Bas, T. Filler, et T. Pevný. Break our steganographic system : The ins and outs of organizing BOSS. Dans *International workshop on information hiding*, pages 59–70, 2011.

[13] P. Bas et T. Furo. Bows-2 contest (break our watermarking system), 2007.