Content Privacy and Access Control on Image-Sharing Platforms

Kun He

ANNÉE 2017

CentraleSupélec

UNIVERSITE BRETAGNE LOIRE

N° d'ordre : XXXX-XX-TH

THÈSE / CENTRALESUPÉLEC
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE CENTRALESUPÉLEC

Mention : Informatique

Ecole doctorale 601 « Mathématiques et Sciences et Technologies de l'Information et de la Communication (MATHSTIC)»

présentée par

Kun HE

Préparée à l'UMR 6164 - IETR (Equipe SCEE)
Institut d'Electronique et de Télécommunications de Rennes

Content Privacy and Access Control on Image-Sharing Platforms

Thèse soutenue à IRT b<>com le 25 Septembre 2017

devant le jury composé de :

Patrick BAS
Responsable de l'équipe SIGMA, École Centrale de Lille / *rapporteur*

William PUECH
Responsable de l'équipe ICAR, LIRMM, Université Montpellier / *rapporteur*

Azza OULED-ZAID
Full professor, Université de Tunis EL Manar Institut Supérieur d'Informatique, Département ASR / *examinateur*

Christophe BIDAN
Responsable de l'équipe CIDRE, CentraleSupélec Campus de Rennes / *directeur de thèse*

Gaëtan LE GUELVOUIT
Responsable du Laboratoire DTI, IRT b<>com / *co-directeur de thèse*

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse, Christophe Bidan, pour ton soutien, ta patience et ton aide au quotidien. Grâce à tes multiples conseils, que ce soit professionnels ou personnels, j'ai pu finaliser cette thèse avec succès. Merci également à mon co-directeur de thèse, Gaëtan Le Guelvouit, de m'avoir proposé après mon stage, une thèse sur ce sujet vraiment intéressant. C'est par ton soutien au quotidien que j'ai pu jour après jour me familiariser avec mon sujet de recherche.

Je tiens ensuite à remercier l'ensemble des membres du jury d'avoir accepté de participer à ce jury. Un merci tout particulier à Patrick Bas et William Puech d'avoir accepté d'être les rapporteurs de cette thèse.

Je tiens à remercier l'ensemble des membres de l'équipe DTI d'IRT b<>com, pour ces 3 années passées à travailler avec vous, qui resteront un excellent souvenir pour moi. Je remercie tout particulièrement Valérie qui fut comme une grande sœur pour moi, Gustav, made by IKEA, pour l'animation de l'équipe et tes talents de pâtissier pour "princess cake", Alexandre, le plus jeune des ingénieurs toujours souriant avec ta bonne humeur, et finalement, Emilie, la meilleure chef de projet.

Merci à tous les bcomiens et bcomiennes, vous êtes une grande famille harmonieuse, vous êtes tous très amicaux et chaleureux, tout particulièrement Stéphane, Loïc, Etienne et ma stagiaire Cyrielle, qui a beaucoup contribué pour la partie développement d'application dans ma thèse.

Merci aussi à tous mes amis, surtout ma copine Chi qui m'a accompagné tout au long de mon séjour à Rennes, durant lequel on a pu rire, pleurer, faire shopping, voyager ensemble. Merci à mon copain Benjamin d'avoir pris soin de moi et de m'avoir accompagné durant les jours difficiles. Ton optimisme et ta force m'ont donné le courage et ensoleillent ma vie au quotidien. Merci pour ton aide pour le français et la partie cryptographie de ma thèse. Il est difficile de trouver les bons mots pour te remercier, et d'exprimer ce que mon cœur ressent.

Je tiens à présent à remercier ma famille pour votre aide et votre soutien. Merci à mes parents pour tout ce que vous avez fait pour moi. Sans vous je ne serai pas la femme que je suis aujourd'hui. Merci pour votre soutien de l'autre bout du monde, qui m'a permis d'avoir la persévérance nécessaire pour finir mes études et vivre en France loin de vous. Je vous aime.

Résumé

0.1 Contexte et motivation

De nos jours, les réseaux sociaux facilitent de plus en plus les relations sociales malgré la distance séparant les utilisateurs. Ainsi, les personnes se sont habituées à partager leurs vies, leurs humeurs, leurs opinions avec leurs amis ou leurs familles sur les réseaux sociaux, en échangeant des photos, des vidéos, leurs statuts, des liens hypertextes, etc. A travers cette aisance dans le partage d'informations personnelles, un problème de sécurité, plus particulièrement de confidentialité, est mise en lumière. En effet, sans protection, tout le monde peut accéder aux publications des utilisateurs et ainsi apprendre des informations sur la vie privée des utilisateurs. Ainsi, la confidentialité des données de l'utilisateur peut être violée, par exemple, les photos des utilisateurs peuvent être ainsi détournées ou être propagées malicieusement. Bien sûr, les fournisseurs de réseaux sociaux essaient d'aider de plus en plus les utilisateurs à maitriser leurs contenus afin de protéger leurs vies privées. La plupart des réseaux sociaux permettent aux utilisateurs de regrouper leurs contacts par catégorie en fonction du type de relations qu'ils ont avec, tels que "Amis", "Famille", "Connaissances" etc. Ainsi, en spécifiant quel catégorie de contacts peut accéder à tel ou tel contenu, les utilisateurs peuvent penser contrôler la diffusion de leurs information personnelles, et par conséquent que leurs vies privées est garantie. Cependant, de notre point de vue, la confidentialité des données personnelles des utilisateurs n'est pas garantie, car les fournisseurs des réseaux sociaux ont accès librement à l'ensemble des données publiées. Par conséquent, notre objectif est de fournir une méthode garantissant le respect de la vie privée de l'ensemble des utilisateurs vis-à-vis à la fois des autres utilisateurs mais aussi du fournisseur de réseaux sociaux.

Une des solutions consiste à créer un nouveau réseau social permettant le partage de contenus (images, vidéos etc.) tout en garantissant la vie privée de l'utilisateur. Au sein d'un tel réseau, l'ensemble des contenus utilisateurs seraient chiffrés et seules les personnes autorisées pourront obtenir la clé nécessaire pour déchiffrer et obtenir les contenus. Créer un autre réseau social, indépendante de l'existant, est la solution la moins contraignante au niveau de la compatibilité. De tels réseaux sociaux ont déjà été proposés, e.g. SuperNova [SD12], Diaspora [BHG+12], PeerSoN [BSVD09], Safebook [CMS09], mais aucun d'entre eux n'est vraiment utilisé aujourd'hui. La raison principale est certainement que les utilisateurs préfèrent encore utiliser les réseaux sociaux qu'ils ont l'habitude d'utiliser (e.g. Facebook, Twitter, Google+ etc.) au lieu de ces nouveaux réseaux sociaux qui ne sont utilisés que par quelques utilisateurs seulement. Par conséquent, nous avons décidé de ne pas créer un nouveau réseau social, mais plutôt de proposer une solution qui permettrait de protéger la vie privée des utilisateurs sur les réseaux sociaux couramment utilisés (e.g. Facebook, Twitter, Google+ etc.).

Afin de protéger la vie privée des utilisateurs, nous affirmons que le contenu publié ne doit pas être lisible. Par conséquent, notre objectif est de proposer une solution permettant à chaque utilisateur de chiffrer le contenu avant de les publier, et de lui permettre de partager la clé de déchiffrement uniquement avec les personnes qu'ils autorisent à accéder au contenu. Ainsi, si l'algorithme de chiffrement utilisé garanti la confidentialité des données chiffrées, les personnes n'ayant pas la clé de déchiffrement (y compris les fournisseurs de la plateforme de partage d'images) ne peuvent pas connaître les contenus publiés.

Dans le cadre de notre solution de chiffrement pour protéger la vie privée des utilisateurs, nous devons nous assurer que la version chiffrée des contenus puissent être publiés sur le réseau social. Une étude publiée sur eMarketer en mars 2014 [soc] a montré la proportion des différents types de

contenu affichés sur Facebook dans le monde entier : les photos représentent 75 % des contenus publiés, les liens hypertextes 10 %, les statuts personnels 6%, les vidéos 4%, et finalement les autres types de contenus représentent moins de 1%. Parmi les principaux types de contenus publiés par la plateforme Facebook, les liens hypertextes et les statuts personnelles sont sous forme de texte. La plupart des algorithmes de chiffrement existants peuvent être directement utilisés pour chiffrer ce type de contenu, et ainsi obtenir un résultat chiffré sous format texte. Pour cela il suffit de coder le texte en base64. Malheureusement, aujourd'hui, le contenu publié par les utilisateurs est souvent de type multimédia (images, vidéos). Pour les images et les vidéos, leurs formats étant différent (e.g. JPEG ou MP4), s'ils sont chiffrés de manière classique, la confidentialité de l'image peut être assurée, mais le résultat obtenu est un fichier binaire, et non une image ou une vidéo. La publication d'un fichier binaire n'est pas autorisé par les réseaux sociaux. De toute évidence, l'utilisation des algorithmes de chiffrement traditionnels n'est pas adaptée à notre contexte d'utilisation. Ainsi, nous devons fournir de nouvelles technologies de chiffrement. Étant donné que la proportion de photos échangés est sensiblement (18 fois) plus grande que celle des vidéos par fois, il est donc primordial de se focaliser sur la protection des photos publiées plutôt que des vidéos publiées.

Lorsque nous publions une image sur n'importe quel réseau social, de nombreux formats sont acceptés, tels que JPEG, PNG, GIF, etc. Le format JPEG est le format le plus utilisé pour stocker et compresser des images. Les dispositifs de capture d'images, tels que les caméras numériques, utilisent le format JPEG pour stocker les images. Les images disponibles sur le Web sont également compressées sous le format JPEG. Parce que le format JPEG est un format de compression à perte, nos recherches ont montré que après avoir publié une image en format JPEG sur une plateforme, celle-ci est normalement compressée par la plateforme, ce qui provoque la modification des coefficients d'image, de sorte que l'image obtenue après déchiffrement a une qualité inférieure de l'image original. Quand à lui, le format PNG est un format de compression sans perte, et ne pose donc pas de problèmes de qualités lors du chiffrement et déchiffrement de l'image. Dans cette thèse, nous concentrerons nos efforts sur les images JPEG et les problématiques relatives de perte de qualité.

0.2 Objectifs de recherches

L'objectif de cette thèse est de proposer une infrastructure indépendante de gestion de contenu afin de contrôler l'accès aux images d'utilisateurs publiées sur les réseaux sociaux existants (considérés comme plateformes de partage d'images) tout en assurant leur confidentialité. Pour atteindre cet objectif, l'idée principale est de permettre aux utilisateurs de chiffrer leurs images et de publier les images chiffrées sur n'importe quelle plateforme de partage d'images existante (e.g. Facebook, Flickr, Twitter, Google+, etc.). Les clés de chiffrement ainsi que les règles de contrôle d'accès des images chiffrées sont stockées dans une infrastructure indépendante. Lorsque quelqu'un accède à une image chiffrée publiée et souhaite accéder à l'image en clair, il doit demander la clé de déchiffrement à notre infrastructure indépendante. Ainsi, la clé de déchiffrement est renvoyée uniquement si les règles de contrôle d'accès l'autorisent. Notez que l'application des règles de contrôle d'accès peut également être utilisée afin de respecter et mettre en œvre le droit à l'oubli.

Après avoir réalisé un état de l'art du domaine, nous nous sommes rendus compte que trois challenges majeures sont à prendre en compte afin de proposer un algorithme de chiffrement d'images JPEG :

- Le premier challenge consiste à définir un algorithme de chiffrement qui conserve le format d'images. Beaucoup de formats d'image (e.g. JPEG, PNG, GIF etc.) sont acceptés par les plateformes de partage d'images existantes. Cependant, selon notre expérimentation (mentionnée dans §4.1), dans la plupart des plateformes, les images mises en ligne sont systématiquement compressées au format JPEG, peu importe leurs formats d'origine. Dans ce contexte, l'algorithme de chiffrement proposé doit préserver le format JPEG [1].

- Le deuxième challenge se focalise sur la sécurité de l'algorithme de chiffrement. Ce algorithme doit garantir que n'importe quel adversaire ne puisse pas obtenir le moindre bit d'information

[1] Notez que le format JPEG est le format le plus utilisé pour le stockage et la compression d'images.

(notamment sur la luminosité de l'image initial) sur une image à partir de la version chiffrée de celle-ci. Par conséquent, la propriété de sécurité d'indistingabilité IND-CPA sera requise pour la solution choisie.

- Le troisième challenge repose sur la compatibilité de l'algorithme de chiffrement avec la majorité des plateformes de partage d'images existantes (e.g. Facebook, Flickr, Twitter, Google+, etc.). En effet, quelques plateformes effectuent un post-traitement des images (e.g. re-compression des images) selon notre expérimentation. Donc, l'algorithme de déchiffrement doit permettre de retrouver les images en clair malgré les différents pré-traitements qui ont été appliqués sur les images chiffrées.

0.3 Contributions

Dans un premier temps, nous avons défini un algorithme de chiffrement qui conserve le format JPEG et qui respecte la propriété de sécurité IND-CPA. En particulier, nous avons proposé trois schémas de chiffrement d'image JPEG qui utilisent le même algorithme de chiffrement, ce dernier étant intégré à différentes étapes du processus de compression JPEG. Nous avons prouvé la sécurité de ces trois schémas de chiffrement. Nous avons ensuite comparé les images obtenus après déchiffrement avec chacun des algorithmes. Cela nous a permis de sélectionner le meilleur algorithme en fonction de la qualité des images obtenues.

Sur le algorithme obtenu, nous nous sommes ensuite focalisés sur la conversation des formats d'images après chiffrement, et la faisabilité de la publication des images chiffrées. En effet, toutes les images chiffrées doivent être acceptées comme des images JPEG par n'importe quelle plateforme de partage d'images. Nous avons ainsi utilisé notre algorithme de chiffrement pour chiffrer les images JPEG que nous avons ensuite mis en ligne sur huit plateformes de partage d'images largement utilisées (Facebook, Flickr, Pinterest, Google+, Twitter, Instagram et deux plateformes de partage d'images chinoises: Weibo et Wechat). Ainsi, nous avons pu vérifier la faisabilité de la publication des images chiffrées sur différentes plateformes de partage d'images.

Dans un troisième temps, nous avons vérifié la qualité des images obtenus après déchiffrement sur un bon nombre de plateformes de partage d'images. Pour cela, sur chacune des plateformes testées, nous avons téléchargé et déchiffré les images, et avons comparé la qualités des images déchiffrées avec les images originales. Nous avons observé que pour les plateformes Flickr, Pinterest, Google+ et Twitter, l'algorithme de chiffrement maintenait une qualité élevée des images aprés déchiffrement. Malheureusement, concernant les plateformes Facebook, Instagram, Weibo et Wechat, nous avons remarqué que les images obtenues après déchiffrement avaient une qualité extrêmement médiocre comparée aux originales. En analysant les résultats, nous avons découvert que ces plateformes de partage d'images effectuent un post-traitement pour toutes les images mises en ligne. Par conséquent, nous avons modifié l'algorithme de chiffrement afin d'obtenir un nouveau algorithme de chiffrement compatible avec Facebook, Weibo et Wechat, qui conserve une bonne qualité des images aprés déchiffrement.

Notre quatrième contribution est une infrastructure indépendante de gestion de contenu qui autorise les utilisateurs à utiliser notre algorithme de chiffrement pour publier leurs images sur Facebook tout en spécifiant les régles de controle d'accés pour ses images. Nous avons ainsi développé une application nommée *PixGuardian* qui permet aux utilisateurs de chiffrer et publier des images. Il permet également de générer, de distribuer et de gérer les clés de chiffrement et déchiffrement pour les utilisateurs, et permet aux utilisateurs de définir les conditions d'accès des images publiées. Ensuite, avec notre plugin du navigateur Chrome, les utilisateurs autorisés peuvent déchiffrer les images. Enfin, nous avons évalué l'acceptabilité de PixGuardian en étudiant son utilisation par 33 utilisateurs pendant six semaines.

0.4 Travaux à venir

Dans notre troisième contribution, on a proposé l'algorithm amélioré, mais nous devons régler certains paramètres de l'algorithme pour chacune plateformes de partage d'images, car ces paramètres

dépendent des caractéristiques spécifiques de chacune d'entre elles. C'est par conséquent, un inconvénient de notre algorithme de chiffrement, vus que certains paramètres de l'algorithme doit être instancié plus ou moins différemment selon la plateforme considérée. Pour autant, ces réglages permettent de nous assurer du maintien de la bonne qualité des images après déchiffrement. Par conséquent, nous voulons continuer à améliorer l'algorithme pour le rendre plus générique. Par exemple, nous pouvons préparer des paramètres pour un petit ratio de quantification. Ces paramètres pourront être appliqué à de plus grand ratio de quantification, mais une perte de la qualité des images déchiffrées pourra être observé. De cette façon, nous pouvons utiliser notre algorithme de chiffrement sur plus de plateformes de partage d'images sans modifier les paramètres de l'algorithme.

Nous rappelons que notre algorithme a pour but de protéger la vie privée des utilisateurs vis à vis de leurs contacts, i.e. les utilisateurs qui sont connectés à leurs profils. En utilisant notre application, les utilisateurs peuvent définir les conditions d'accès à leurs images, et les personnes autorisées à accéder aux images, peuvent utiliser, en particulier notre plugin pour accéder aux images originelles en clair publiées sur Facebook via le navigateur Chromo. Cependant, nous ne pouvons pas faire confiance aux contacts complètement. En effet, ils peuvent via des captures d'écran des images en clair, partager ces images à des personnes non autorisées. Nous n'avons actuellement aucun moyen d'empêcher cela, ni même de savoir si quelqu'un à réaliser un tel partage. Pour remédier à cela, nous pourrions intégrer un tatouage numérique dans les images, ce qui nous aiderait à tracer les utilisateurs qui auraient retransmis les images et pouvoir agir en conséquent.

Au départ, notre objectif était de protéger le contenu publié sur les réseaux sociaux. Suivant l'étude effectué par eMarketer [soc], nous avons pris connaissance du pourcentage des différents types de contenu publiés sur les pages Facebook dans le monde entier. Nous proposons d'utiliser un procédé de chiffrement pour protéger la vie privée de l'utilisateur et nous devons veiller à ce que les résultats chiffrés puissent être publiés sur le réseau social en assurant que les résultats gardent un format correct. Parmi ces principaux types de contenu publiés, les images sont plus largement publiées sur les réseaux sociaux que les vidéos, nous nous sommes concentrés dans cette thèse, sur comment protéger les images publiées en priorité. Dans un futur proche, un algorithme de chiffrement pouvant protéger la vie privée de l'utilisateur lors de la publication de la vidéo sur les réseaux sociaux ou autres plateformes de partage de vidéos, sera proposé. De nos jours, de nombreuses technologies de tatouages numériques sont proposées afin de garantir que les droits d'auteurs des utilisateurs qui publient des contenus ne sont pas violés. Mais même si les droits d'auteurs des utilisateurs sont respectés, le respect de leurs vies privées n'est pas nécessairement garanti car les contenus publiés peuvent toujours être visibles. L'algorithme de chiffrement qui sera utilisé pour protéger les contenus vidéo, devra pouvoir gérer différents formats (e.g. MPEG, HEVC), résolutions etc., et plus particulièrement la compression à perte. Nous testerons également d'insérer le chiffrement à différentes étapes du processus de compression (soit dans un domaine temporel, soit dans un domaine fréquentiel). La solution qui garantira la meilleure conservation de qualité des vidéos traitées, sera choisie. Si nécessaire, nous ajouterons également le mécanisme de correction lors du déchiffrement après avoir téléchargé la vidéo chiffrée.

