

IFT6350 – Projet Cryptographie Visuelle

Thomas LEPLUS

28 avril 2003

1 Introduction

Les cryptologues ont montré depuis longtemps qu'il existe un système cryptographique parfait dénommé le masque jetable. Le principe est que deux interlocuteurs, disons Alice et Bob, veulent échanger confidentiellement un message m long de n bits. Pour cela, ils doivent partager au préalable une clé aléatoire k longue aussi de n bits. Par la suite, Alice fait un OU exclusif bit à bit entre m et k , puis elle envoie le résultat c (le cryptogramme) à Bob. Celui-ci n'a plus qu'à refaire un OU Exclusif entre c et k pour retrouver m .

La théorie de l'information de Claude SHANNON permet de prouver qu'un tel système est parfaitement incassable tant que la clé est bien aléatoire et que l'on ne réutilise jamais une clé plusieurs fois (d'où le nom de masque jetable).

Dans la pratique, ce système est presque inutilisable car Alice et Bob doivent partager autant de bits aléatoires qu'ils ne souhaitent en transmettre. Si Alice et Bob ont les moyens de se mettre secrètement d'accord sur n bits aléatoires, ils auraient probablement aussi bien pu transmettre directement le message par la même méthode¹.

Lors de la conférence EUROCRYPT'94, Moni NAOR et Adi SHAMIR [1] ont introduit l'idée de cryptographie visuelle. La question est de savoir si l'on peut réaliser de la cryptographie avec des masques graphiques à la place des masques binaires comme le masque jetable.

¹En fait, une implantation effective du masque jetable était le téléphone rouge entre la Maison Blanche et le Kremlin : les deux camps échangeaient par avance des grandes quantités de bits aléatoires grâce à la valise diplomatique et ils consommaient ensuite progressivement ces bits pour chiffrer leurs communications par masque jetable.

2 Théorie

Transposons les bits en termes de masquer ou non la lumière : pour 0, on laisse passer la lumière (le transparent) et pour 1, on bloque la lumière (l'opaque). La base du problème est qu'un tel masque graphique ne permet pas de réaliser de OU exclusifs. En effet, si l'on fait la table de vérité pour ce genre masque (figure 1), on obtient plutôt un OU inclusif.

Une solution proposée par NAOR et SHAMIR est d'utiliser des masques diagonaux. En effet, on voit sur la figure 2 que si l'on superpose deux diagonales identiques, on obtient cette même diagonale (50% d'opacité) alors que si l'on superpose deux diagonales perpendiculaires, les diagonales se complètent pour donner un carré opaque (100% d'opacité). Partant de ce constat, le principe du chiffrement d'une image en noir et blanc est le suivant :

1. Alice et Bob partagent un masque secret de même dimension que l'image à échanger et constitué aléatoirement de diagonales telles que celles de la figure 2.
2. Alice prend l'image qu'elle veut envoyer à Bob, la compare pixel par pixel au masque aléatoire et la transforme selon les règles suivantes :
 - (a) Si le pixel de l'image est blanc, le diagonale de l'image masquée est la perpendiculaire à celle du masque.
 - (b) Si le pixel de l'image est noir, le diagonale de l'image masquée est la même que celle du masque.
3. Alice transmet l'image masquée à Bob.
4. Bob superpose l'image masquée et le masque sur un fond blanc. Le résultat obtenu aura des carrés (100% d'opacité) là où l'image était noire et de diagonales (50% d'opacité) là où l'image était blanche.

Le bon fonctionnement de l'algorithme est basé sur le fait que le cerveau humain est capable naturellement de reconnaître une image en noir et blanc à partir d'une image en noir (100% d'opacité) et disons gris (50% d'opacité). La différence de contraste est au moins assez suffisante pour que l'on puisse reconnaître des formes simples. Un bon exemple est proposé sur le site de Douglas SIMMONS (voir les références) avec le drapeau canadien (figure 3 : la première image est l'original, la seconde est le masque, la troisième est l'image masquée et enfin la quatrième image est le résultat de la superposition de l'image masquée et du masque).

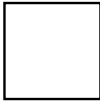











X	Y	X sur Y
		
		
		
		

FIG. 1 – Table de vérité pour le transparent et l'opaque.