Après avoir recueilli les commentaires des utilisateurs qui ont testé notre solution PixGuardian, différentes observations nous sont parvenues. Nous avons notamment remarqué que les utilisateurs préfèrent partager des images à partir de leur smartphones directement plutôt que de transférer les images sur leurs ordinateurs avant de les publier sur les plateformes. Nous pensons que cette préférence est non négligeable, car les gens aiment prendre des photos en utilisant leur smartphones, et il est en effet plus commode de les publier directement via ce support. La plupart des plateformes de partage d'images ont déjà une version sur smartphone, et meme certaines plateformes telles que Instagram et Snapchat ne sont disponibles que sur smartphones encore. Par conséquent, notre solution de chiffrement d'images permettant d'assurer leurs confidentialités après publication, doit etre compatible avec une utilisation sur smartphone. Pour cela, il nous faudra considérer la taille de l'écran des différents modèles de smartphone, faire des tests pour savoir comment compresser et rogner une image publiée, afin de proposer un algorithme de chiffrement et une application similaire à PixGuardian, approprié pour smartphone.

Contents

Chapter 1

Introduction

Contents

1.1 Background and motivation

The social networks facilite social relationships despite the distance separating users. Thus, people have become accustomed to share their lives, moods, opinions with their friends or their family on social networks, thanks to the photos, videos, status, links, etc. But, this sharing raises some security issues, especially with respect to users' privacy. Indeed, without protection, everybody can access the users' publications and thus potentially learns information related to the users' privacy. Of course, the social network providers allow users to control the access to their contents by grouping their friends according to the relations they have with, such as "Friends", "Family", "Acquaintances", etc. By allowing users to specify the friends that can access to each content, this access control gives them the impression that their privacy is guaranteed. However, from our point of view, the privacy of the users is not guaranteed with respect to the social network providers since they can clearly know any published contents. Therefore, we aim to provide a method to protect user's privacy completely, such that only the user himself and the authorised friends can access to the published contents.

One solution could be to generate a social network that guarantees user's privacy. In such a network, all the users' contents would be encrypted, and only the authorised people would be able to get the key to decrypt the contents. A number of such social networks has already been proposed SuperNova [SD12], Diaspora [BHG+12], PeerSoN [BSVD09], Safebook [CMS09], but none of them is really used today. The main reason is surely that users prefer to still using the social networks they are accustomed to use (i.e., Facebook, Twitter, Google+ ...) instead of these new privacy-aware social networks used by only a few users. Given that, we have decided to not create a new social network, but to propose a solution that allows to protect the users's privacy on the common widely used social networks (i.e., Facebook, Twitter, Google+ ...).

In order to protect the user's privacy, we assert that the uploaded contents should not be readable by the social network provider. Therefore, our intent is to allow each user to encrypt the contents before publishing them, and to allow only authorized friends to access to the decryption key. In this way, if the encryption algorithm is secure, people who do not have the key (including the social network provider) can not know the published contents.

However, to be compatible with existing social networks, our solution has to ensure that the encrypted contents are accepted for publication, i.e., they have the right format. For contents in text format, this can easily be ensured: we only have to encode the encrypted text in Base64.

Unfortunately, today, the users' contents are often multimedia contents. Thus, a research published on eMarketer in March 2014 [soc] showed the percentage that types of content posted on Facebook pages worldwide: images account for 75% of content posted, then links account for 10%, status for 6%, and videos for 4%. Of course, the images and videos have specific formats (i.e., JPEG and MP4), and if we encrypt them using a traditional cipher, the encrypted result is no longer an image or a video but a binary file. And when we try to publish a binary file, current social networks do not accept such a publication. A solution for that problem is to use an encryption algorithm that preserves the image/video format after encryption. Note that, according to the eMarketer study, the proportion of the images is 18 times as much as the proportion of the videos. Therefore, its seems to us more priority to propose such a solution for images than videos.

When we publish an image on any social networks, there are many formats can be accepted, such as JPEG, PNG, GIF, etc. JPEG is the most widely used standards for storing and compressing image among them. Digital cameras and other image capture devices use JPEG standard to store image. Images on the World Wide Web are compressed by using JPEG standard as well. According to our research (mentioned in §4.1), because JPEG is a lossy compression format, after publishing the JPEG image on the social networks, the image is normally compressed by the platforms, that causes the changes of the coefficients of image, so that the encrypted image cannot be decrypted correctly. Compare with PNG, the format which is a lossless compression format, and does not cause problems that the published images cannot be decrypted correctly, there are more research significance for the format JPEG.

1.2 Research objective

The objective of this thesis is to propose an independent content management infrastructure to control access to and ensure privacy of the users' images published on the existing social networks (viewed as image-sharing platforms). To achieve this objective, the main idea is to allow users to encrypt their images, and publish the encrypted images on any existing image-sharing platforms (i.e., Facebook, Flickr, Twitter, Google+, ...). The encryption keys as well as the access control rules of encrypted images are stored in our independent infrastructure. When someone accesses some published encrypted image, he/she has to request the encryption key to our independent infrastructure, the encryption key being sent back only if the access control rules authorise it. Notice that the enforcement of the access control rules can also be used to implement the right to be forgotten.

The definition of such a solution raises three challenges:

- The first challenge is the definition of an encryption algorithm that preserves the image format. Many image formats are accepted by existing image-sharing platforms, such as JPEG, PNG, GIF, etc. However, according to our experimentation (mentioned in §4.1), in most of platforms, the uploaded images are systematically compressed in JPEG format, no matter what are their original formats. In this context, the proposed encryption algorithm has to preserve the JPEG format [1].

- The second challenge is that the encryption algorithm must be secure. Especially, we require that the adversary should not get any information of the plaintext image from the encrypted image. In particular, he/she cannot distinguish the encryption of a brighter image from a darker image. Therefore, IND- CPA secure is our judgment criteria for the security of image encryption algorithm.

- The third challenge is that the encryption algorithm has to be compatible with the majority of existing image-sharing platforms (i.e., Facebook, Flickr, Twitter, Google+, ...). However, according to our experimentation (mentioned in §4.1), some of the existing sharing platforms performs a post-processing of the published images. Thus, the proposed encryption algorithm has to allow to decrypt the encrypted image even if it has been post-processed.

[1] Notice that JPEG is the most widely used standards for storing and compressing images.

1.3 Contributions

Our first contribution is the definition of an encryption algorithm that preserves the JPEG format and that is IND-CPA secure. More specifically, we have proposed three JPEG image encryption schemes that use the same encryption algorithm but are integrated into different steps of JPEG compression process. We have proved the security of these encryption schemes and the comparison of the decrypted images allowed us to select the best scheme with respect to the quality of the resulted images.

Our second contribution is that all of the encrypted images are accepted as JPEG images by any image-sharing platform. We have used our encryption algorithm to encrypt JPEG images and we have uploaded the encrypted images on eight widely used image-sharing platforms (Facebook, Flickr, Pinterest, Google+, Twitter, Instagram and two Chinese image-sharing platforms: Weibo and Wechat). Thus, we have checked that all of the encrypted images have been accepted by these image-sharing platforms as having a correct image format.

Our third contribution is that our encryption scheme is compatible with the most of images-sharing platforms. We have downloaded and decrypted the images, and we have compared the decrypted images with the original ones. We have thus observed that for Flickr, Pinterest, Google+ and Twitter, the recovered images had a high quality. On the opposite, for Facebook, Instagram, Weibo and Wechat, we have noticed that the recovered images had an extremely poor quality. By analysing the results, we have discovered that these image-sharing platforms performs a post-processing for any uploaded images. Given that knowledge, we have been able to improve the encryption algorithm so as to obtain a new encryption scheme that is compatible with Facebook, Weibo and Wechat.

Our fourth contribution is an independent content management infrastructure that allows users to use our encryption algorithm to publish their images on Facebook while specifying the access control rules for these images. We have developed an application named PixGuardian to help the users to encrypt, to publish and to decrypt images. PixGuardian also allows to generate, distribute and manage the encryption keys for the users. It also permits users to define the access conditions for their images. Finally, we have evaluated the acceptability of PixGuardian by studying its use by 33 users during six weeks.

1.4 Organisation of this thesis

This thesis is divided into 7 chapters. Chapter 2 firstly introduces the fundamentals of two technical domains needed for our goal namely modern cryptography and image representation and compression. In the modern cryptography, we introduce the pseudo-random number generators which are the basis of our encryption algorithm, then we introduce the symmetric encryption which are the type of our encryption algorithm, and then the asymmetric encryption and the relation between these two types which are used in the key management in our solution. We also present the security of cryptographic algorithm especially the notion of IND-CPA that we hope our algorithm can provide. In the image representation and compression, we briefly introduce RGB, YCbCr color model, and the two-dimensional discrete cosine transform (2D-DCT), which are all used in JPEG compression. Because in this thesis, we focus to study JPEG images, we make a detailed introduction of the complete JPEG compression and decompression processes in this chapter. Finally, we enumerate related works about image encryption to position our contributions.

Chapter 3 proposes the basic encryption algorithm, and we prove that this algorithm is IND-CPA secure. Then we implement this encryption algorithm into three different steps of JPEG compression process. The first one encrypts the image before DCT in spatial domain. The second one encrypts image after DCT in frequency domain. The third one encrypts image after quantization. All of them can preserve the JPEG image format after encryption, but comparison of these three schemes shows that only the last one which encrypts image after quantization can preserve high image quality after decryption.

Chapter 4 applies the best scheme which is proposed in the previous chapter on several widely used image-sharing platforms. First of all, we analyse the characteristics of several widely used image-sharing platforms. Then we apply the scheme after quantization on different platforms and

analyse the experimental results. The results show that the proposed basic encryption algorithm can be used to protect the privacy on Flickr, Pinterest, Google+ and Twitter. However, it cannot be used for other existing image-sharing platforms that perform a post-processing to the published images (i.e., Facebook, Weibo and Wechat).

Chapter 5 improves the basic encryption algorithm which is proposed in chapter 3 to reduce the losses of quality resulting from the post-processing performed by some of the image-sharing platforms. We detail the improved algorithm, and how to select the parameters for different platforms in this encryption algorithm. Then we show the experimental results on Facebook, Weibo and Wechat. In the end, we evaluate the performances of the improved encryption algorithm which is developed in C++.

Chapter 6 first presents the general scenario about sharing an image with specific person including how to generate and distribute the keys, build and share the certificate, upload and download the image with the different access conditions. Then we explain how our application PixGuardian manages the keys and controls the access conditions for users. We also show how to use this application to encrypt and share the images on Facebook by using a plugin in browser Chrome to decrypt the images. Finally, we give evaluation results about the use of PixGuardian by real users.

Chapter 7 summaries our works in this thesis and gives some remarks on future research.

1.5 List of Publications

- **Image Ciphering to Ensure Privacy on Open Image-Sharing Platforms: ICIP 2015** *with Cyrielle Feron (IRT B<>Com), Gaëtan Le Guelvouit (IRT B<>Com), Christophe Bidan (CentraleSupélec Rennes)*
- **Robust and Secure Image Encryption Schemes during JPEG Compression Process: EI 2016** *with Christophe Bidan (CentraleSupélec Rennes), Gaëtan Le Guelvouit (IRT B<>Com), Cyrielle Feron (IRT B<>Com)*
- **Privacy Protection for JPEG Content on Image-Sharing Platforms: IH&MMSec 2016 and APVP 2016** *with Christophe Bidan (CentraleSupélec Rennes), Gaëtan Le Guelvouit (IRT B<>Com)*
- **Experimentation of Privacy Protection for JPEG Contents on Image-Sharing Platforms: SIN 2016** *with Christophe Bidan (CentraleSupélec Rennes), Gaëtan Le Guelvouit (IRT B<>Com)*

Chapter 2

State of the art

Contents

The main objective of this thesis is to protect privacy of user's images published on existing image-sharing platforms. When users publish their images on the existing image-sharing platforms, they can choose who has the right to view their images. However, this access control is not sufficient, since the providers of the image-sharing platforms can clearly know the content of the published images. Therefore, in order to protect user's privacy, we need to guarantee the confidentiality of user's contents. One of the most efficient and popular way to ensure the confidentiality of contents is to encrypt them. So we aim to propose an approach to encrypt user's images securely and efficiently.

In order to facilitate the following discussions about our contribution, we introduce in this section the fundamentals of two technical domains needed for our goal namely modern cryptography and image representation and compression, and describe the state of the art of image encryption.

In section 2.1, we introduce the basic theory of modern cryptography. In section 2.2, we introduce the basic knowledge of JPEG compression. In section 2.3, we enumerate related works on image encryption.

2.1 Modern cryptography

The main objectives of security is to ensure the three following properties: confidentiality, integrity and availability.

- Confidentiality means that only authorised persons can access to the data which are intended for them.
- Integrity means that only authorised users can modify the data.
- Availability means that the data are always available for the authorised users.

As one of the most commonly used security techniques, cryptography allows to guarantee the confidentiality and the integrity of the data. Especially, encryption allows to protect the confidentiality of the user's data, and is thus a natural approach to protect the user's privacy.

In the following subsections, we present the basics of modern cryptography, and more specifically, the cryptographic methods that allow to ensure the confidentiality. We first present the notion of hash functions and pseudo-random number generators. Then, we introduce the two techniques of encryption, namely the symmetric and asymmetric encryption. Finally, we discuss the problem of security of cryptographic algorithms.

2.1.1 Hash functions and Pseudo-random number generators

In this subsection, we briefly present two fundamental notions of modern cryptography: the hash functions and the pseudo-random number generators.

2.1.1.1 Hash functions

A *hash function* takes an arbitrary finite length string (which could be theoretically as long as we want) as input and outputs a fixed length value. If H is a hash function, and m an input, $H(m)$ is called the hash value of m.

A hash function is necessarily many-to-one, that means the size of the output value space is smaller than the size of the input value space. A *cryptographic hash function* is a *one-way function*, i.e., it is hard to recover the original value m just having the hash value $h = H(m)$. More specifically, the security of cryptographic hash functions is defined by the four following properties:

One-wayness (OW): given the hash value h, it is infeasible to find any input value m such that $H(m) = h$.

Target collision resistance (TCR): given input value m, it is infeasible to find another input value $m' \neq m$ such that $H(m) = H(m')$.

Collision resistance (CR): it is infeasible to find two input values m and m', such that $m \neq m'$ and $H(m) = H(m')$.

Non malleability: given a hash value $h = H(m)$, it is infeasible to produce $h' = H(m')$ where m and m' are related (e.g. $m' = m + 1$).

Notice that if the hash function H is CR, it is TCR, but not reverse. Similarly, if the hash function H is OW, it is not TCR, and TCR does not apply OW neither. Standard examples of hash functions are the hash functions of the SHA family notably SHA-2 [EH06] and SHA-3 [GBA].

2.1.1.2 Pseudo-random number generator (PRNG)

In the modern cryptography, we often look forward a random value that can be used as a secret or an initial value. Ideally, such a random value should be indistinguishable to a real truly random. Unfortunately, such value is really hard to generate and require some unpredictable physical means like atmospheric noise. Thus, in a computational world, we generate values which are nearly random: these values are called *pseudo-random* and they are generated by *pseudo-random number generators* denoted by PRNG.

A PRNG is used to generate a sequence of values with the approximate properties of sequence of random numbers. This sequence is generally determined by a relatively small set of initial values, denoted as *seed* (which can be also a random value, or a counter).

The PRNG design can be based on cryptographic primitives such as cryptographic hash functions or some mathematical problems such as chaos-based system. Here we give a brief introduction how these methods can be used to construct a PRNG function.

A cryptographic hash function H (see § 2.1.1.1) can be converted into a PRNG by combining for instance a *seed* with a counter c and hashing them [ILL89]: $H(seed, c)$, $H(seed, c+1)$, $H(seed, c+2)$, and so on. The value of the seed has to be random and remain secret. Relying on the properties of cryptographic hash functions, such PRNG generates a uniformly random result from a high-entropy but non-uniform source, and never outputs identical sequences with different inputs.

Chaotic system, known as the deterministic system with irregular behaviour, is one of the significant fields for study in the area of non-linear dynamic systems. The word *chaos* means disorder, clutter and confusion, which is described as regularity in irregularity. The chaotic system describes a dynamic system, which has a predictable behaviour in a short period of time [ASY97, Str01]. Short-term predictions may be precise, yet long-term predictions are absolutely impossible. In addition, they can be controllable using a small control signal and it is also notable that accessing a chaotic system history is impossible. Recently, with the fast development of chaos theory and practices, due to its noise-like wide power spectrum and high sensitivity to the initial condition, numbers of chaos-based algorithms have been proposed to generate pseudo-random numbers.

2.1.2 Encryption techniques

The encryption of some data allows to ensure the confidentiality of these data. The basic idea is to use a *secret* (also called the *key*) to encrypt the data before sending or publishing it, and any user knowing the appropriate key is able to decrypt the *ciphertext* and obtain the *plaintext*. Conversely, the adversary that does not know the decryption key is not able to retrieve the data from the *ciphertext*.

An encryption algorithm consists in three functions: the key generation function *KeyGen*, the encryption function *Enc*, and the decryption function *Dec*.

Key generation: $(k_\mathrm{e}, k_\mathrm{d}) = KeyGen(1^\lambda)$ is the function that given a security parameter λ generates the encryption key k_e and the corresponding decryption key k_d.

Encryption: $C = Enc_{k_\mathrm{e}}(M)$ is the function that allows to encrypt the plaintext M using the encryption key k_e. The result is the ciphertext C.

Decryption: $M = Dec_{k_\mathrm{d}}(C)$ is the function that allows to decrypt the ciphertext C using the decryption key k_d. The result is the plaintext M if and only if M has been encrypted using the corresponding encryption key k_e.

The procedure is illustrated in Fig. 2.1. The encryption algorithm is such that it is *hard* to recover the plaintext M only from the ciphertext C. Moreover, it is also *hard* to recover the decryption key k_d even knowing about the plaintext M and its corresponding ciphertext C. In subsection 2.1.3, we further discuss the notion of security of cryptographic algorithms.

It exists two kind of encryption techniques: the *symmetric* encryption in which the same key is used to encrypt and decrypt ($k_\mathrm{e} = k_\mathrm{d}$), or the *asymmetric* encryption where a *public key* is used to encrypt messages and a *private key* (corresponding to the public key used for encryption) allows to decrypt the ciphertexts. We note that this private key is not necessary known by the entity that encrypts the message.