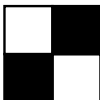





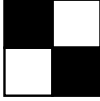
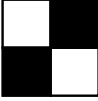

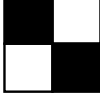
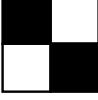
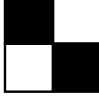
X	Y	X sur Y
		
		
		
		

FIG. 2 – Table de vérité pour les masques diagonaux.

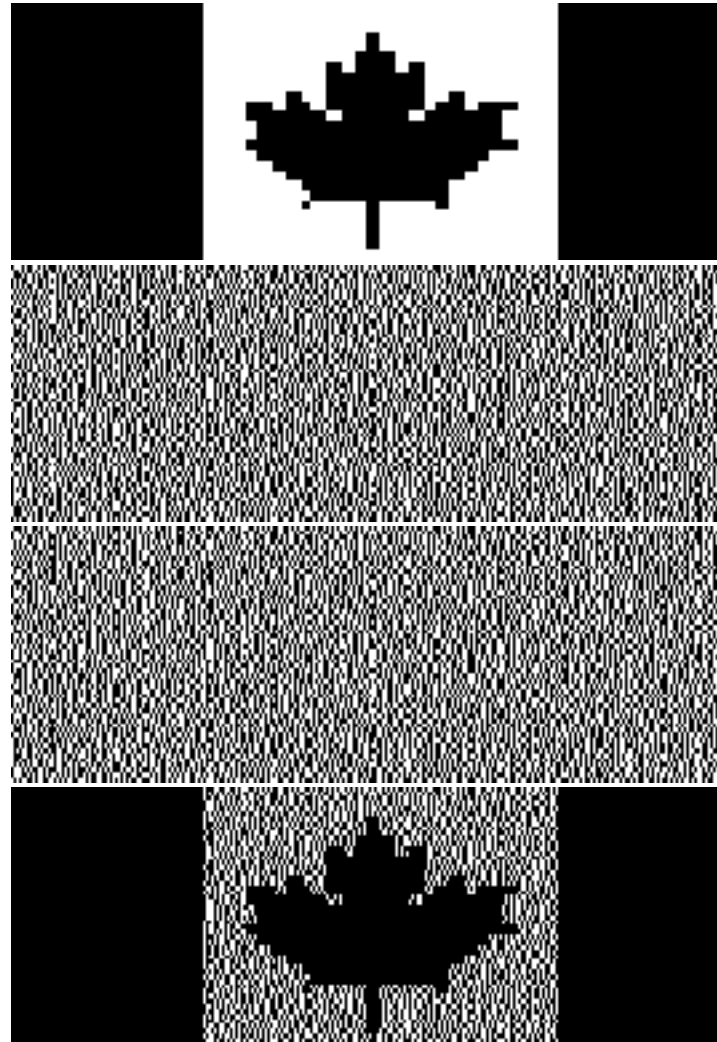


FIG. 3 – Drapeau canadien (par Douglas SIMMONS).

3 Identification

L'idée du projet est d'utiliser la cryptographie visuelle pour faire un système d'identification visuelle. Le principe est que l'utilisateur reçoit un masque. Par la suite, pour identifier l'utilisateur, il suffit de lui soumettre une image masquée en correspondance : si l'utilisateur est vraiment le propriétaire du masque, il peut reconnaître l'image derrière le masque en superposant l'image masquée et le masque alors qu'un usurpateur n'a pas la moindre idée du contenu de l'image originale.

4 Conception du système

4.1 Les drapeaux

La première version du projet était inspirée de l'exemple du drapeau canadien utilisé par Douglas SIMMONS (figure 3). L'idée était la suivante : chaque utilisateur est attribué un identifiant et un masque (imprimé sur un support acétate pour le prototype). Le logiciel garde une liste des pairs (identifiant, masque).

Ensuite, lorsque un utilisateur veut s'identifier, le logiciel commence par demander à l'utilisateur son identifiant. Avec cet identifiant, le système peut retrouver dans sa liste le masque de l'utilisateur. Le système choisit également aléatoirement n drapeaux dans une librairie de représentations de drapeaux à sa disposition. Un des n drapeau (choisit aléatoirement) est affiché masqué et les n drapeau sont affichés normalement. L'utilisateur légitime doit être capable de dire quel drapeau parmi les n proposés est celui représenté par le masque. La probabilité qu'un imposteur devine le bon drapeau par hasard est de $\frac{1}{n}$. Ce principe d'identification est traditionnellement appelé un *challenge* d'identification.

4.2 Génération d'images aléatoires

Le problème de la méthode ci-dessus est que le nombre de drapeau est bien trop limité² (un ordinateur peut tester toutes combinaisons de masque et drapeaux dans un temps raisonnable).

²il y a environ 230 drapeaux nationaux. De plus, certains drapeaux sont identiques en noir et blanc.

La solution à la limitation du nombre de drapeaux est de remplacer les drapeaux par des images générées aléatoirement. Cependant, il ne faut pas non plus que les images soient trop aléatoires (comme par exemple du bruit) pour que l'utilisateur n'ai pas trop de mal à les reconnaître.

Il faut que l'image comporte des formes relativement distinguables mais disposées de façon aléatoires (pour concerver le grand nombre de *challenges* possibles). Une solution pourrait être de générer les images en choisissant une figure géométrique (triangle, carré, pentagone...) et en lui appliquant une rotation aléatoire sur elle-même pour augmenter le nombre de résultats possibles.

Il est intéressant dans ce cas qu'il est souhaitable de favoriser les figures géométriques asymétriques pour que la rotation change le plus possible le résultat final.

4.3 Attaque par répétition

Après avoir implanté la génération d'images aléatoires, il s'est avéré que le système souffre d'une faille de sécurité majeure. Par propriété du OU exclusif, il s'avère que si un attaquant applique l'image masquée sur chacune des images claires proposées comme solution, dans un cas il obtient le masque et dans les autres cas il obtient un résultat non pertinent. L'attaquant ne peut pas directement distinguer le masque des résultats non pertinents mais s'il fait cette manipulation sur plusieurs *challenges*, le masque sera le résultat qui apparaîtra à chaque fois.

La réponse à cette attaque consiste à ne plus masquer une image et demander à l'utilisateur de la reconnaître mais plutôt de mettre $n - 1$ figures géométriques parmi n dans l'image masquée et demander à l'utilisateur quelle est la figure manquante.

4.4 Attaque statistique

Une autre attaque est toujours possible sur le système, bien que plus difficile. En effet, il est probable que le procédé de génération de nombre aléatoire a tendance à générer des images avec toujours une plus forte proportion de pixels blancs ou noirs.

Pour éviter qu'un attaquant puisse profiter de cette propriété du générateur d'images aléatoires, on utilise un dernier bit aléatoire à la fin de la génération de l'image pour décider si l'on fait ou non une négation de l'image générée.

En effet, la négation de l'image transforme des figures noires sur fond blanc en figures blanches sur fond noir, ce qui permet d'alterner la tendance statistique des images générées, rendant ainsi plus difficile l'exploitation de cette tendance par un attaquant.

5 Implantation

Le système de *login* illustrant la méthode présentée précédemment a été réalisée entièrement en Java. Le système se décompose en plusieurs bibliothèques et deux applications dont les principaux détails sont détaillés ci-dessous.

5.1 Bibliothèque graphique

La bibliothèque graphique est destinée à la gestion des images dans le projet. Elle se compose essentiellement de la classe `PortableBitmap` qui implémente les images en noir et blanc au format *Portable Bitmap* tel que décrit dans la page de manuel Unix (`man pbm`).

Cette classe offre également une méthode pour tracer des droites dans l'image avec l'algorithme de BRESENHAM et une méthode pour remplir une zone avec l'algorithme à germe *Flood Fill*.

Le code source de l'algorithme BRESENHAM est tiré de l'implémentation de Kenny HOFF disponible à l'adresse :

<http://wwwx.cs.unc.edu/~hoff/projects/comp235/bresline/src/bresline.c>

Le code source de l'algorithme *Flood Fill* est directement inspiré du pseudo-code proposé en cours.