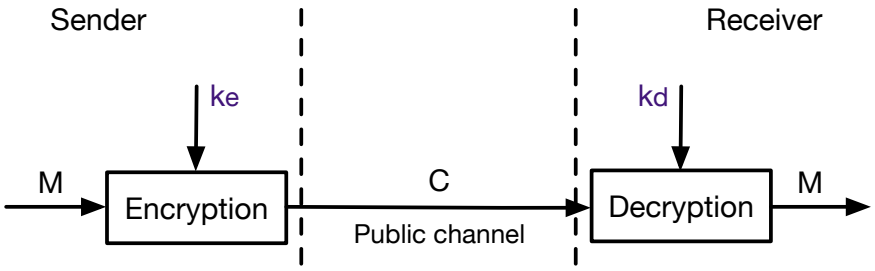


Figure 2.1: Encryption and decryption functions of a cryptosystem

2.1.2.1 Symmetric encryption

Symmetric encryption uses the same key to encrypt and to decrypt (as shown in Fig. 2.2), i.e. $k_e = k_d$. The algorithm can be described as follow:

Key generation: $K = KeyGen(1^\lambda)$ is the function that, given the security parameter λ, generates one secret key K which has to be used for encryption and decryption.

Encryption: $C = Enc_K(M)$ that uses the encryption algorithm Enc with the secret key K to encrypt the plaintext M. The result is the ciphertext C.

Decryption: $M = Dec_K(C)$ that uses the decryption algorithm Dec with the secret key K to decrypt the ciphertext C, and get the plaintext M.

The main advantage of the symmetric encryption is that the encryption and decryption algorithms are very fast, mainly because they use simple binary operations to realise encryption and decryption. Due to that, symmetric encryption is suitable for encrypting large amounts of data. However, since the same key is used to encrypt and decrypt, the *sender* (i.e., the entity that encrypts the data) and the *receiver* (i.e., the entity that decrypts the data) have to share the secret key, and thus this secret key must be *exchanged* via a separate secure channel. That is one of the main drawbacks of symmetric encryption [1].

Notice that the security of such symmetric encryption is dependent on the key length, and so that the security parameter λ is generally the number of bits of the secret key K.

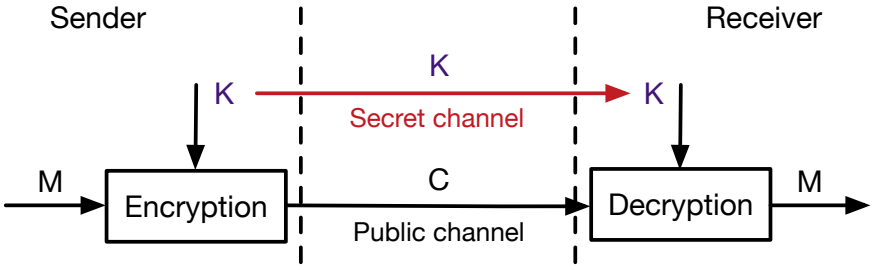


Figure 2.2: Symmetric encryption and key exchange

There are two kinds of symmetric encryption algorithms: stream cipher or block cipher algorithms.

Stream cipher is a kind of symmetric-key encryption algorithm operating on plaintext using a *pseudo random keystream* (i.e., a random sequence generated by a pseudo-random number generator (PRNG) based on the secret key and an additional random *seed*). Each plaintext digit is encrypted one by one at a time with the corresponding digit of the *keystream*, to give a digit of the ciphertext. Since a digit is typically a bit, the encryption algorithm is often the exclusive-OR (i.e. XOR). Well known examples of stream cipher are RC4 [Sch96] and A5 [3GP16].

[1] Another drawback is that symmetric encryption cannot guarantee the digital signature.

Block cipher is a kind of symmetric algorithm in which the plaintext is divided into fixed-length groups of bits – called blocks – and each block is encrypted using the secret key. Each block can be encrypted independently of each others (Electronic CodeBook mode) or the encryption of a given block can depend on the encryption of previous blocks (Cipher Block Chaining mode). Notice that block cipher can also be used in a stream cipher mode (Cipher FeedBack and Counter modes) [Dwo]. Well-known block cipher algorithms are DES (Data Encryption Standard) [Den82], AES (Advanced Encryption standard) [DR99], etc.

2.1.2.2 Asymmetric encryption

In order to solve the problem of secure key exchange, asymmetric cryptography has been proposed by Whitfield Diffie and Martin E. Hellman in 1976 [DH76]. In such cryptosystem, the keys used to encrypt and decrypt are different, as shown in Fig. 2.3. The algorithm can be described as follow:

Key generation: $(k_{\text{pub}}, k_{\text{prv}}) = KeyGen(1^{\lambda})$ is the function that given the security parameter λ generates a public key k_{pub} (that can be publicly published) and a private key k_{prv} (that must be kept secret).

Encryption: $C = Enc_{k_{\text{pub}}}(M)$ that uses the encryption algorithm Enc with the public key k_{pub} to encrypt the plaintext M. The result is the ciphertext C.

Decryption: $M = Dec_{k_{\text{prv}}}(C)$ that uses the decryption algorithm Dec with the private key k_{prv} to decrypt the ciphertext C, and get the plaintext M.

Since the public key k_{pub} can be published, the additional secret channel is no longer needed to exchange the secret key.

In such cryptosystem, the encryption should ensure that it is *hard* to recover M from C without k_{prv} but knowing k_{pub}, and no one can derive the private key k_{prv} from the public key k_{pub}. In order to guarantee these properties, asymmetric cryptosystems are based on some complex mathematic problems such as factorization or discrete logarithm, because until now, no one knows any algorithm which can solve these problems in polynomial time. But on the other hand, these complex problems have a significant impact on the computational cost of the encryption and decryption processes. This drawback implies that normally we do not use asymmetric cryptosystem to encrypt large data, but keep it to encrypt small data such as symmetric keys.

The most well-known asymmetric encryption algorithm is the RSA cryptosystem [RSA78]. We note that it is considered as the first asymmetric cryptosystem and is still in use nowadays (only the length of keys is risen with the times).

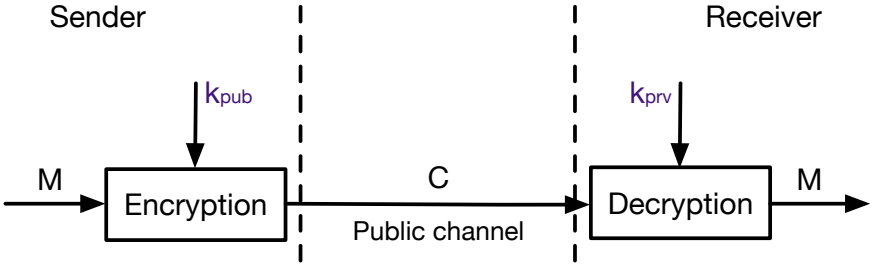


Figure 2.3: Procedures of asymmetric cryptosystem

Notice that asymmetric cryptosystems can also be used for digital signature. Indeed, instead of using the public key of the receiver, if the sender uses his/her private key to *sign* the plaintext, he/she is the only one to be able to perform this operation, but everyone can *verify* the *signature* (i.e., the ciphertext obtained by the encryption with the private key). More specifically, using asymmetric cryptosystem for digital signature can be described as follow:

Key generation: $(k_{\text{pub}}, k_{\text{prv}}) = KeyGen(1^{\lambda})$ is the function that given the security parameter λ generates a public key k_{pub} (that can be publicly published) and a private key k_{prv} (that must be kept secret).

Signing: $\sigma = Sign_{k_{\text{prv}}}(M)$ that uses the algorithm $Sign$ with the private key k_{prv} to sign the plaintext M. The result is the signature σ.

Verifying: $Verify_{k_{\text{pub}}}(\sigma)$ that uses the algorithm $Verify$ with the public key k_{pub} to verify the signature σ.

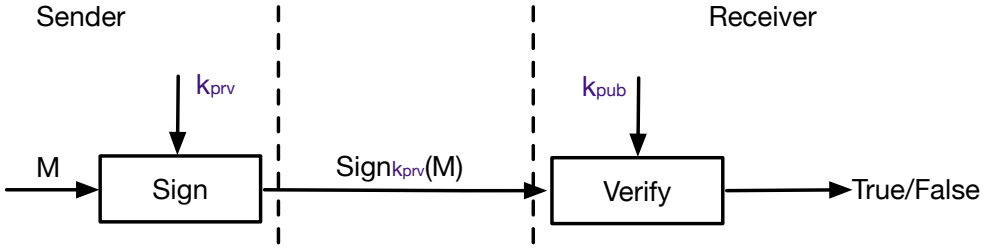


Figure 2.4: Integrity providing by a digital signature.

Notice that, as for encryption, digital signature based on asymmetric cryptosystem can only be used on small data. The solution is then to rely on hash function to compute a cryptographic hash value, and to sign this hash value instead of the plaintext data. The figure 2.4 illustrates how the integrity could be guaranteed by digital signature.

2.1.2.3 Symmetric vs. asymmetric

In the previous section, we have briefly introduced symmetric and asymmetric encryptions, and given their advantages and disadvantages. Thus, symmetric encryption is faster but requires secure key exchange, whereas asymmetric encryption no longer needs secure key exchange but is slower. The following table 2.1 summarizes the comparison between symmetric and asymmetric encryptions.

	Confidentiality	Advantages	Disadvantages
Symmetric cryptosystem	Encryption by using the secret key	Fast	Need a secure key exchange
Asymmetric cryptosystem	Encryption by using the public key	Do not need a secure key exchange	Slow

Table 2.1: Properties of symmetric and asymmetric encryptions.

The classical approach in modern cryptography is thus to combine these different types of encryption: symmetric encryption is used to encrypt large data whereas asymmetric encryption is used to encrypt the secret key needed by the symmetric encryption. For instance, let us consider the example of a user that wants to protect the sending of an image Img to one of his/her friends. Since images are large data, he/she has to encrypt Img using symmetric encryption with a randomly chosen secret key K. Then, he/she is able to send the encrypted image $Enc_K(Img)$ to his/her friend, as well as the secret key K encrypted with the public key k_{pub} of his/her friend. When receiving the data, the friend is able to use his/her private key k_{prv} to decrypt and retrieve the secret key K, and then use this secret key K to decrypt the $Enc_K(Img)$ to obtain the plaintext image Img. This process is shown in Fig. 2.5.

Notice that this process works only if the sender can trust that the used public key is the one of his/her friend. A Public Key Infrastructure (PKI) allows specifically to resolve this problem. The basic idea is to raise on Certificate Authorities (CAs) to certify the link between the identity of an entity and a public key. The link is mainly established through a registration process and issuance of certificates. PKI permits to avoid the identity usurpation and to protect against the use of wrong public keys notably generated by a Man-In-The-Middle adversary (MitM).

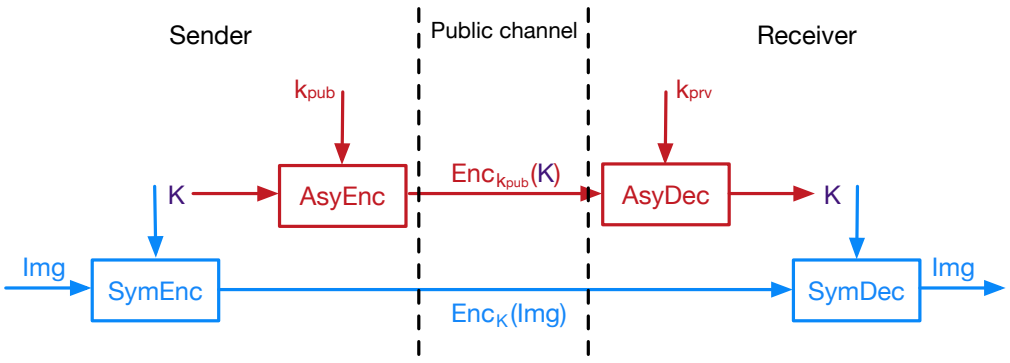


Figure 2.5: Combination of symmetric and asymmetric cryptosystem

2.1.3 Security of cryptographic algorithms

Today, one of the main challenge in cryptography is to prove the security of the cryptographic algorithms. The security of the cryptographic algorithms may be evaluated according to the information theory, the computational complexity theory, or from a probabilistic point of view. We briefly present each of these approaches.

2.1.3.1 Information-theoretic security

A cryptosystem is information-theoretically secure if its security can be proven according to the information theory, as defined by Claude Shannon [Sha49]. That means that even if the adversary has unlimited computing power and time, he/she is not able to deduce any information of the plaintext from the ciphertext. Thus, this notion of security is very strong, and designing a crytosystem that is information-theoretically secure is very difficult. The common example of cryptosystem that is information-theoretically secure is the One-Time-Pad.

2.1.3.2 Computational security

According to the computational complexity theory, some problems are known to be *hard*, that is, we do not know any polynomial-time algorithm that allows to resolve these problems. A cryptosystem is computationally secure if breaking it in polynomial-time is equivalent to solve a problem that is known as hard in the computational complexity theory [Sta74]. In other words, any adversary with polynomial-time computational resources is not able to retrieve the plaintext from the ciphertext.

Asymmetric cryptosystems are often based on complex mathematic problems, such as factorisation or discrete logarithm, that are proved or assumed to be hard in the computational complexity theory. Thus, the security of these cryptosystems is often computational. The typical examples are the RSA or the Diffie-Hellmann cryptosystems.

2.1.3.3 Semantic security

In 1982, Goldwasser and Micali have introduced the notion of probabilistic encryption [GM82]. Basically, the probabilistic encryption consists in ensuring that, even if a plaintext m is encrypted and sent twice, then an adversary is not able to deduce that the two corresponding ciphertexts correspond to the same plaintext. Probabilistic encryption can be achieved by randomizing the encryption, using for instance, a random seed in a stream cipher, or a random initial vector in a block cipher.

The notion of semantic security is a generalisation of the notion of probabilistic encryption. The word "semantic" came from the definition that the encryption reveals no information no matter what kind of semantics are embedded in the plaintext. Given two plaintexts m_0, m_1 of the same length, and their corresponding ciphertexts c_0, c_1, a cryptosystem is semantically secure, if an adversary cannot determine which ciphertext corresponds to which plaintext with probability greater than $\frac{1}{2}$. That means, ciphertext of some unknown message does not reveal any additional information.

The semantic security is generally expressed in terms of indistinguishability (noted as IND). More specifically, if the plaintexts m_0 and m_1 are chosen by the adversary, we say that a cryptosystem is IND-CPA (Indistinguability under Chosen Plaintext Attack). Moreover, if the adversary is able to decrypt some chosen ciphertexts before sending the two plaintexts m_0 and m_1, we say that the cryptosystem is IND-CCA (Chosen Cyphertext Attack), and if he/she can also decrypt some chosen ciphertexts after receiving c_0 and c_1, we say that the cryptosystem is IND-CCA2 (adaptative Chosen Ciphertext Attack).

2.1.3.4 IND-CPA for a symmetric cryptosystem

In this thesis, we aim to propose an image encryption algorithm to protect user's published images, and this algorithm should guarantee the security of user's images. Since the images are large data, our encryption algorithm is symmetric. And because we want to make sure that, even if we encrypt a black image and a white image, it is hard to know which encrypted image corresponds to which original image, we require that our encryption algorithm is IND-CPA. In that purpose, we give here more details on how to prove that a symmetric cryptosystem is IND-CPA.

IND-CPA [MO14] for a symmetric cryptosystem is represented by a specific game between an adversary and a challenger. Let $\mathcal{C}$ be the challenger and $\mathcal{A}$ be the adversary. Here is the game:

1. $\mathcal{C}$ generates a secret key K and does not reveal it.

2. $\mathcal{A}$ may perform any number of encryptions and other operations (but not decryption) on the result and the challenger forwards the related answers.

3. $\mathcal{A}$ chooses and sends two different plaintexts m_0 and m_1 to $\mathcal{C}$.

4. $\mathcal{C}$ chooses one bit $b \in \{0, 1\}$ randomly, and sends the challenge $c = E_K(m_b)$ to $\mathcal{A}$ with c the encryption of the message m_b with the secret key K.

5. $\mathcal{A}$ gives his guess for the value of b to $\mathcal{C}$.

The adversary is modelled by a probabilistic polynomial time Turing machine. That means the adversary must complete the game and output a guess within a polynomial number of time steps. The symmetric cryptosystem is indistinguishable under chosen plaintext attack if the adversary wins the game with a negligible advantage. It means that he wins the game with a probability equals to $\frac{1}{2} + \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a negligible function in the security parameter λ (i.e. the size of the key in bits).

A cryptosystem is IND-CPA$ secure if it has pseudorandom ciphertexts in the presence of chosen plaintext attack. The game representation of this security definition is the same as the game for IND-CPA except for steps 3 and 4, which become:

3. $\mathcal{A}$ sends one plaintext m to $\mathcal{C}$.

4. $\mathcal{C}$ chooses one bit $b \in \{0, 1\}$ randomly. If $\mathcal{C}$ chooses 0, $\mathcal{C}$ sends $c = E_K(m)$ to $\mathcal{A}$. Else, $\mathcal{C}$ sends a random number of n bits. n being the size of $E_K(m)$.

In other words, a cryptosystem is IND-CPA$ secure if the adversary cannot distinguish between a ciphertext and a sequence of random numbers with a probability superior to $\frac{1}{2} + \epsilon(\lambda)$. An IND-CPA$ secure cryptosystem is clearly IND-CPA secure. If an adversary cannot distinguish between the challenge c and a sequence of random numbers with a probability superior to $\frac{1}{2} + \epsilon(\lambda)$, then he cannot guess the right value of b with a better probability.

2.2 Image representation and compression

We aim to encrypt an image and to ensure that the results of encryption and decryption are still in image format. Considering images on most of image-sharing platforms are compressed by using JPEG standard [PM93], and the downloaded images are always in JPEG format, we focus on processing JPEG images. In this section, we briefly introduce the basic knowledge of image representation and JPEG compression processes.

In order to represent an image, the most common format is bitmap, which can be displayed by different color models and color coding methods. So in the first subsection, we introduce how to display the images by using different kind of bitmaps. In the second subsection, we present two-dimensional discrete cosine transform which is the transform used in JPEG standard. Then in the third subsection, we detail the JPEG compression and decompression processes.

2.2.1 Representation of images

A bitmap, as known as raster graphic, is a dot matrix data structure. Each element of this matrix represents a pixel of an image and the color information of each pixel is displayed by different color coding methods, for example, RGB combinations or grayscale values. RGB color model which is the most commonly used one consists of three additive primary colors: red, green and blue. They are added together in a fixed manner to reproduce various colors (see Fig. 2.7). Nowadays, RGB color model mostly uses 8 bits per channel[2] to indicate red, green and blue intensity; thus each pixel is represented by 24 bits (8 bits/pixel/channel $\times$ 3 channels = 24 bits/pixel). In the case of 24-bit color, each pixel can be denoted by #xxyyzz, where xx, yy and zz are three byte hexadecimal numbers and indicate the three colors respectively. The range of each color value is 00-FF, which correspond to the decimal range 0-255. So in one channel, there are $2^8 = 256$ possible colors, and one image has 16,777,216 (256^3) discrete combinations of RGB values (shown in Fig. 2.6). A grayscale bitmap is 8-bit image, which has $2^8 = 256$ possible gray values, and a binary bitmap (1 bit/pixel) only has two possible values (black and white).

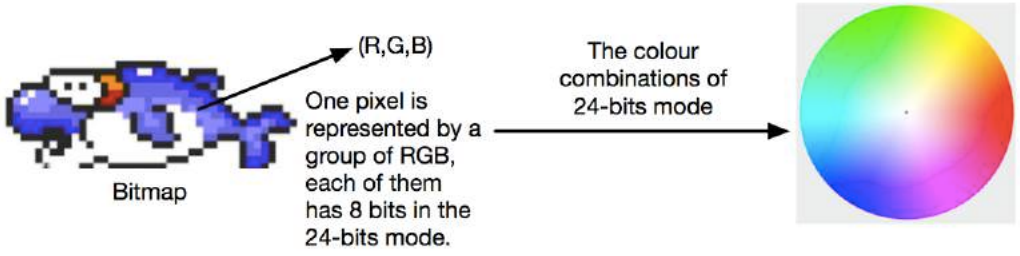


Figure 2.6: A bitmap represented by 24-bit RGB pixels

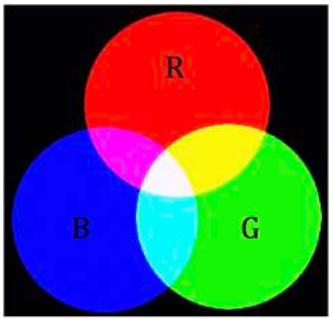


Figure 2.7: Mixture of RGB models

Another color model is YCbCr. It is a way of encoding RGB information. The Y component represents the brightness of a pixel, Cb and Cr components represent the chrominance (Cb is blue component and Cr is red component). This coding system is very useful, because the human eyes are more sensitivity to the changes of brightness rather than differences of color. YCbCr and RGB both can be converted to each other as follow:

[2]It also can use more, for high dynamic range (HDR) image or video, e.g. 12 bits per channel.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.00093 & 1.401687 \\ 1 & -0.3437 & -0.71417 \\ 1 & 1.77216 & 0.00099 \end{bmatrix} \begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix} \tag{2.2}$$

Computer images are always displayed as bitmaps, but this format is not efficient for storage. For example, it takes $(1024 \times 768) \times 24$ bits/pixel $= 2.25$ MB to store an image of 1024×768 pixels. A commonly used compression method for bitmap is to use an indexed color table. Several representative colors in the bitmap image (usually less than 256 colors) generate the table, and the original colors are represented by using the index of this color table. In this way, the original image can be lossy compressed; it is only suitable for compressing web graphics or other graphics with less color, not suitable for colorful graphics like photographies.

For that purpose, we need some other efficient raster graphic formats or compressed variations to store an image with fewer memories, such JPEG or PNG. For instance, if we store the image of 1024×768 pixels into the JPEG format with highest quality (without compression), it only takes about 800 KB, and with lowest quality, it only takes about 20 KB. JPEG is the most popular format, its compression processes are based on the two-dimensional discrete cosine transform (2D-DCT), which is introduced in the next subsection.

2.2.2 Two-dimensional discrete cosine transform (2D-DCT)

Because of the complexity of an image, when we process an image directly in the spatial domain, it involves some expensive computations. Therefore, we often use various image transformation methods, such as Fourier transform, Walsh transform, wavelet transform, etc. to convert the spatial domain processing into another domain. In this way, not only the amount of calculation can be reduced, but also more effective treatments can be applied (e.g. Fourier transform can carry out the digital filtering in frequency domain).

The compression processes of JPEG are based on the two-dimensional discrete cosine transform (2D-DCT). Suppose that we have an image signal $f(x, y)$, the two-dimensional DCT can be represented as:

$$F(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos[\frac{(2x+1)u\pi}{2N}] \cos[\frac{(2y+1)v\pi}{2N}], \tag{2.3}$$

where

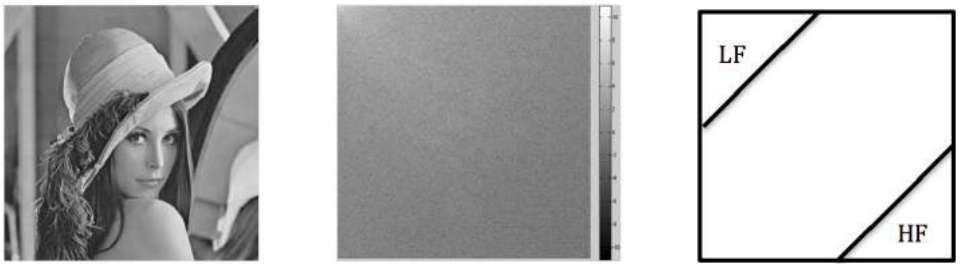
$$\alpha(u), \alpha(v) = \begin{cases} \sqrt{\frac{1}{N}}, & u, v = 0 \\ \sqrt{\frac{2}{N}}, & u, v = 1, 2, \cdots, N-1. \end{cases} \tag{2.4}$$

The inverse transform 2D-IDCT is:

$$f(x,y) = \alpha(u)\alpha(v) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \cos[\frac{(2x+1)u\pi}{2N}] \cos[\frac{(2y+1)v\pi}{2N}], \tag{2.5}$$

where $\alpha(u)$ and $\alpha(v)$ are defined as above and $x, y = 0, 1, 2, \cdots, N-1$.

From Fig. 2.8, we clearly see the *energy compaction* property of DCT. It means that after computing the DCT, most of information of signal is concentrated in the low-frequency part which is in the top-left corner. This property is very useful when we compress an image. Because we only want to hold the most important information which contains more energy, and reduce the others as much as possible.



(a) Original image (b) Energy distribution (c) Frequency distribution

Figure 2.8: Energy and frequency distributions of the image after 2D-DCT transform

2.2.3 The JPEG format

JPEG[3] is a standard for image coding which was created by the Joint Photographic Experts Group. It is one of the widely used standards for storing and compressing image. Digital cameras and other image capture devices use JPEG standard to store and transmit images. It allows to lossy compress any digital photography. Images on the Internet are compressed by using JPEG standard as well. But because this compression algorithm is lossy, the image quality may suffer a visible damage. There is a tradeoff between storage size and image quality. Next, we respectively talk about each step of JPEG compression processes.

2.2.3.1 Image encoding

JPEG standard specifies some different encoding methods to create a JPEG file. Here we choose the most common method of JPEG encoding which applies to the RGB color model (described in Sec. 2.2.1) with 24 bits per pixel. In this method, there are six steps to encode an image, such as DCT, quantization, entropy encoding, etc. Fig. 2.9 illustrates the complete JPEG compression and decompression processes of a color image. For a grayscale image, the color space transformation and the downsampling are not needed. We give a brief description of all of them as follows.

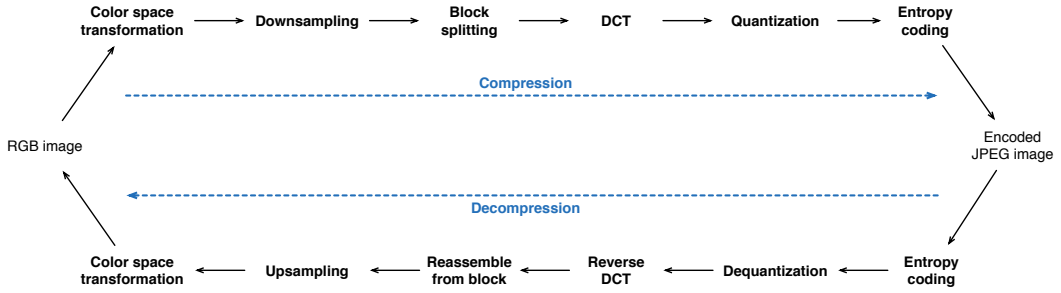


Figure 2.9: JPEG encoding and decoding processes

Color space transformation. First, the color image is converted from RGB to YCbCr. As stated in Sec. 2.2.1, the human visual system can see more fine details in the brightness of an image. Using this knowledge, the encoder can be designed more efficiently to compress the image by using YCbCr. According to Eq. 2.1 and Eq. 2.2, if the pixel value of RGB is (255, 255, 255), after converting into YCbCr, the value becomes to (255, 128, 128). If the value

[3] http://en.wikipedia.org/wiki/JPEG

is (0, 0, 0) in RGB space, it becomes to (0, 128, 128) after converting into YCbCr. If the value of RGB is (128, 128, 128), the converted YCbCr value is (128, 128, 128). The scheme of conversion is showed in Fig. 2.10.

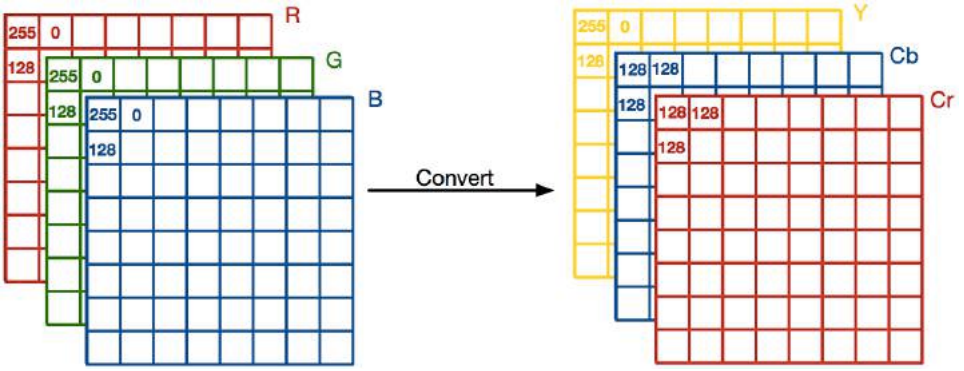


Figure 2.10: Example of RGB converting into YCbCr

Downsampling. According to the previous step, the chrominance components can be reduced to realize the compression. This process is called downsampling or chroma subsampling. The Y components never change and as a standard, we note that 4 pixels of Y components are always sampled 4 times, so the ratio is always 4:x:x. The same ratio of sampling is used for both Cb and Cr components. The illustration from Fig. 2.11 clearly shows how they are implemented. The ratios of downsampling could be 4:4:4 (no downsampling), 4:2:2 (chrominance components become half of the original in the horizontal direction), 4:2:0 (most commonly used, chrominance components become half of the original both in the horizontal direction and vertical directions), or 4:1:1 (chrominance components become quarter of the original in the horizontal direction), etc.

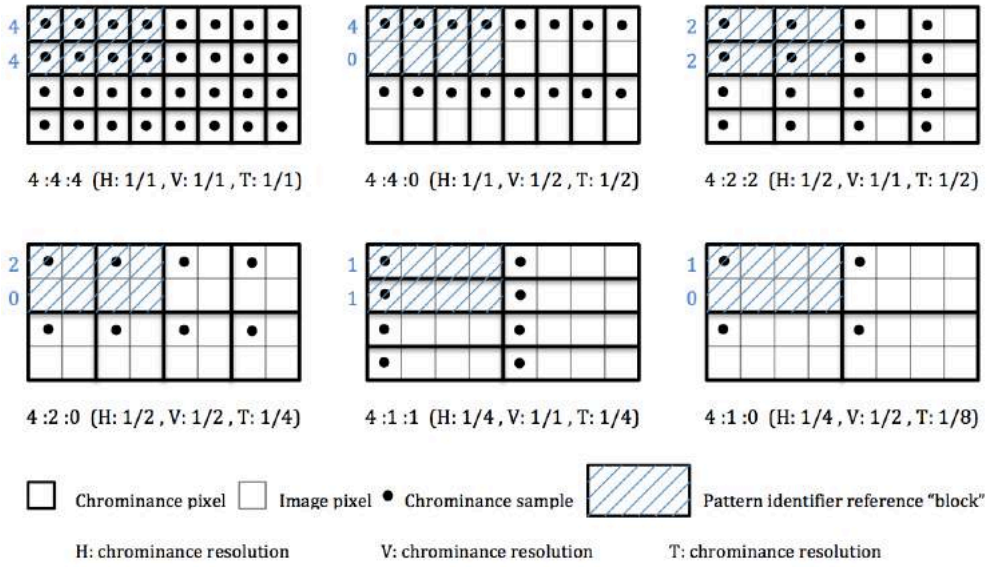


Figure 2.11: Chrominance subsampling patterns [Ker05]

Block splitting. Given the three channels Y, Cb, Cr, the image is split into non-overlapping blocks, and each block has 8×8 components. If the number of components are not multiple of 8×8, we must fill the last incomplete blocks with some form of dummy data. Usually, the most common filling method is repeating the edge pixels.

Discrete cosine transform. In this step, each 8×8 block in each channel is converted to frequency domain by using two-dimensional discrete cosine transform (introduced in Sec. 2.2.2). For a 24 bits per pixel image, in each channel, the values of components fall in the range [0, 255]. But in order to reduce the dynamic range requirements in the following DCT processing, the range is shifted to [-128, +127]. Then, according to Eq. 2.3, we compute the DCT of each 8×8 block.

Take an 8×8 block (2.6) as example. The first step is the range shift. All the components are minus 128, and we obtain the matrix (2.7). Then we compute the DCT, and this 8×8 block becomes matrix (2.8). The absolute value of coefficient in top-left corner is larger than the others, that is the DC (direct component) coefficient. The remaining 63 coefficients are AC (alternating component) coefficients. Due to the energy compaction property of DCT, most of the information of the original image is preserved in the low frequency coefficients which contains the DC coefficient and some AC coefficients in the upper left corner.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \tag{2.6}$$

$$\overset{\text{Minus 128}}{\downarrow}$$

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \tag{2.7}$$

$$\begin{bmatrix} -415.3750 & -30.1857 & -61.1971 & 27.2393 & 56.1250 & -20.0952 & -2.3876 & 0.4618 \\ 4.4655 & -21.8574 & -60.7580 & 10.2536 & 13.1451 & -7.0874 & -8.5354 & 4.8769 \\ -46.8345 & 7.3706 & 77.1294 & -24.5620 & -28.9117 & 9.9335 & 5.4168 & -5.6490 \\ -48.5350 & 12.0684 & 34.0998 & -14.7594 & -10.2406 & 6.2960 & 1.8312 & 1.9459 \\ 12.1250 & -6.5534 & -13.1961 & -3.9514 & -1.8750 & 1.7453 & -2.7872 & 3.1353 \\ -7.7347 & 2.9055 & 2.3798 & -5.9393 & -2.3778 & 0.9414 & 4.3037 & 1.8487 \\ -1.0307 & 0.1831 & 0.4168 & -2.4156 & -0.8778 & -3.0193 & 4.1206 & -0.6619 \\ -0.1654 & 0.1416 & -1.0715 & -4.1929 & -1.1703 & -0.0978 & 0.5013 & 1.6755 \end{bmatrix} \tag{2.8}$$

Quantization. This step is the main part of compression. In each 8×8 DCT block, the coefficient in top-left corner is the DC (direct component) coefficient, and the remaining 63 coefficients are AC (alternating component) coefficients. The coefficients in upper left corner are the low frequencies, where the most important visual characteristics of the image are placed. In contrast, the highest frequencies are in the lower right corner and correspond to the details of the image. The quantization step allows to reduce the information. Each DCT coefficient in each 8×8 block is divided by the corresponding element of quantization table, and then rounding to the nearest integer. This rounding operation directly causes the lossy compression in the encoding process.

In Fig. 2.12, we provide the standard default quantization tables for luminance and chrominance channels that used by the IJG (Independent JPEG Group) code library:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

(a) Luminance table

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

(b) Chrominance table

Figure 2.12: Standard default quantization tables

The factor of quality can be 1 (terrible) to 100 (very good). Different quantization table of different compression ratio can be calculated as:

$$q(i,j) = \begin{cases} \left\lfloor \dfrac{t(i,j) \times (\frac{5000}{f}) + 50}{100} \right\rfloor, & 0 < f < 50 \\ \left\lfloor \dfrac{t(i,j) \times (200 - 2f) + 50}{100} \right\rfloor, & 50 \leq f \leq 100 \end{cases} \tag{2.9}$$

where $t(i,j)$ $(i,j \in [0,7])$ is the element of standard default quantization tables in Fig. 2.12, $f \in]0,100]$ is the factor of quality, $q(i,j)$ $(i,j \in [0,7])$ is the element of new quantization table. The element of quantization table controls the compression ratio, a larger value provides a greater compression.

According to Eq. 2.9, if the factor is 80, the luminance quantization table is shown as:

$$\begin{bmatrix} 6 & 4 & 4 & 6 & 10 & 16 & 20 & 24 \\ 5 & 5 & 6 & 8 & 10 & 23 & 24 & 22 \\ 6 & 5 & 6 & 10 & 16 & 23 & 28 & 22 \\ 6 & 7 & 9 & 12 & 20 & 35 & 32 & 25 \\ 7 & 9 & 15 & 22 & 27 & 44 & 41 & 31 \\ 10 & 14 & 22 & 26 & 32 & 42 & 45 & 37 \\ 20 & 26 & 31 & 35 & 41 & 48 & 48 & 40 \\ 29 & 37 & 38 & 39 & 45 & 40 & 41 & 40 \end{bmatrix} \tag{2.10}$$

Then suppose that the previous DCT block (2.8) is quantized by this table, the quantized DCT coefficient block results in:

$$\begin{bmatrix} -69 & -8 & -15 & 5 & 6 & -1 & 0 & 0 \\ 1 & -4 & -10 & 1 & 1 & 0 & 0 & 0 \\ -8 & 1 & 13 & -2 & -2 & 0 & 0 & 0 \\ -8 & 2 & 4 & -1 & -1 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.11}$$

Entropy coding. Different from the previous step, the entropy coding provides lossless compression. First of all, in each block, the quantized DCT coefficients are arranged in a *zigzag* order (shown in Fig. 2.13). In the matrix, the low frequencies are in upper left corner, and the high frequencies are in the lower right corner, after zigzag arrangement, the similar frequencies are grouped together. Then the run-length encoding (RLE) algorithm is applied to them, inserting length coding zeros, and then encoded by Huffman coding with general-purpose Huffman tables which are provided by JPEG standard. At the end of encoding, a bitstream is written into a JPEG image file.