5.2 Bibliothèque 2D

La bibliothèque 2D contient les éléments de géométrie 2D nécessaire au projet : les points, les vecteurs, les matrices de transformation. La bibliothèque 2D propose aussi une classe *Polygon2D* pour générer rapidement des polygones réguliers (triangle équilatéral, carré...). Une transformation de mise à l'échelle permet de faire un rectangle d'un carré. Pour faire des cercles, on se contente de polygones réguliers avec beaucoup de côtés.

La programmation de ces classes est largement inspirée du code source fourni avec les TP d'infographie. Le principal ajout est que les polygones

savent se dessiner sur une image de la librairie graphique. De plus, il s'est avéré utile d'ajouter un centre aux polygones afin de pouvoir facilement placer le germe de l'algorithme *Flood Fill* si l'on veut remplir un polygone.

5.3 Librairie cryptographique

La librairie cryptographique contient essentiellement un générateur de masques aléatoires et un générateur d'image aléatoires correspondant aux spécifications décrites précédemment.

De plus, la librairie cryptographique permet de faire des opérations logiques sur les masques (NON, ET, OU inclusif, OU exclusif...).

5.4 Générateur de clés

Le générateur de clés est une application en ligne de commandes destinée à l'administrateur du système. Elle permet de générer un masque d'identification pour un utilisateur.

L'utilisation du programme est très simple. La commande est la suivante :

```
# java -jar KeyGen.jar id
```

Le programme crée alors dans le répertoire courant deux fichiers : 'id.key' dont le format est destiné à être lu par l'application principale et une image 'id.pbm' qui est destinée à être imprimée sur un support transparent et donnée à l'utilisateur.

5.5 Application principale

L'application principale est celle qui permet à un utilisateur du système de s'identifier en utilisant le masque imprimé qui lui a été fourni.

L'application est lancée avec la commande suivante :

```
# java -jar MaskLogin.jar
```

Ensuite, la fenêtre de l'application (figure 4 à la fin du rapport) apparaît et l'utilisateur saisit son identifiant dans le champs prévu à cet effet. L'application cherche alors dans son répertoire courant d'exécution le fichier 'id.key' correspondant et génère un *challenge* en conséquence (figure 5). L'utilisateur plaque son masque (figure 6) sur l'écran et il observe alors le *challenge* en clair (figure 7), ce qui lui permet de choisir parmi les figures proposées celle qui est absente du *challenge* (figure 8).

6 Améliorations

6.1 Comparaison avec les mots de passe

Le système développé dans ce projet à plusieurs avantages par rapport à un simple système par mot de passe. Tout d'abord, il y a bien plus de masques possibles (2^{3750} pour la taille de masque utilisée dans l'application) que de mots de passe possibles (environ 2^{120} mots de passe de moins de 20 caractères et sensibles à la casse).

Par contre, il est important de noter qu'un seul *challenge* ne suffit pas car dans l'application, un attaquant à 1 chance sur 7 de trouver la bonne réponse par hasard. La probabilité de succès d'un attaquant à n *challenges* étant $\frac{1}{7^n}$, au moins 3 *challenges* sont nécessaires pour un minimum de sécurité (seulement 1 chance sur 343 de réussir par hasard).

Enfin, un avantage des mots de passe est que, à moins d'une négligence de l'utilisateur, on ne peut pas perdre ou se faire voler son mot de passe, contrairement à un objet physique tel que le masque.

6.2 Comparaison avec les cartes à puce

La carte à puce est un système d'identification particulièrement efficace. Toutefois, la fabrication de cartes à puces est coûteuse alors que l'impression de masque ne l'est pas. Les masques peuvent donc être utilisés par des petites entreprises cherchant un moyen d'identification légèrement supérieur aux mots de passe mais tout de même bon marché.

En toute honnêteté, il faut rappeler que la sécurité de la méthode de cryptographie visuelle est basée sur celle du masque jetable. Pour conserver la sécurité théoriquement parfaite du système, il faudrait donc utiliser une seule fois chaque masque. Disons qu'un bon compromis dans la pratique serait de changer souvent les masques des utilisateurs, ce qui peut rendre le procédé un peu plus coûteux.