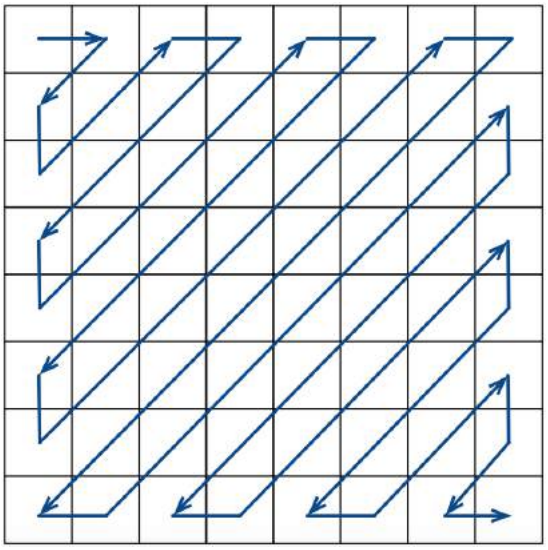


Figure 2.13: Zigzag ordering of JPEG image components

2.2.3.2 Image decoding

The decoding process is used to display the image, and consists of doing all the above in reverse. After decoding the bitstream in the JPEG image file by using the Huffman decoding and the run-length decoding, the quantized DCT coefficient matrix can be obtained. Since this step is lossless compression, the quantized DCT matrix is totally same as the matrix (Eq. 2.11). Then multiplying by the quantization table above results in a DCT coefficient matrix (Eq. 2.13). Because of the rounding operation, some information has been lost and cannot be recovered, so this matrix only closely resembles to the original one.

$$\begin{bmatrix} -414 & -32 & -60 & 30 & 60 & -16 & 0 & 0 \\ 5 & -20 & -60 & 8 & 10 & 0 & 0 & 0 \\ -48 & 5 & 78 & -20 & -32 & 0 & 0 & 0 \\ -48 & 14 & 36 & -12 & -20 & 0 & 0 & 0 \\ 14 & -9 & -15 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

Then according to Eq. 2.5, we compute the two-dimensional inverse DCT of each 8×8 block. Because all the pixel values should be integers, so the outputs of IDCT need to be rounded (will lost some information) as matrix (Eq. 2.13). Then shifting the range back to $[0, 255]$ by adding 128 to each component and get the 8×8 block (2.14).

$$\begin{bmatrix} -76 & -71 & -70 & -66 & -58 & -60 & -63 & -57 \\ -64 & -71 & -65 & -39 & -24 & -44 & -60 & -54 \\ -61 & -77 & -61 & -8 & 13 & -26 & -60 & -55 \\ -63 & -78 & -57 & 1 & 24 & -20 & -60 & -57 \\ -57 & -68 & -58 & -21 & -6 & -36 & -62 & -56 \\ -49 & -60 & -67 & -59 & -51 & -62 & -67 & -55 \\ -44 & -55 & -70 & -73 & -68 & -68 & -62 & -46 \\ -41 & -50 & -63 & -62 & -54 & -55 & -49 & -32 \end{bmatrix} \tag{2.13}$$

Add 128
↓

$$\begin{bmatrix} 52 & 57 & 58 & 62 & 70 & 68 & 65 & 71 \\ 64 & 57 & 63 & 89 & 104 & 84 & 68 & 74 \\ 67 & 51 & 67 & 120 & 141 & 102 & 68 & 73 \\ 65 & 50 & 71 & 129 & 152 & 108 & 68 & 71 \\ 71 & 60 & 70 & 107 & 122 & 92 & 66 & 72 \\ 79 & 68 & 61 & 69 & 77 & 66 & 61 & 73 \\ 84 & 73 & 58 & 55 & 60 & 60 & 66 & 82 \\ 87 & 78 & 65 & 66 & 74 & 73 & 79 & 96 \end{bmatrix} \tag{2.14}$$

So far, the components are approximately reconstructed. For color images, the last steps are upsampling and convert YCbCr color space back into RGB, and then the decoded image is obtained.

We can compare the original matrix (Eq. 2.6) and the decoded matrix (Eq. 2.14) by taking the difference, and calculate the error values as following:

$$\begin{bmatrix} 0 & -2 & 3 & 4 & 0 & -7 & -1 & 2 \\ -1 & 2 & -8 & 1 & 5 & 1 & 1 & -2 \\ -5 & 8 & 1 & -7 & 3 & 2 & -2 & 0 \\ -2 & 8 & 0 & -7 & 2 & -2 & 2 & -2 \\ -4 & 1 & -2 & -3 & 4 & -4 & 2 & -2 \\ 0 & -3 & -1 & 1 & 0 & 2 & -3 & 2 \\ 1 & -2 & 6 & 4 & -5 & 1 & -1 & 1 \\ 0 & 1 & 4 & 2 & -9 & 3 & -1 & -2 \end{bmatrix} \tag{2.15}$$

The average absolute error is about $\frac{1}{64}\sum_{x=0}^{7}\sum_{x=0}^{7}|e(x,y)| = 2.66$ per pixels.

2.3 Related Works

With the development of digital imaging applications and image-sharing platforms, people are increasingly concerned about the security and privacy of their personal information. To realise our objectives of protecting the privacy of the user-generated content published on the existing image-sharing platforms, our intent is to allow each user to encrypt the images before publishing them, and to give the decryption key only to the friends that are authorized to access the images. In this way, if the image encryption algorithm is secure, people who do not have the key (including the providers of image-sharing platforms) cannot know the contents of published images. So far many image encryption algorithms have been proposed, but not all of them can be used to protect user's privacy on image-sharing platforms.

2.3.1 Traditional encryption

As we described in Sec. 2.1, various traditional encryption algorithms have been proposed and widely used, such as AES, DES, RSA, etc. Most of which are used on text or binary data and provide very good encryption performances. It is also possible to use them for multimedia data encryption scheme. If we have an image, we directly encrypt it by using traditional cryptography algorithms: the confidentiality of images is thus guaranteed. For example, Dang *et al.* [DC00] choose DES to encrypt compressed image data. And an image encryption system based on Hill cipher is proposed in [Dey12].

However, the result of the encryption is a binary file rather than an image, and when we try to publish this binary file on image-sharing platforms, none of the platforms view it as a correct format (the acceptable formats being JPEG, PNG, GIF, or TIFF, etc.). Thus, we can not simply used traditional encryption to protect the user's privacy on image sharing platforms: we need an encryption scheme such that the result of the encryption is accepted as an image by the images sharing platforms, i.e., an encryption scheme that preserves the format of the images.

2.3.2 Encryption preserving image format

In the following sections, we introduce some existing solutions that allow to *scramble* or to *encrypt* the image while preserving the image format.

2.3.2.1 Scrambling algorithms

Scrambling is a technique which is efficient and easy to implement. So, many approaches to protect image are based on it. The objective of image scrambling is to produce a non-intelligible image, which prevents humans or even computer vision system from understanding the true content. Image scrambling can be done in the spatial domain or in the frequency domain.

- **In the spatial domain**

  Taking into account the execution time, some researchers prefer to permute the pixel directly in the spatial domain, and the simplest image scrambling is random permutation without any regular pattern. Wright et al. [WFL15] proposed two techniques based on scrambling techniques. In the first proposed scheme, they only permute the locations of the pixels within the blocks. In the second one, they permute the sub-blocks within the blocks and then shuffle pixels in sub-blocks.

  Usman et al. [UJN+07] proposed a row and column random permutation in order to break the correlation of the edges of the image. The columns and the rows of the image are divided into several parts. The number of column or row in each part is random. Then the parts are randomly scrambled many times to produce a modified image. And after that, a pixel arrangement is suggested to reorder the position of the pixels according to a certain rule before the final image is obtained. Premaratne et al. [PP12] proposed a similar approach.

  The scrambling in spatial domain only change the position of the pixels, not their values. Consequently, the original and the scrambled images have the same statistical properties (such as gray value distribution, mean of gray values, entropy, etc.): thus, it make easy to differentiate two scrambled images according to the statistical properties of the corresponding original images. Thus, the scrambling in the spatial domain is not IND-CPA secure.

- **In the frequency domain**

  On the opposite to the scrambling in spatial domain, the scrambling in frequency domain can change the statistical characteristics of images, and thus is more robust against statistical attacks.

  In [KE14], the authors proposed a tool which allows to scramble a selective region of JPEG image. The scrambling consists in flipping the signs of quantized DCT coefficients based on a random bitstream. However, since only the signs of coefficients are scrambled, it is easy to differentiate two scrambled images: thus the solution is not IND-CPA secure.

Thus, the scrambling can be done in the spatial domain or in the frequency domain but in both case, even if the image is unrecognized after scrambling, the solution is not IND-CPA secure since only the position of the pixels or the DCT coefficients changes, but not their value. The scrambling solutions are generally easily broken [QN98], except when they are combined with encryption algorithms (see §2.3.2.4).

2.3.2.2 Modifications applied to traditional encryption

As we mentioned in section 2.3.1, traditional encryption algorithms can be used to encrypt images. However, since the result of the encryption is a binary file and not an image, it can not be uploaded on image-sharing platforms. One solution consists in using traditional encryption algorithms to encrypt the pixel values in order to preserve the image format after encryption. For instance, Zhang et al. [YPWSp+09] proposed to combine a chaotic map with DES to encrypt the image at the pixel level. The chaotic map is used to generate a pseudo-random bitstream which is used to generate the subkeys of the DES. Similarly, Zeghid et al. [ZMK+07] proposed to modify the AES algorithm to encrypt the pixels of images, but instead of using a chaotic map, they use another pseudo-random

number generator. In [KSHR10], the authors proposed to modify the AES algorithm to encrypt the pixels of images, by adjusting the *ShiftRow transformation* operation of the traditional AES algorithm.

Notice that the two first solutions are based on the replacement of the original subkeys by random bitstreams. Thus, the original encryption algorithms are not really modified, and we can reasonably assert that the security of the modified encryption algorithms is maintained. In particular, used as probabilistic algorithms, they are IND-CPA secure. On the contrary, the last solution is based on the modification of a fundamental operation of the AES algorithm. As a consequence, because the modified encryption algorithm is different from the AES algorithm, we can not assert that this algorithm is secure because of the security of the AES algorithm.

In any case, given that image-sharing platforms generally prefer JPEG format, the use of traditional encryption algorithms to encrypt the pixel values of the images is not adapted to our context. Indeed, the encrypted image will have to be compressed before download, and since the compression is a lossy process, when decompressing, the decompressed image will be slightly different from the original encrypted image, and thus, it won't be correctly decrypted.

A better approach should be to encrypt the DCT coefficients of the images. Thus, Yang, Lu and Han [SZS04] proposed an asymmetric encryption scheme based on matrix transformation. The private key and public key are two matrices with size $P \times K$ and $K \times P$ created by Gaussian white noise. Given the original image into DCT / frequency domain, the private key is used to encrypt (multiply) the frontal $K \times P$ coefficients of every divided $P \times P$ block respectively. In the decryption process, the frontal $K \times P$ coefficients are recovered by using the public key. The security of the encryption is thus based on the results of matrix theory. Moreover, since the encryption process occurs into the frequency domain, the encrypted images are directly in JPEG format, and could be uploaded to image-sharing platforms. However, to our knowledge, this solution has not been tested on existing image-sharing platforms.

2.3.2.3 Selective Encryption

In the field of image encryption, a classical approach is the *selective encryption* (SE). The main idea of this approach is to encrypt only the significant parts of the image, which is pratically sufficient to protect most visually important information of the image or video [PU02]. The objective is to reduce the computation time or power during the image processing, while preventing unauthorised users to access the full image. The selective encryption can be done in the spatial domain or in the frequency domain [VDB02].

- **In the spatial domain**

  In case of selective encryption in spatial domain, a popular technique is the use of bitplanes. A 8-bit grayscale image can be decomposed into 8 bitplanes (Figure 2.14), from the least significant bitplane to the most significant bitplane. One or some of bitplanes can be chosen to be encrypted, and these encrypted bitplanes with other original bitplanes compose the encrypted image.

  Droogenbroeck and Benedett [VDB02] proposed to encrypt the least significant bitplanes using the XOR function (with a random bitstream). Conversely, in [PSU02], given that the most significant bits contain more information (see Table 2.2 for the percentage of information provided by each bit [ZZWY11]), the authors proposed to encrypt the most significant bitplanes using AES encryption. However, in both case, the encryption only adds noise to the image, because only some of the bitplanes are encrypted. In particular, Droogenbroeck and Benedett [VDB02] have shown that even if 7 over the 8 bitplanes are encrypted the image is still recognisable, and so it is easy to differentiate two encrypted images (i.e., the encryption scheme is not IND-CPA secure).

- **In the frequency domain**

  According to the section 2.2.3, we know that after DCT transform, in each 8×8 blocks, the coefficients in the upper left corner are the low frequencies, where the most important visual characteristics of the image are placed. In contrast, the highest frequencies are in the lower right corner and correspond to the details of the image. Since the human eye is most sensitive

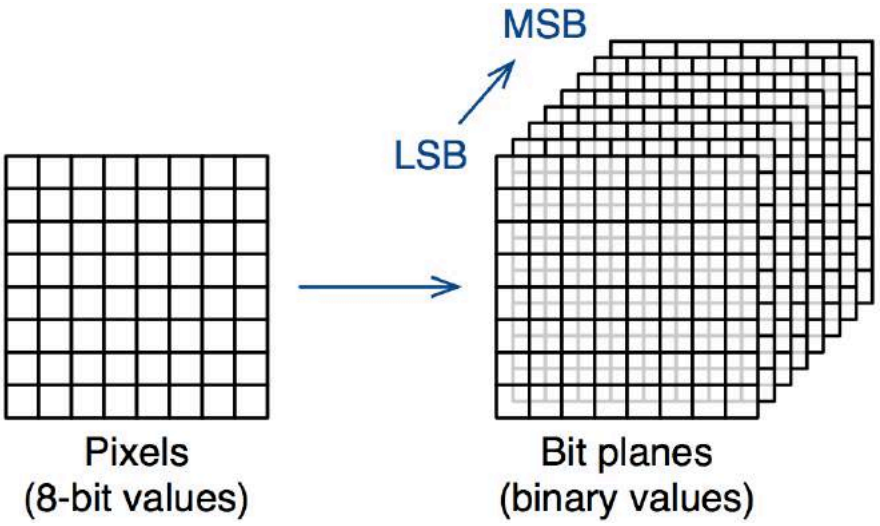


Figure 2.14: Decomposing an image into 8 bitplanes

Bit position in the pixel	Percentage of the pixel information (%)
1 (LSB)	0.3922
2	0.7843
3	1.5686
4	3.137
5	6.275
6	12.55
7	25.10
8 (MSB)	50.20

Table 2.2: The percentage of pixel information provided by different bits.

to lower frequencies than to higher frequencies [FMBD08], some researchers have proposed a selective encryption of DCT coefficients. For instance, in 1997, Kunkelmann *et al.* [KR] have proposed to encrypt partial DCT coefficients to protect the JPEG-based video.

Considering that the DC coefficients are highly predictable, Droogenbroeck and Benedett [VDB02] proposed to encrypt only the AC coefficients. From their experimental results, we can see that the resulting image is still recognisable. More generally, in [PR+05], the authors compared the experimental results of encrypting DCT coefficients in low frequencies and in high frequencies. If only the DC coefficients are encrypted, the image seems to be encrypted. However, it is very easy to access some visual information by simply replacing the ciphered DC coefficients with a constant value. If we encrypt the DC coefficients and some of the AC coefficients, the result is better, but the rest unencrypted AC coefficients makes it easy to differentiate two encrypted images (i.e., the encryption scheme is not IND-CPA secure).

In [KJK10], the authors proposed to not only encrypt DC and first few AC coefficients but also further divided the DC coefficient into bitplanes. In this way, these bitplanes can be encrypted gradually in order to get more levels for perceptual encryption. However, as stated before, the rest unencrypted AC coefficients makes it easy to differentiate two encrypted images (i.e., the encryption scheme is not IND-CPA secure).

Thus, the selective encryption can be done in the spatial domain or in the frequency domain but in both case, the solution is not IND-CPA secure since the parts of the images that are not encrypted make it easy to differentiate two encrypted images.

2.3.2.4 Combining Scrambling and Encryption

As stated before, the scrambling is not IND-CPA secure because the pixel or the DCT coefficient values are not changed. In order to improve the scrambling, a solution consists in using encryption algorithms to encrypt the pixel or the DCT coefficient values.

- **In the spatial domain**

  Guan, Huang and Guan [GHG05] proposed to use a chaotic system to realise both scrambling and encryption. More specifically, they use a Chaotic system to permute the pixels of image, as well as to provide bitstream to be XORed with the pixel values. In [GC08], the authors proposed a very similar solution. Wang et al. [WWLC11] proposed to shuffle blocks of pixels as well as to change the pixel values based on chaotic maps. First, the image is partitioned into 8×8 pixels blocks with adjustment. Then in each block, the 64 pixel values are changed by the iteration result of chaotic maps, and the new position of the block is calculated at the same time. The main drawback of these three solutions is that their security is based on chaotic system, whereas the security of such systems is still today an open issue. In particular, these systems can not be proved to be IND-CPA secure.

  In [BYJ08], the authors proposed to scramble the image based on a random permutation, and then to encrypt the pixels using the Blowfish algorithm, which is one of famous symmetric-key block cipher. The proposed solution allows to decrease the correlation between elements of the original image since the block sizes are randomised. Moreover, the solution could eventually be proved to be IND-CPA secure [4], even if the authors do not prove it. As such, this solution is a secure approach to protect privacy of user's images. However, given that image-sharing platforms generally prefer JPEG format, this solution is not adapted to our context. Indeed, the encryption occurs on the pixel values, and the encrypted images will be compressed before download. Since the compression is a lossy process, the decryption of the downloaded images will not be possible.

- **In the frequency domain**

  In [LDL12], the authors proposed to permute each 8×8 DCT block based on a chaotic map, and to encrypt the DC coefficient of each block. More specifically, the DC coefficients are extracted and rearranged into a matrix, on which is performed a dot product operation with a random matrix to obtain the encrypted DC coefficients. Since only DC coefficients are encrypted, the solution is a kind of selective encryption (see section §2.3.2.3), and thus, is not IND-CPA secure, even if it mixes scrambling and encryption.

  In [AGA12], the authors proposed to permute all the DCT coefficients first, and then to XOR the scrambled DCT coefficients with a random image. Thus, the solution could eventually be proved to be IND-CPA secure [5], even if the authors do not prove it. As such, this solution could be used to protect privacy of user's images published on existing image-sharing platforms. However, to our knowledge, the solution has not been tested on image-sharing platforms.

2.3.3 Protecting privacy of online published images

In previous section, we have presented the encryption algorithms that allow to encrypt images while preserving the image format but without really consider the publication of the resulting encrypted images. In this section, we introduce some works that specifically consider the protection of user's privacy when they publish their images.

Yuan *et al.* [YKE15] proposed a solution that consists in using a trustworthy server to store the protected image, and to simply publish on Facebook the protected image along with a URL pointing to that trustworthy server. Thus, the protected image is always stored in the trustworthy server. When the authorised users click the published image, according to the URL, they can obtain the original image from the server directly. Therefore, this approach does not need to treat

[4] Probabilistic encryption using the Blowfish algorithm is IND-CPA secure.
[5] The XOR operation with a random bitstream is by definition IND-CPA$ secure.

the processing on different image-sharing platforms, but only needs a dedicated server for images, and thus needs important storage and network resources, which is not satisfactory from our point of view.