6.3 Limitations technologiques

Une amélioration évidente par rapport au prototype serait d'imprimer un masque plus précis. Cela demande une imprimante laser plus précise mais cela n'a rien d'impossible. La taille des pixels des écrans est également un

facteur limitant mais de toutes façon l'oeil humain ne permet pas d'utiliser confortablement des masques beaucoup plus petits.

Dans la pratique, un problème inattendu dans la réalisation du prototype a été la calibration de la taille des carrés imprimés sur le masque pour qu'ils correspondent avec les pixels de l'écran utilisé pour la démonstration. En effet, la taille des pixels varie sensiblement d'un écran à un autre, d'une station de travail à un ordianteur portable et même parfois d'un endroit à l'autre du même écran !

7 Conclusion

La cryptographie visuelle est un domaine en pleine émergence et dont les applications restent encore à explorer. En effet, la cryptographie moderne repose sur des fonctions mathématiques que les ordinateurs savent calculer rapidement mais qu'ils ne savent pas inverser (par exemple, la multiplication de nombres premiers est rapide alors que la factorisation d'un nombre en nombres premiers est très difficile).

L'idée de la cryptographie visuelle est d'exploiter le fait que la reconnaissance d'image est difficile pour un ordinateur alors que le cerveau humain le fait tellement rapidement que l'on en est à peine conscient. Peut-être serait-il intéressant de réfléchir à d'autres domaines que l'infogprahie où les qualités du cerveau humain pourraient permettre d'améliorer un alogirthme ? Il s'agirait en quelque sorte d'algorithmes interactifs où l'humain et l'ordinateur se relaient pour réaliser une tâche.

8 Références

La base théorie derrière la cryptographie visuelle est clairement expliquée sur le site de Douglas SIMMONS :

<http://www.cacr.math.uwaterloo.ca/~dstinson/visual.html>

La documentation de Java s'est également avérée très utile :

<http://java.sun.com/j2se/1.4.1/docs/api/>

Les articles utilisés pour ce projet sont listés sur la page de bibliographie suivante.

Références

- [1] M. Naor and A. Shamir. Visual cryptography. In Alfredo De Santis, editor, *Advances in Cryptology - EuroCrypt '94*, pages 1–12, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science Volume 950.

9 Saisies d'écran

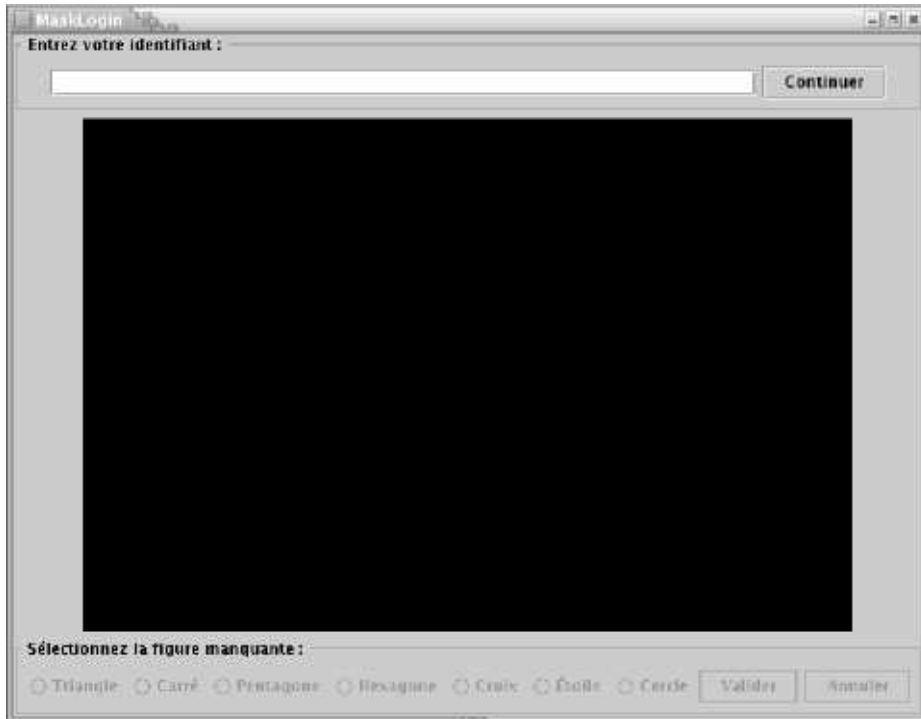


FIG. 4 – Application principale.