In P3 [RGO13], the authors propose to divide the images into two parts: a public part and a secret part. The secret part contains the most important information, i.e. the most significant bits of significant coefficients. This part is encrypted and shared directly between sender and authorised recipients using a trustworthy server. The rest of information is left in the public part and can be published on any existing image-sharing platforms. The recipient can decrypt the secret part and download the public part, then combine these two parts. In this way, most important image contents can be reconstructed lossless by decrypting the secret part. Besides the fact that this solution requires, as the previous one, a dedicated server for storing the private part of the images, we can notice that the public parts of two distinct images allow easily to differentiate the original images: thus, this solution is not IND-CPA secure.

Another recent work is X-pire [BGLL14], that uses steganography. In this scheme, Backes et al. proposed to encrypt an image first and embed the encrypted data into a container image. The container image is in JPEG format and has the image size expected by the platforms. Then depending on the standard quantization table of each platform, they find the maximum quantization value. Then they are able to deduce the number of bits that can be embedded into each pixels. For example, if the maximum value is 36, its binary representation is 00100100, and we can deduce that a division by 36 will not change the two most significant bits: thus we can use these two bits to embed the encrypted image. Thus, the resulting image can be uploaded on image-sharing platform, and the encrypted image embedded into the uploaded image can be retrieve from the downloaded image, and its decryption allows to retrieve the *plaintext* image. However, the main drawback is of that approach is that only a few bits of information can be embedded into each pixel. For instance, if we consider that an average of 2 bits can be embedded into each pixel, there is only 25% bits can be used to embed.

Cryptagram [TSBS13] is a solution similar to the use of steganography, but instead of embedding the encrypted image into another image, an image is built from the result of the encryption. More specifically, the *plaintext* image is firstly encrypted with AES and an encrypted base-64 sequence is obtained. Each value in this sequence can be represented by two 3-bits binary values. Then, the interval of gray scale 0-255 is divided into eight sub-intervals, and each 3-bits represents the medium value of the corresponding sub-interval. So each base-64 value in the encrypted sequence can be mapped to two color values as two pixels of the container image. Assuming that JPEG lossy compression will preserve these sub-intervals, the resulting image can be published on image-sharing platform. Otherwise, the authors proposed to use Reed-Solomon codes to improve the result and to recover the plaintext image efficiently. As for the use of steganography, the main drawback of this approach is that each pixel of the resulting image can only encode a few bits of information. For instance, if we want to publish a B&W image as the container image on Facebook or Google+, the maximum upload dimension is 2048 x 2048 pixels, so for this scheme that attains an efficiency of 3-bits per pixels, only 1.5MB data can be embedded at most.

2.3.4 Summary

In the table 2.3, we summary the related works we mentioned above. We classify them according to their method to encrypt images, and for each method, we specify if it preserves the image format, if it is IND-CPA Secure, and if it fits with image-sharing platforms. In this table, we can see that only three solutions both are IND-CPA secure and can be used on image-sharing platforms [YKE15, TSBS13, BGLL14] . However, [YKE15] is based on a trustworthy server, which is not satisfactory from our point of view, and [TSBS13, BGLL14] result in significantly limiting the size

of the images when encrypted.

Method	References	Performance		
		Format preserving	IND-CPA Secure	Image-sharing Platform
Traditional encryption algorithm	[DC00], [Dey12]	×	√	×
Scrambling in the spatial domain	[WFL15], [UJN+07], [PP12]	√	×	√
Scrambling in the frequency domain	[KE14]	√	×	√
Modify the traditional encryption algorithm	[YPWSp+09], [ZMK+07], [KSHR10], [SZS04]	√	√	×
Selective encryption in spatial domain	[VDB02], [PSU02]	√	×	√
Selective encryption in frequency domain	[KR], [VDB02], [PR+05], [KJK10]	√	×	√
Scrambling and encryption in spacial domain	[GHG05], [GC08], [WWLC11], [BYJ08]	√	√	×
Scrambling and encryption in frequential domain	[LDL12]	√	×	√
	[AGA12]	√	*	√
Based on trustworthy server	[RGO13]	√	×	√
	[YKE15]	√	√	√
Steganography	[TSBS13], [BGLL14]	√	√	√

Table 2.3: The comparison of related works.

* Possibly but should be proved.

2.4 Conclusion

In this chapter, we have introduced the basic theory of modern cryptography. From the theoretic conceptions to the practical applications, we have a general understanding of modern cryptography.

We have also introduced the basic knowledge of digital image processing. We have presented how to represent an image by using different color models. Moreover, we have introduced the Discrete Cosine Transform which is a very important step in the coding process of JPEG standard. Considering JPEG is the widely used standards for storing and compressing images, and that most of image-sharing platforms are compressed by using JPEG standard, we have detailed the JPEG compression processes.

Because the main objective of this thesis is to help the users to protect their privacy on the existing image-sharing platforms by allowing them to encrypt their images, we have then presented some related works about image encryption. The major existing solutions are either not IND-CPA secure, or not compatible with publishing on image-sharing platforms. The only solutions that are both IND-CPA secure and compatible with image-sharing platform either are based on a trustworthy server or have a significant impact on the size of the images when encrypted.

In this thesis, we propose a IND-CPA secure encryption algorithm that preserves the JPEG format so as to be able to publish the encrypted images on existing image-sharing platforms. Our encryption algorithm is a symmetric stream cipher : based on a PRNG, we generate a pseudo-random numbers sequence from a seed and the secret key, this sequence being used to encrypt the DC and AC coefficients in frequency domain. Besides preserving the JPEG format, we have to face another challenge. Indeed, as [TSBS13, BGLL14] have found, some of the existing image-sharing platforms perform post-uploading processing even if the uploaded images are in an appropriate format. And this post-uploading processing partially modifies the DCT coefficient values of the uploaded images, thus making impossible to correctly decrypt the downloaded image if the encryption algorithm does not have appropriate mechanisms.

Using the proposed encryption algorithm, we define an architecture that allows users to share their symmetric keys, and thus to specify access rights. The proposed architecture is based on a trustworthy server that plays the role of Certification Authority and that is in charge of storing the encrypted keys. Each user generates his/her own public and private keys and asks the trustworthy server to certify its public key. When he/she publishes a encrypted image, he/she encrypts the used secret key with the public keys of his/her friends to whom he/she want to grant access right.

Chapter 3

The basic encryption algorithm

Contents

In order to protect user's privacy on the existing sharing platforms, we aim to encrypt the image before publishing it. However, according to the existing methods we introduced in 2.3, the encrypted result has to be viewed as a correct digital format by the image-sharing platforms. Therefore, in this chapter, we propose an encryption algorithm and implement it at three different steps of the JPEG compression process. All of them can preserve the JPEG image format after encryption and we prove that this algorithm is IND-CPA secure.

3.1 The algorithm description

The main idea is to take all the coefficients that have to be encrypted to construct a sequence. Then encrypt it using a pseudo-random sequence of the same length which is generated by the function *prf*, and put the encrypted stream back.

Our encryption algorithm is symmetric. We use a pseudo-random function *prf* which takes a value n as input and returns a pseudo-random value in the range $[0, n[$. This function is associated with an initialization function prf_{init} that takes values *key* and *seed* as inputs, and allows to initialize the function *prf*.

There are three functions in our encryption algorithm:

Key generation: $key = KeyGen(\lambda)$ takes λ as input and returns a random value *key* of λ bits.

Encryption: $C = Enc(I, key, seed)$ uses *key* and a random value *seed* to initialize the pseudo-random function and generates a pseudo-random sequence. Then encrypts an image I using this sequence. The result is encrypted image C.

Decryption: $I' = Dec(C, key, seed)$ decrypts the encrypted image C using the pseudo-random sequence which is initialized by *key* and the random value *seed*.

In the following subsections, we detail the encryption and decryption functions .

3.1.1 Encryption

The plaintext image I is in JPEG format. First of all, the function $prf_{\text{init}}(key, seed)$ initializes the pseudo-random function prf using random values key and $seed$.

1. Suppose that xc denotes the coefficient we want to encrypt and that the range of xc is $[-xc_{\min}, +xc_{\max}]$. We note $n_{xc} = xc_{\min} + xc_{\max} + 1$.
2. Each coefficient xc is encrypted as follow:

$$e_{xc} = (((xc + xc_{\min}) + prf(n_{xc})) \mod n_{xc}) - xc_{\min} \tag{3.1}$$

From the result, we make sure that each encrypted coefficient e_{xc} falls in the range $[-xc_{\min}, +xc_{\max}]$.

Finally, by performing the above operations and the rest of JPEG compression processes, the encrypted image C in JPEG format is obtained.

3.1.2 Decryption

First of all, we use the function $prf_{\text{init}}(key, seed)$ with the two random values key and $seed$ to initialize the pseudo-random function prf.

1. Each encrypted coefficient e_{xc} falls in the range $[-xc_{\min}, +xc_{\max}]$. We note $n_{xc} = xc_{\min} + xc_{\max} + 1$.
2. e_{xc} is decrypted as follow:

$$d_{xc} = (((e_{xc} + xc_{\min}) + (n_{xc} - prf(n_{xc}))) \mod n_{xc}) - xc_{\min} \tag{3.2}$$

Finally, by performing the above operations and the rest of JPEG compression processes, the decrypted image I' is obtained.

We can prove that, the decrypted image I' is equal to original image I, in other words, for all xc, we have $d_{xc} = xc$. The proof is as follow:

Proof.

$$\begin{aligned}
d_{xc} =& (((e_{xc} + xc_{\min}) + (n_{xc} - prf(n_{xc}))) \mod n_{xc}) - xc_{\min} \\
=& ((((((xc + xc_{\min}) + prf(n_{xc})) \mod n_{xc}) - xc_{\min} + xc_{\min}) \\
&+ (n_{xc} - prf(n_{xc}))) \mod n_{xc}) - xc_{\min} \\
=& ((xc + xc_{\min} + prf(n_{xc}) + n_{xc} - prf(n_{xc})) \mod n_{xc}) \\
&- xc_{\min} \\
=& ((xc + xc_{\min} + n_{xc}) \mod n_{xc}) - xc_{\min} \\
=& ((xc + xc_{\min}) \mod n_{xc}) - xc_{\min}
\end{aligned} \tag{3.3}$$

Because $xc \in [-xc_{\min}, xc_{\max}]$ and $n_{xc} = xc_{\min} + xc_{\max} + 1$,
therefore, $xc + xc_{\min} \in [0, n_{xc}[$. Then we have $(xc + xc_{\min}) \mod n_{xc} = xc + xc_{\min}$. Thus we can prove that $d_{xc} = xc + xc_{\min} - xc_{\min} = xc$. □

3.1.3 Security of the Encryption Algorithm

In our encryption algorithm, we use prf_{init} and prf functions to constitute a Pseudo Random Number Generator (PRNG). The prf_{init} function initializes the PRNG and prf permits to have pseudo random numbers from a long sequence of pseudo random bits thanks to prf_{init}.

Suppose that our PRNG is secure, which means that we cannot distinguish between the generated sequence of pseudo random bits and a sequence of real random bits of the same size.

In our encryption algorithm, we initialise the PRNG with $prf_{\text{init}}(key, seed)$ where key and $seed$ are real random numbers. Then, for each coefficient xc of the image, we encrypt it as Eq. (3.1), e_{xc} is the encrypted lcoefficient.

Theorem. *If prf is secure then when our encryption algorithm encrypts one* 8×8 *block of image, it is IND-CPA secure.*

Proof. If *prf* is a secure PRNG, we cannot distinguish between $prf(n_{xc})$ and a real random number according to the definition of a secure PRNG. Therefore, we can replace $prf(n_{xc})$ by a real random number r in Eq. (3.1), and the encryption equation becomes $e_{xc} = (((xc + xc_{\min}) + r) \mod n_{xc}) - xc_{\min}$ where r is a real random number. This change affects the algorithm in a negligible manner.

Operations based on a real random number give a random number as result. We have $e_{xc} = r'$ where r' is a random number. This change does not affect the algorithm.

We proved that, if *prf* is secure, encrypting a coefficient is equivalent to return a random number. That is the definition of IND-CPA$ secure. In the general case, the image is processed block by block (the typical block size is 8×8). So our algorithm to encrypt the coefficients in each 8×8 block is IND-CPA$ secure. Because the attacker cannot distinguish the coefficients in one block are encrypted or are random numbers, therefore the attacker cannot distinguish the block is encrypted or this is a random block.

Besides, if an encryption algorithm is IND-CPA$ secure, then it is IND-CPA secure. Therefore, our encryption algorithm is IND-CPA secure to encrypt a 8×8 block of one image. □

In the next section, we detail how to integrate this algorithm into three different steps of the JPEG compression process, in order to ensure that the encrypted results are JPEG image, which can be accepted by any image-sharing platforms.

3.2 Integration of encryption in JPEG compression process

As we presented above, we encrypt JPEG image and get a JPEG image as result. According to us, the coefficients we want to encrypt can be in any steps of JPEG compression process. Based on the concept of JPEG encoding in Sec 2.2.3, we implement our encryption algorithm at three different steps of the JPEG compression process. The first one encrypts image before DCT in spatial domain. The second one encrypts image after DCT in frequency domain. The third one encrypts image after quantization. We detail them in this section, and experimental results are presented to compare the three schemes. We use LibJPEG [G+98] to encode and decode JPEG image, and all experiments were implemented in C/C++. As examples, we first take the grayscale and the colour image "Lena" of 512×512 pixels to do a detailed experiment, and then we take 10 grayscale images and 10 colour images of different sizes to do more tests.

We use the algorithm proposed in Sec. 3.1 to encrypt Y, Cb,Cr components respectively and the method to encrypt each of them is the same as to encrypt a grayscale image. Therefore, in the next subsections, we concisely present how to encrypt a grayscale image. In all experiments, we choose two quantization tables to compress image with compression ratio Q=71 and Q=100 (as shown in Fig. 3.1). We compare their runtime, Peak Signal to Noise Ratio (PSNR) and Universal Image Quality Index (UIQI) [WB02], which are techniques to measure the image quality. PSNR is defined via the Mean Squared Error (MSE), and it is worth mentioning that in all of our experiments, for colour images the MSE is calculated by the sum of all squared value differences divided by image size and then by three. For the UIQI calculation of colour image, we first convert the colour image to grayscale image, and then use the defined formula to calculate its UIQI value. All the results of "Lena" are summarized in Tab. 3.1 and Tab. 3.2, the average results of other images are explained in Tab. 3.3.

Notice that the plaintext images are already in JPEG format, so we have to first decompress the image to retrieve the bitmap.

3.2.1 Encryption Before DCT

In this scheme, we encrypt the pixels of image in spatial domain. We first decompress the image just after the reverse DCT and apply the encryption. The range of coefficients we want to encrypt is $[-128, 127]$. Because in spatial domain, there is no energy compaction property, if we only encrypt a part of pixels in each block, the unencrypted parts are always visible (shown in Fig. 3.2). Therefore, in order to ensure the security, we need to encrypt all the coefficients of the image using

	Q	Runtime (ms)		PSNR (dB)		UIQI	
		Encryption	Decryption	Compression without encryption	After decryption	Compression without encryption	After decryption
Encryption before DCT	71	89	64	47.7	33	0.9833	0.7054
	100	71	61	Inf	55	1	0.9912
Encryption after DCT	71	35	32	47.7	34.7	0.9833	0.8715
	100	31	31	Inf	35.1	1	0.9313
Encryption after quantization	71	19	10	47.7	35	0.9833	0.9217
	100	29	13	Inf	35.1	1	0.9313

Table 3.1: Summary of the results for grayscale image "Lena".

	Q	Runtime (ms)		PSNR (dB)		UIQI	
		Encryption	Decryption	Compression without encryption	After decryption	Compression without encryption	After decryption
Encryption before DCT	71	186	171	38.4	29.6	0.9154	0.6834
	100	198	151	Inf	49.9	1	0.991
Encryption after DCT	71	73	65	38.4	33.3	0.9154	0.8279
	100	82	64	Inf	35	1	0.8951
Encryption after quantization	71	44	21	38.4	34.4	0.9154	0.8752
	100	25	22	Inf	35	1	0.8951

Table 3.2: Summary of the results for color image "Lena".

	Q	Grayscale image		Color image	
		Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Compression without encryption	71	45.1	0.9561	42.3	0.9245
	100	Inf	1	Inf	1
Encryption before DCT	71	16.4	0.4283	18.8	0.4641
	100	29.3	0.9026	35.4	0.9307
Encryption after DCT	71	28.2	0.72	30.1	0.7059
	100	30.9	0.8966	33.2	0.8937
Encryption after quantization	71	30.8	0.8598	33	0.8469
	100	30.9	0.8966	33.2	0.8937

Table 3.3: Summary of the results for 20 other images.

$$\begin{bmatrix} 9 & 6 & 6 & 9 & 14 & 23 & 30 & 35 \\ 7 & 7 & 8 & 11 & 15 & 34 & 35 & 32 \\ 8 & 8 & 9 & 14 & 23 & 33 & 40 & 32 \\ 8 & 10 & 13 & 17 & 30 & 50 & 46 & 36 \\ 10 & 13 & 21 & 32 & 39 & 63 & 60 & 45 \\ 14 & 20 & 32 & 37 & 47 & 60 & 66 & 53 \\ 28 & 37 & 45 & 50 & 60 & 70 & 70 & 59 \\ 42 & 53 & 55 & 57 & 65 & 58 & 60 & 57 \end{bmatrix}$$

(a) $Q = 71$.

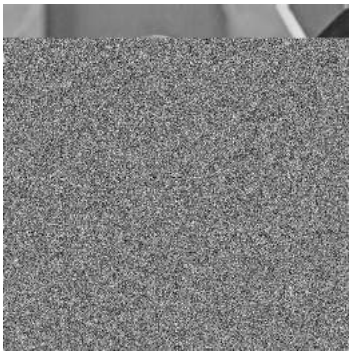
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(b) $Q = 100$.

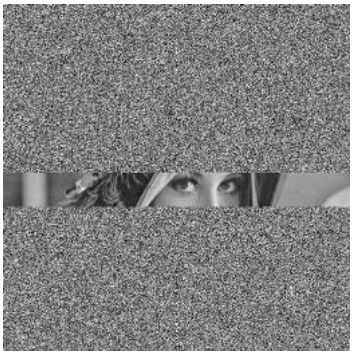
Figure 3.1: Quantization tables of grayscale image or luminance component of color image.

our encryption algorithm in spatial domain. Then we finish the compression process and obtain the encrypted image in JPEG format.

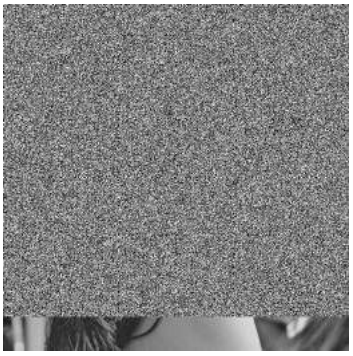
(a) Original image



(b) Unencrypted first 50 lines



(c) Unencrypted middle 50 lines



(d) Unencrypted last 50 lines

Figure 3.2: Encrypt only parts of image before DCT.

During decryption, we decompress encrypted image just after the reverse DCT, and decrypt all coefficients. Then we finish the compression process and obtain the decrypted image.

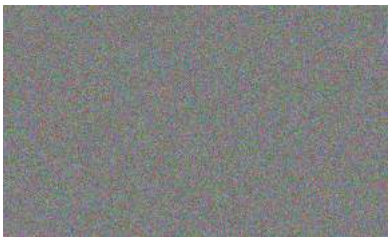
Fig. 3.3 shows an example of the resulting encrypted and decrypted images with $Q = 71$. A part of the decrypted image is zoomed, and we notice that there are a lot of little specks distributed in the images. According to our experiments of other 20 images, we find that there are a large number of specks in some images.

As shown in Tab. 3.1 and Tab. 3.2, if Q=100, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 55 dB and with UIQI of 0.9912, and reconstruct the plaintext color image with PSNR of 49.9 dB and with UIQI of 0.991. They are very high values which mean the decrypted images are almost the same as the plaintext images. For $Q = 71$, the value of PSNR is 33 dB for grayscale image and 29.6 dB for color image, which means there are perceived distortions, but the quality is acceptable. The value of UIQI is 0.7054 for grayscale image and 0.6834 for color image, which means based on the human visual system (HVS), the decrypted

image is similar to the original one, but has some distortions. If the original image is compressed with compression ratio $Q = 71$, the PSNR value is 47.7 dB for grayscale image and 38.4 dB for color image, the UIQI value is 0.9833 for grayscale image and 0.9154 for color image. Comparing them with the values of PSNR and UIQI after decryption, there is a gap but not too large. The runtime of this scheme is around 60∼90 ms for grayscale image and 150∼200 ms for color image on an Intel i7 laptop, which is already a very efficient encryption.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3.3. If $Q = 100$, the average PSNR value of 10 grayscale images is 29.3 dB, which is not a high value, but the average UIQI value is 0.9026, meaning that the visual quality of these images is high. For 10 color images the average PSNR value is 35.4 dB and the average UIQI value is 0.9307, which means the visual quality is high. If $Q = 71$, for 10 grayscale images the average PSNR value is 16.4 dB, which is extremely poor, and the average UIQI value is 0.4283 which is only acceptable. For 10 color images the PSNR value is 18.8 dB and the UIQI value is 0.4641, which also means the quality is just acceptable. If the original image is only compressed with compression ratio $Q = 71$, the average PSNR value is 45.1 dB for 10 grayscale images and is 42.3 dB for 10 color images; the average UIQI value is 0.9561 for 10 grayscale images and is 0.9245 for 10 color images. Comparing them with the value of PSNR and UIQI after decryption, the gap is large.

(a) Original image



(b) Encrypted image

(c) Decrypted image

Figure 3.3: Experimental results of encryption before DCT.

Therefore, we conclude that encryption before DCT cannot reconstruct the plaintext image with a high quality. And what's more, because in each block, we have to encrypt all the 64 coefficients to ensure the security, so this scheme is not efficient enough.

3.2.2 Encryption After DCT

Then we consider to integrate the encryption into the frequency domain. The first choice is the encryption after DCT. In this scheme, we first decompress the image just before the reverse DCT. The image is split into 8×8 blocks. If $X_{i,j}$ denotes a DCT coefficient of one 8×8 block ($i, j \in [0, 7]$), the two-dimensional 8×8 DCT transform can be expressed as $X = F \times x \times F^T$,

where $x_{u,v} \in [-128, 127]$ $(u, v \in [0, 7])$ is the coefficient before DCT, and F is a matrix:

$$\begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \cdots & \frac{1}{2\sqrt{2}} \\ \frac{1}{2}cos\frac{\pi}{16} & \frac{1}{2}cos\frac{3\pi}{16} & \frac{1}{2}cos\frac{5\pi}{16} & \cdots & \frac{1}{2}cos\frac{15\pi}{16} \\ \frac{1}{2}cos\frac{2\pi}{16} & \frac{1}{2}cos\frac{6\pi}{16} & \frac{1}{2}cos\frac{10\pi}{16} & \cdots & \frac{1}{2}cos\frac{30\pi}{16} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}cos\frac{7\pi}{16} & \frac{1}{2}cos\frac{21\pi}{16} & \frac{1}{2}cos\frac{35\pi}{16} & \cdots & \frac{1}{2}cos\frac{105\pi}{16} \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ F_7 \end{bmatrix}$$

F_i is the vector composition of each row of F. Then $X_{i,j} = F_i \times x \times F_j^T = x \cdot G_{i,j}$, where $G_{i,j}$ denotes $F_i^T \times F_j$, and it is a constant matrix [1]. Because the range of $x_{u,v}$ is $[-128, 127]$, so if we want to obtain the maximum value of $X_{i,j}$, we define the element of x in corresponding position as 127 when the element of $G_{i,j}$ is positive, and as -128 when the element of $G_{i,j}$ is negative. On the contrary, to get the minimum value, we define the corresponding element of x as -128 when the element of $G_{i,j}$ is positive, and as 127 when the element of $G_{i,j}$ is negative. In this way, we calculate the range of all 8×8 coefficients (shown in Fig. 3.4). In the following, the range of DCT coefficients we want to encrypt $X_{i,j}$ is $[Xmin_{i,j}, Xmax_{i,j}]$.

$$\begin{bmatrix} -1024 & -924.25 & -942.36 & -924.25 & -1020 & -924.25 & -942.36 & -924.25 \\ -924.25 & -837.49 & -853.9 & -837.49 & -924.25 & -837.49 & -853.9 & -837.49 \\ -942.36 & -853.9 & -870.62 & -853.9 & -942.36 & -853.9 & -870.62 & -853.9 \\ -924.25 & -837.49 & -853.9 & -837.49 & -924.25 & -837.49 & -853.9 & -837.49 \\ -1020 & -924.25 & -942.36 & -924.25 & -1020 & -924.25 & -942.36 & -924.25 \\ -924.25 & -837.49 & -853.9 & -837.49 & -924.25 & -837.49 & -853.9 & -837.49 \\ -942.36 & -853.9 & -870.62 & -853.9 & -942.36 & -853.9 & -870.62 & -853.9 \\ -924.25 & -837.49 & -853.9 & -837.49 & -924.25 & -837.49 & -853.9 & -837.49 \end{bmatrix}$$

(a) The minimum values

$$\begin{bmatrix} 1016 & 924.25 & 942.36 & 924.25 & 1020 & 924.25 & 942.36 & 924.25 \\ 924.25 & 837.49 & 853.9 & 837.49 & 924.25 & 837.49 & 853.9 & 837.49 \\ 942.36 & 853.9 & 870.62 & 853.9 & 942.36 & 853.9 & 870.62 & 853.9 \\ 924.25 & 837.49 & 853.9 & 837.49 & 924.25 & 837.49 & 853.9 & 837.49 \\ 1020 & 924.25 & 942.36 & 924.25 & 1020 & 924.25 & 942.36 & 924.25 \\ 924.25 & 837.49 & 853.9 & 837.49 & 924.25 & 837.49 & 853.9 & 837.49 \\ 942.36 & 853.9 & 870.62 & 853.9 & 942.36 & 853.9 & 870.62 & 853.9 \\ 924.25 & 837.49 & 853.9 & 837.49 & 924.25 & 837.49 & 853.9 & 837.49 \end{bmatrix}$$

(b) The maximum values

Figure 3.4: The range of DCT coefficients in one block.

The encryption is implemented block by block and each block is represented as a matrix. Remember that the human eye is more sensitive to lower frequencies than to higher frequencies [FMBD08]. Therefore, we try to only encrypt the lower frequency coefficients to reduce the calculation. But we need to do some test to decide how many coefficients to encrypt.

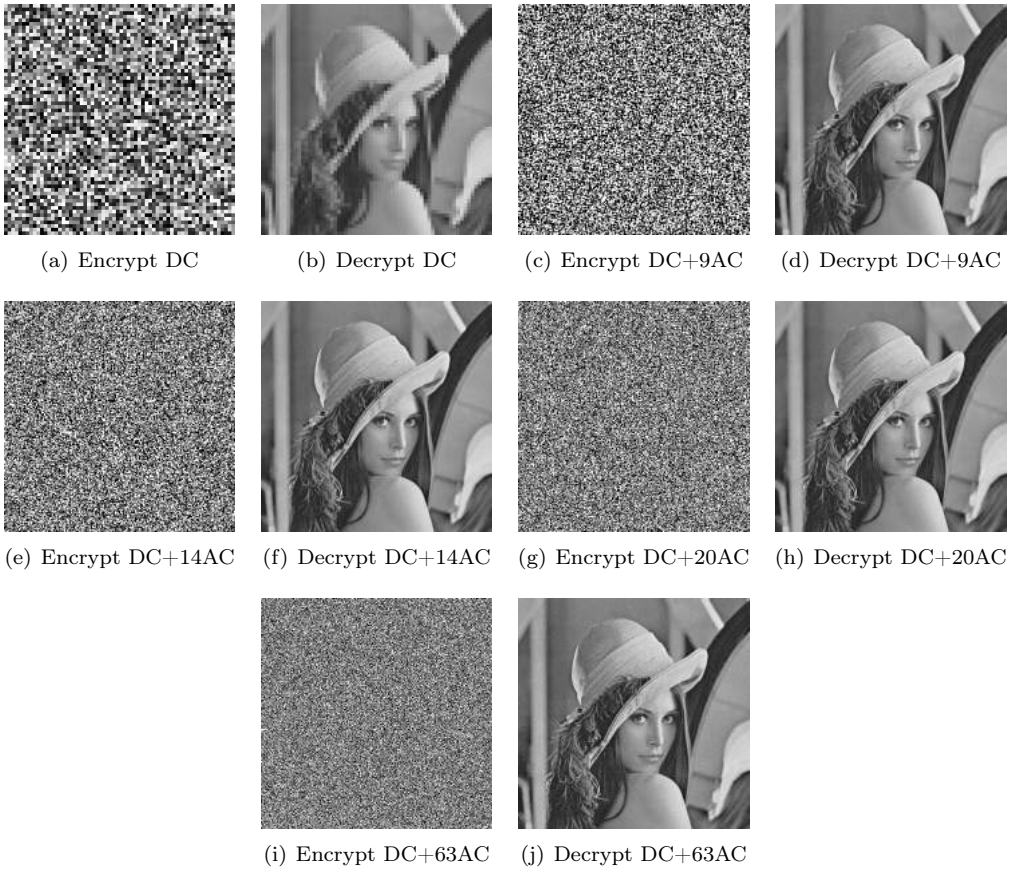
When we do the test, we encrypt selective coefficients and set the remaining coefficients to 0, in order to ensure that the attackers can only obtain encrypted coefficients. After encryption, the encrypted DCT block is quantized and entropy encoded, and the compression quality impact the results of encryption and decryption because of the roundoff error of quantization. During decryption, we decompress encrypted image just before the reverse DCT, and only decrypt the DC coefficient and the chosen AC coefficients according to the "zigzag" order block by block. Here, we choose $Q = 71$ and $Q = 100$ to do the test. We list some typical encryptions in Tab. 3.4 about their runtime, PSNR and UIQI values through comparing the original image and the decrypted image.

First, we only encrypt DC coefficient (results are shown in Fig. 3.5(a), 3.5(b)). The encrypted result is obscure, but the recovered image has a very poor quality (PSNR value is only 23.7 dB). And what's more, considering that DC coefficient contains the most relevant information and is

[1] "." means dot product: it is the sum of the products of the corresponding elements of the two matrices.

Encrypted coefficients	Q=71				Q=100			
	Runtime (ms)		PSNR (dB)	UIQI	Runtime (ms)		PSNR (dB)	UIQI
	Enc	Dec			Enc	Dec		
DC	6.9	5.4	23.71	0.3224	7.2	5.4	23.71	0.3272
DC+2AC	7.5	6.0	27.44	0.6022	7.9	6.3	27.46	0.6215
DC+5AC	8.4	6.8	30.03	0.7281	8.3	7.3	30.12	0.7698
DC+9AC	9.7	8.0	32.19	0.7837	9.6	8.9	32.48	0.8670
DC+14AC	10.6	9.6	34.12	0.7867	11.0	10.5	35.12	0.9313
DC+20AC	12.3	12.0	34.52	0.7304	12.6	12.2	37.86	0.9681
DC+27AC	12.9	14.0	32.17	0.6159	14.0	14.5	40.94	0.9848
DC+44AC	16.7	19.0	26.05	0.4176	18.3	20.0	52.48	0.9970
DC+63AC	21.4	24.5	21.36	0.2841	23.6	25.2	Inf	1

Table 3.4: Summary of the results of encrypting different coefficients after DCT.



(a) Encrypt DC (b) Decrypt DC (c) Encrypt DC+9AC (d) Decrypt DC+9AC

(e) Encrypt DC+14AC (f) Decrypt DC+14AC (g) Encrypt DC+20AC (h) Decrypt DC+20AC

(i) Encrypt DC+63AC (j) Decrypt DC+63AC

Figure 3.5: Typical results of encrypting and decrypting different coefficients after DCT ($Q = 71$).

highly predictable, only encrypting DC coefficient is not secure. Therefore, we increase the number of encrypted AC coefficients. When we encrypt DC and the first 2 AC coefficients, the encrypted image is still invisible. However, we lose too many details in the recovered image. Then we encrypt more AC coefficients, until all 63 AC coefficients are encrypted. Some typical results are shown in Fig. 3.5. We can find that when we encrypt DC coefficient and 9 AC coefficients, the recovered image is already similar to the original image with enough details. Then if $Q = 100$, the more AC coefficients are encrypted, the better quality the recovered image has. But on the other hand, the more time it takes. If $Q = 71$, after encrypting 20 AC coefficients, the quality begins to reduce. Therefore, encrypting 9, 14 or 20 AC coefficients are the three best options.

While $Q = 71$, the PSNR value of encrypting 14 AC coefficients is better than encrypting 9 AC coefficients, but the UIQI value is almost the same. Although encrypting 14 AC coefficients takes 1 ms longer, considering encrypting more AC coefficients, we can keep more details of image, we prefer encrypting 14 AC coefficients. The PSNR value of encrypting 20 AC coefficients is better than encrypting 14 AC coefficients, but the UIQI value is worse. And encrypting 20 AC coefficients takes 2 ms longer. Therefore, considering the quality of recovered image and the efficiency of algorithm, we prefer encrypting 14 AC coefficients.

After the test, we decide only encrypt the DC coefficient and the first 14 AC coefficients according to "zigzag" order and set the remaining coefficients to 0 as described in matrix (3.4). In this way, we reduce the calculation and ensure the security. Then the encrypted DCT block is quantized with $Q = 71$ or $Q = 100$ and entropy encoded. The decryption is inverse operation.

$$\begin{bmatrix} e_{dc} & e_{ac1} & e_{ac5} & e_{ac6} & e_{ac14} & 0 & 0 & 0 \\ e_{ac2} & e_{ac4} & e_{ac7} & e_{ac13} & 0 & 0 & 0 & 0 \\ e_{ac3} & e_{ac8} & e_{ac12} & 0 & 0 & 0 & 0 & 0 \\ e_{ac9} & e_{ac11} & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{ac10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

Fig. 3.6 shows the resulting encrypted and decrypted images with $Q = 71$. A part of decrypted image is zoomed, and we find that there is few specks in the image. For other 20 images, we have the same conclusion.

(a) Original image

(b) Encrypted image

(c) Decrypted image

Figure 3.6: Experimental results of encryption after DCT.

As shown in Tab. 3.1 and Tab. 3.2, if compression ratio is $Q = 71$, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 34.7 dB and with UIQI of 0.8715, and reconstruct the plaintext color image with PSNR of 33.3 dB and with UIQI of 0.8279. They are higher than the PSNR and UIQI values of first scheme with $Q = 71$, which means the decrypted image is more similar to the plaintext image. If $Q = 100$, the value of PSNR is 35.1 dB for grayscale image and 35 dB for color image, the value of UIQI is 0.9313 for grayscale image and 0.8951 for color image. They are high values which mean the decrypted image is very similar to the plaintext image. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is smaller than the first scheme. The runtimes of encryption and decryption are around 30 ms for grayscale image and 60∼80 ms for color image, it takes less time than the encryption before DCT. So this scheme is more efficient than the first one.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3.3. If $Q = 100$, for 10 grayscale images the average PSNR value is 30.9 dB and the average UIQI value is 0.8966; for 10 color images the average PSNR value is 33.2 dB and the average UIQI value is is 0.8937. It means that the decrypted image is similar to the original one, but has some distortions. If $Q = 71$, for 10 grayscale images the average PSNR value is 28.2 dB and the average UIQI value is 0.72; for 10 color images the average PSNR value is 30.1 dB and the average UIQI value is 0.7059. They are higher than the PSNR and UIQI values of first scheme. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is smaller than the first scheme.

Therefore, we can conclude that this scheme can reconstruct the plaintext image with a higher quality than the first one. We only encrypt 15 coefficients in this scheme, so it is more efficient than encryption before DCT. But it still cannot reconstruct a highest quality image.