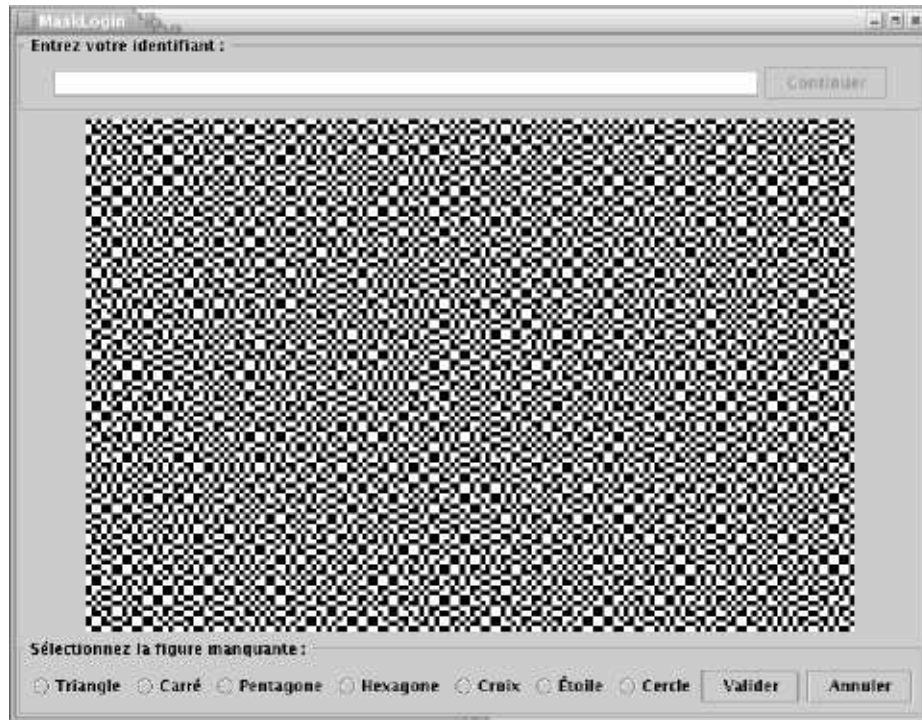


FIG. 5 – Un *challenge*.

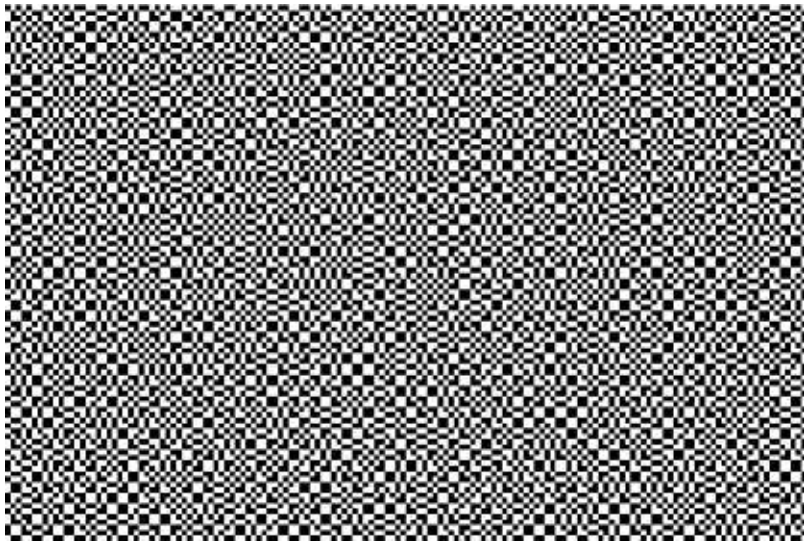


FIG. 6 – Le masque.

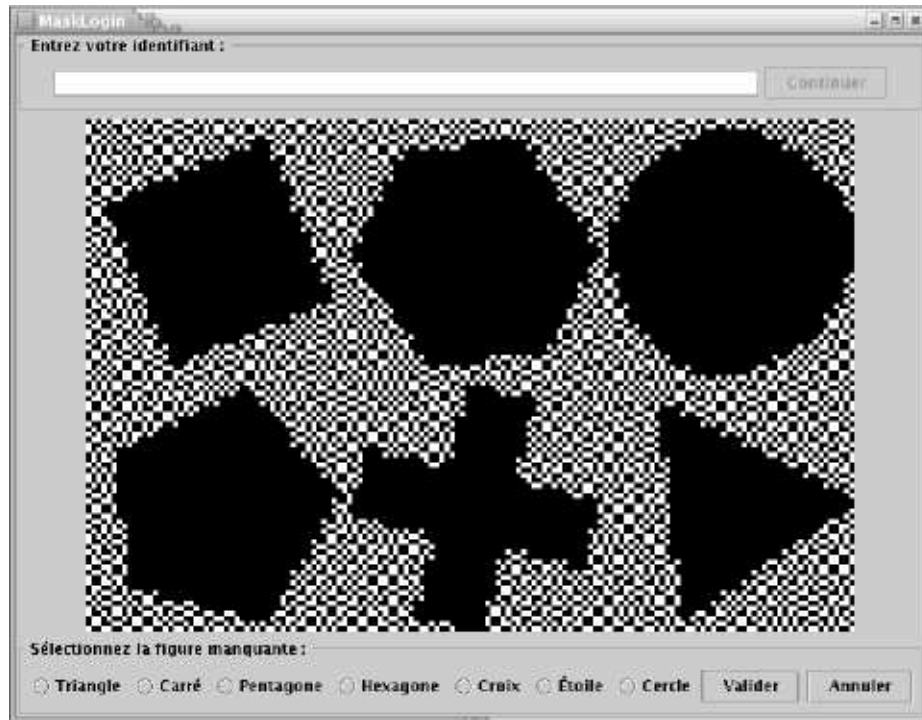


FIG. 7 – La superposition du masque sur le *challenge*.

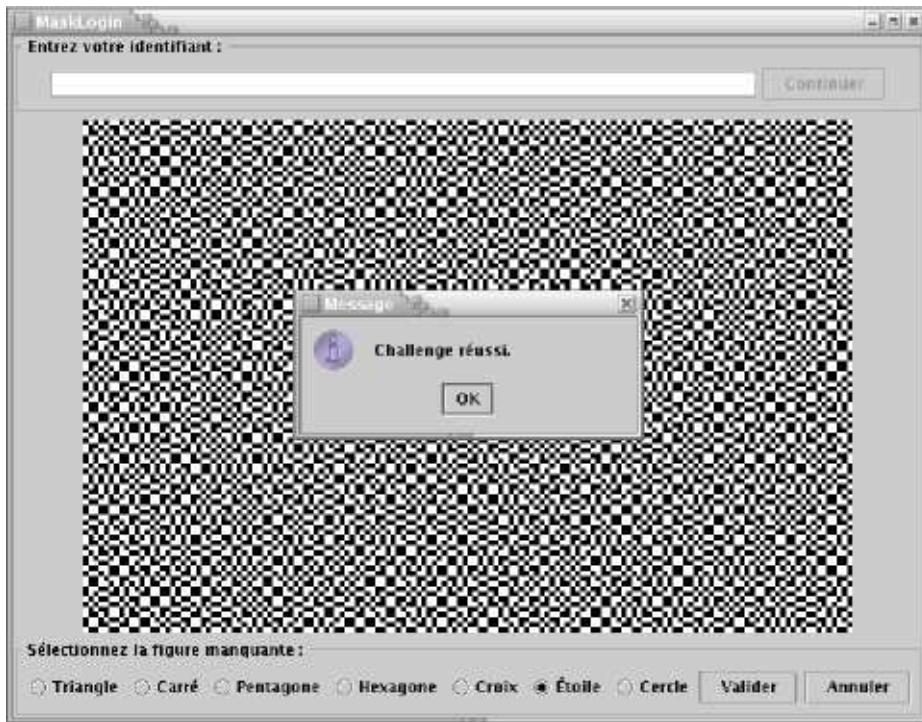


FIG. 8 – La validation.