3.2.3 Encryption After Quantization

Another choice to integrate the encryption into the frequency domain is after quantization. In this scheme, we first decompress the image just before the dequantization and re-quantize it. The image is split into 8×8 blocks and the DCT coefficients of one 8×8 are quantized as following:

$$Q_{i,j} = round(\frac{X_{i,j}}{q(i,j)}) \tag{3.5}$$

Where $X_{i,j}$ are the unquantized DCT coefficients of one 8×8 block ($i, j \in [0, 7]$), $Q_{i,j}$ are the quantized DCT coefficients of one 8×8 block ($i, j \in [0, 7]$). The range of quantized coefficients is

$$\left[\text{round}\left(\frac{Xmin_{i,j}}{q(i,j)}\right), \text{round}\left(\frac{Xmax_{i,j}}{q(i,j)}\right)\right].$$

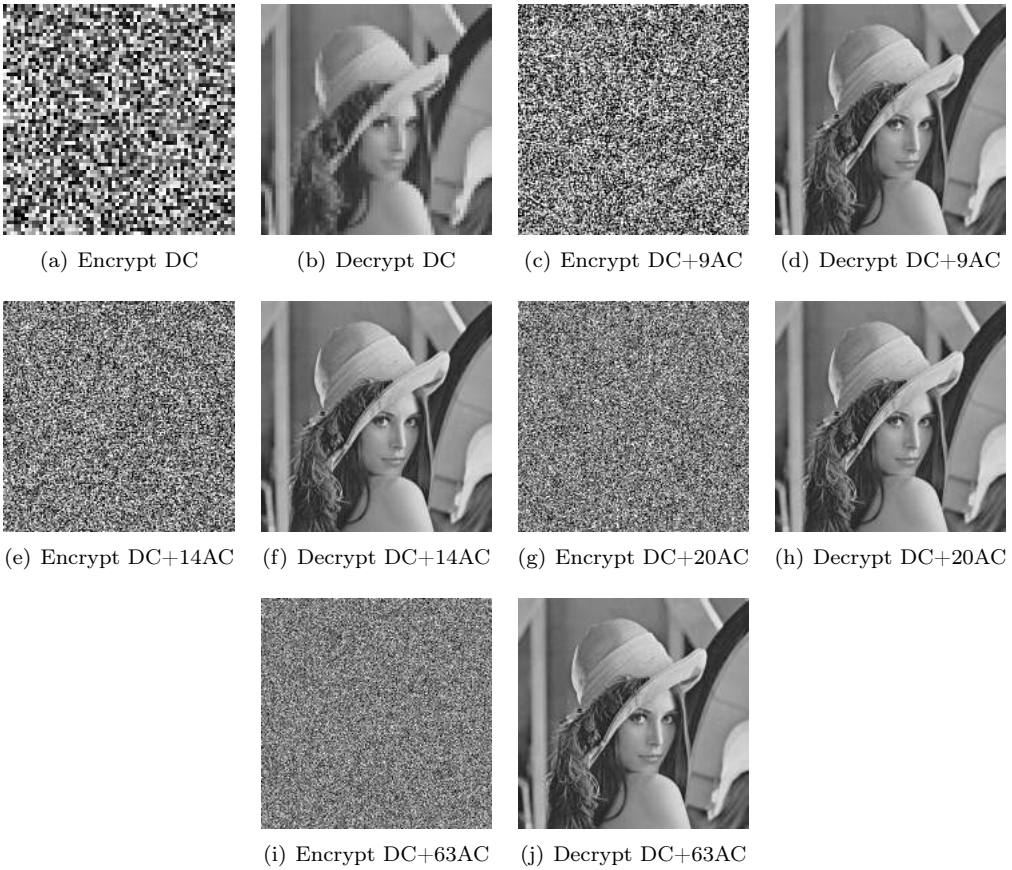
Where $q(i, j)$ are the elements of the quantization table ($i, j \in [0, 7]$). The encryption is implemented block by block and each block is represented as a matrix. As for the encryption after DCT, considering the energy compaction property, we need to do some test to decide how many coefficients we have to encrypt .

When we do the test, we encrypt selective coefficients and set the remaining coefficients to 0, in order to ensure that the attackers can only obtain encrypted coefficients. After encryption, the encrypted quantized DCT block is entropy encoded. The compression quality of re-quantization during the encryption impact the results. During decryption, we decompress encrypted image just before the dequantization, and only decrypt the DC coefficient and the first 14 AC coefficients according to the "zigzag" order block by block. Notice that, when the compression quality is $Q = 100$, this scheme is the same as the encryption after DCT. So, we only list some typical encryptions of $Q = 71$ in Tab. 3.5 about their runtime, PSNR and UIQI values through comparing the original image and the decrypted images.

According to the test results, we still find that only encrypting DC coefficient cannot provide the recovered image with a high quality. Therefore we increase the number of encrypted AC coefficients, from 1 to 63. Some typical results are shown in Fig. 3.7. We find that the more AC coefficients are encrypted, the better quality the recovered image has. But on the other hand, the more time it takes. After encrypting 9 AC coefficients, the recovered image is already similar to

Encrypted coefficients	Runtime (ms)		PSNR (dB)	UIQI
	Enc	Dec		
DC	7.5	2.1	23.71	0.3267
DC+2AC	8.2	2.9	27.46	0.6197
DC+5AC	8.3	3.2	30.10	0.7666
DC+9AC	9.5	4.2	32.44	0.8606
DC+14AC	11.0	6.2	35.00	0.9217
DC+20AC	12.3	6.1	37.56	0.9562
DC+27AC	14.0	7.9	40.26	0.9721
DC+44AC	17.7	9.4	46.67	0.9827
DC+63AC	22.3	12.6	47.71	0.9833

Table 3.5: Summary of the results of encrypting different coefficients after quantization (Q=71).



(a) Encrypt DC (b) Decrypt DC (c) Encrypt DC+9AC (d) Decrypt DC+9AC

(e) Encrypt DC+14AC (f) Decrypt DC+14AC (g) Encrypt DC+20AC (h) Decrypt DC+20AC

(i) Encrypt DC+63AC (j) Decrypt DC+63AC

Figure 3.7: Typical results of encrypting and decrypting different coefficients after quantization ($Q = 71$).

the original image with enough details. But considering the trade-off among the efficiency and the quality of recovered image, we prefer encrypting DC and the first 14 AC coefficients. Because although encrypting 14 coefficients takes 2 ms longer, the PSNR and UIQI values of encrypting 14 AC coefficients are 35.00 dB and 0.9217 which are better than encrypting 9 AC coefficients. For encrypting 20 coefficients, the PSNR and UIQI values are 37.56 dB and 0.9562, which are only a little higher than encrypting 14 coefficients. In fact, the recovered images of encrypting 14 coefficients have an enough good quality for us, and the encryption is more efficient.

After the test, we decide only encrypt the DC coefficient and the first 14 AC coefficients according to "zigzag" order and set the remaining coefficients to 0 as described in matrix (3.4).Then the encrypted quantized DCT block is entropy encoded. The decryption is inverse operation.

Fig. 3.8 shows the resulting encrypted and decrypted images with $Q = 71$. A part of decrypted image is zoomed, and we find that there is no visible noise distributed in the images. For other 20 images, we have the same conclusion.

As shown in Tab. 3.1 and Tab. 3.2, if compression ratio is $Q = 71$, the decryption algorithm can reconstruct the plaintext grayscale image with PSNR of 35dB and with UIQI of 0.9217, and reconstruct the plaintext color image with PSNR of 34.4 dB and with UIQI of 0.8752, which are higher than the first and the second ones. That means the decrypted image is the most similar to the plaintext image. If Q=100, the results are the same as in second scheme. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is the smallest one. For $Q = 71$, the runtime of encryption and decryption are 19 ms and 10 ms for grayscale image and 44 ms and 21 ms for color image, it takes a little less time than the encryption after DCT. For $Q = 100$, even though it has the same encrypted result as the second method, the runtime of encryption and decryption are less than the second one, which are 29 ms and 13 ms for grayscale image and 25 ms and 22 ms for color image. So this scheme is the most efficient one.

We do more analyses for 10 grayscale images and 10 color images using this scheme, the average values of PSNR and UIQI are given in Tab. 3.3. If $Q = 100$, the average PSNR and UIQI values are the same as in the second scheme. If $Q = 71$, for 10 grayscale images the average PSNR value is 30.8 dB and the average UIQI value is 0.8598; for 10 color images the average PSNR value is 33 dB and the average UIQI value is 0.8469. They are higher than the PSNR and UIQI values of first and second schemes, which means the decrypted image is the most similar to the plaintext image. The gap between the values of PSNR and UIQI after compression without encryption and the values after decryption is the smallest. So we can conclude that this scheme is the best one to reconstruct the plaintext image.

In this scheme, we can conclude that although the same number of coefficients are encrypted as the second scheme, it takes fewer time. Therefore, compared with the encryption before DCT and encryption after DCT, this scheme is the best one to reconstruct the plaintext image and the most efficient one.

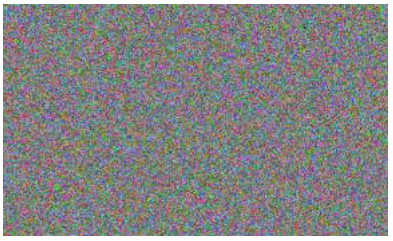
3.2.4 Security of encryption scheme

We use the third scheme to encrypt a black image and a white image, the encrypted results are shown in Fig. 3.9. From the result we can conclude that the attacker cannot distinguish these two images are the encrypted images or juste random images. Therefore in this case, our encryption algorithm is IND-CPA$ secure. If we encrypt two images with the same size and the same chroma (the grayscale image or the colour image), the attacker cannot distinguish the original images from two encrypted images. And in this case, our encryption algorithm is IND-CPA secure. But in any case, we assume that the attacker does not use the metadata to distinguish the images.

3.3 Conclusion

In this chapter, we first propose an encryption algorithm and prove that it is IND-CPA secure. Then in order to preserve the image format (JPEG format), we integrate this algorithm into three different steps of JPEG compression process: before DCT, after DCT and after quantization. In this way, the encrypted results are JPEG images and can be accepted by any image-sharing platforms.

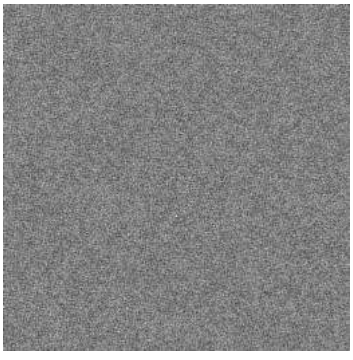
(a) Original image



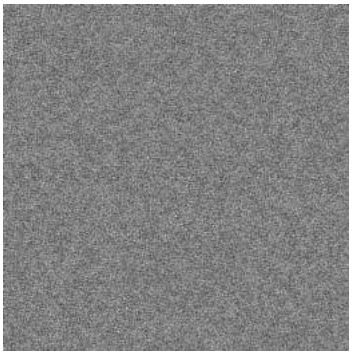
(b) Encrypted image

(c) Decrypted image

Figure 3.8: Experimental results of encryption after quantization.



(a) The encrypted black image



(b) The encrypted white image

Figure 3.9: Security of our encryption scheme.

We introduce how to apply the algorithm in three different steps of JPEG compression process and test the impact of the number of encrypted coefficients on encrypted results. Especially in frequency domain, there is energy compaction property. We encrypt the most appropriate number of coefficient chosen by the test. Compared the results of the three schemes, we consider that the encryption after quantization is the best one to reconstruct the plaintext image and the most efficient one.

In the next chapter, we apply the encryption scheme after quantization on some image-sharing platforms to verify whether it can be used to protect the image published on different image-sharing platforms.

Chapter 4

Experimentations on the image-sharing platforms

Contents

In this chapter, we apply the best scheme which is proposed in the previous chapter on several widely used image-sharing platforms. However, in order to provide sufficient information for our experiments, first of all, we analyze and get to know about the characteristics of several widely used image-sharing platforms. Then we apply the scheme after quantization on different platforms and analyze the experimental results. According to the results, we have found that the some of the existing image-sharing platforms performs post-processing to the published images, and our proposed basic encryption algorithm can be used to protect the privacy on Flickr, Pinterest, Google+ and Twitter.

4.1 Analysis of image-sharing platforms

We aim to use the encryption algorithm to encrypt images and publish the results on some image-sharing platforms to verify whether they can be accepted by different platforms, and then download and decrypt them to analyze the results. But before uploading the encrypted images to the image-sharing platforms, we should firstly know about the characteristics of these platforms. For instance, we need to know what is the maximum image size they can accept, and what is the image compression quality they limit, etc. In this section, we analyze some image-sharing platforms and summarize their characteristics, which provide sufficient information for our following experiments.

We choose the six most widely used image-sharing platforms, i.e. Facebook [1], Flickr [2], Pin-

[1] https://www.facebook.com
[2] https://www.flickr.com

terest [3], Google+ [4], Twitter [5], Instagram [6] and two Chinese image-sharing platforms: Weibo [7] and Wechat [8]. In order to analyze these platforms, we prepare a set of images with different sizes, different compression qualities and different downsampling ratios. One account is created in each platform to upload the prepared images, another account is created to download the images.

All the tests and analyses of platforms – except for Instagram and Wechat – were done on OS X 10.10 with Safari browser. We upload prepared images to these platforms and use another account to download them. Comparing original images and downloaded images, we analyze whether the platforms change the size, the quantization table or the downsampling ratio and what are the values after change.

For Instagram and Wechat, we can only use their applications for smartphone to publish images. The test environment for smartphone is iOS 9.1. The version of Instagram is 7.12.0. The version of Wechat is 6.3.7. We first send the prepared images via a lossless way, i.e. e-mail or "Dropbox", to a smartphone. In this way, images are not processed before being published. Then we upload images to Instagram and Wechat using their native smartphone applications. For Instagram, we login with another account on a laptop to download the images and then compare them with the original ones on the laptop as well. For Wechat, we use another account on smartphone to download the images and send them back to laptop via a lossless way. Then we compare them with the original images on the laptop to analyze which parameters are changed.

Most of image-sharing platforms we test resize the published images with a limitation image size, requantize the published images with a standard quantization table, and for color images, use a standard downsampling ratio. In the following subsections, we detail their limitations and summarize the values in Tab. 4.1.

Platform	Image size	Compression quality	Downsampling ratio
Facebook	2048	71	4:2:0
Flickr	-	-	-
Pinterest	-	-	-
Google+	2048	*	4:2:0
Twitter	600(Width)	75	4:2:0
Instagram	*	*	4:2:0
Weibo	1024	95	-
Wechat	1280	*	4:2:0

Table 4.1: Performances of platforms.

- No change of the original image information
* There is change but no fixed value or the value is special

4.1.1 Image Size

Facebook and Google+ resize the images with a fixed size, and the widest side of image does not exceed 2048 pixels. Weibo limits the widest side to 1024 and Wechat to 1280. Twitter only limits the width of image to 600. Flickr and Pinterest do not limit the image size. Instagram limits the image size but there is no fixed size. However, we find that if we upload an image of 640×640 pixels, the image size is not changed after downloading.

[3] https://www.pinterest.com
[4] https://plus.google.com
[5] https://twitter.com
[6] https://www.instagram.com
[7] http://www.weibo.com
[8] http://www.wechat.com

4.1.2 Quantization Table

It is worth mentioning that, the compression quality value we describe here represents the quantization tables calculated by the standard default tables that used by the IJG (Independent JPEG Group) code library. Facebook requantizes the published images using the quantization tables of $Q = 71$. Flickr and Pinterest do not limit compression quality. Google+ does not limit the compression quality if the image size is smaller than 2048; if the size is larger than 2048, it recompresses the image but there is no fixed quantization table. Twitter and Weibo limit the compression quality only if the quality is larger than the fixed one. Instagram recompresses the images but for each limit size there is a fixed quantization table. For instance, the quantization tables for the image of 640×640 pixels are shown in Fig. 4.1. Wechat has fixed quantization tables, which are shown in Fig. 4.2, but they are not scaled by the standard default tables.

$$\begin{bmatrix} 3 & 3 & 3 & 5 & 8 & 10 & 13 & 16 \\ 3 & 3 & 3 & 5 & 8 & 10 & 13 & 16 \\ 3 & 3 & 5 & 8 & 10 & 13 & 16 & 19 \\ 5 & 5 & 8 & 10 & 13 & 16 & 19 & 22 \\ 8 & 8 & 10 & 13 & 16 & 19 & 22 & 24 \\ 10 & 10 & 13 & 16 & 19 & 22 & 24 & 24 \\ 13 & 13 & 16 & 19 & 22 & 24 & 24 & 24 \\ 16 & 16 & 19 & 22 & 24 & 24 & 24 & 24 \end{bmatrix} \quad \begin{bmatrix} 4 & 4 & 6 & 11 & 26 & 26 & 26 & 26 \\ 4 & 6 & 6 & 17 & 26 & 26 & 26 & 26 \\ 6 & 6 & 14 & 26 & 26 & 26 & 26 & 26 \\ 11 & 17 & 26 & 26 & 26 & 26 & 26 & 26 \\ 26 & 26 & 26 & 26 & 26 & 26 & 26 & 26 \\ 26 & 26 & 26 & 26 & 26 & 26 & 26 & 26 \\ 26 & 26 & 26 & 26 & 26 & 26 & 26 & 26 \\ 26 & 26 & 26 & 26 & 26 & 26 & 26 & 26 \end{bmatrix}$$

(a) Luminance table (b) Chrominance table

Figure 4.1: Fixed quantization tables of Instagram for image of 640×640 pixels.

$$\begin{bmatrix} 9 & 9 & 9 & 16 & 22 & 30 & 38 & 46 \\ 9 & 9 & 9 & 16 & 22 & 30 & 38 & 46 \\ 9 & 9 & 16 & 22 & 30 & 38 & 46 & 55 \\ 16 & 16 & 22 & 30 & 38 & 46 & 55 & 64 \\ 22 & 22 & 30 & 38 & 46 & 55 & 64 & 72 \\ 30 & 30 & 38 & 46 & 55 & 64 & 72 & 72 \\ 38 & 38 & 46 & 55 & 64 & 72 & 72 & 72 \\ 46 & 46 & 55 & 64 & 72 & 72 & 72 & 72 \end{bmatrix} \quad \begin{bmatrix} 11 & 12 & 18 & 31 & 75 & 75 & 75 & 75 \\ 12 & 17 & 17 & 51 & 75 & 75 & 75 & 75 \\ 18 & 17 & 42 & 75 & 75 & 75 & 75 & 75 \\ 31 & 51 & 75 & 75 & 75 & 75 & 75 & 75 \\ 75 & 75 & 75 & 75 & 75 & 75 & 75 & 75 \\ 75 & 75 & 75 & 75 & 75 & 75 & 75 & 75 \\ 75 & 75 & 75 & 75 & 75 & 75 & 75 & 75 \\ 75 & 75 & 75 & 75 & 75 & 75 & 75 & 75 \end{bmatrix}$$

(a) Luminance table (b) Chrominance table

Figure 4.2: Fixed quantization tables of Wechat.

We try to limit the size of an image to the fixed one of each image-sharing platform, and upload this image with $Q = 71$ to these image-sharing platforms, except for Instagram and Wechat. For Instagram, we quantize an image using its quantization tables for image of 640×640 pixels. For Wechat, we limit the size of image and quantize it using its own fixed quantization tables. In any case, we find that the quantization tables are not changed after downloading the image.

4.1.3 Downsampling Ratio

Flickr, Pinterest and Weibo do not limit the downsampling ratio of color images, the others change the downsampling ratio to $4 : 2 : 0$ for color images. If we upload a color image with downsampling ratio $4 : 2 : 0$ to these image-sharing platforms, we find that the ratio is not changed after downloading.

4.2 Experimental results of basic encryption algorithm on image-sharing platforms

According to the analyses above, we have implemented the encryption algorithm after quantization and upload the encrypted images to the six most widely used image-sharing platforms and two Chinese image-sharing platforms. Then we have downloaded and decrypted them. In this section, we detail the experimental processes and present the experimental results. We have used LibJPEG

to encode and decode JPEG image, and all experiments were implemented in C/C++. As examples, we first take the grayscale and the color image "Lena" of 512×512 pixels to do a detailed experiment, since this size is accepted by all platforms. Then we take five grayscale images and five color images of different sizes from 400×600 pixels to 2736×3648 pixels to do more tests.

The encryption is implemented on the laptop before uploading. Before encryption, we resize prepared images to the limitation image size of each platform and change the downsampling ratio for color images to $4:2:0$. Then we requantize the images using quantization tables of $Q = 71$ (shown in Fig. 4.3) for all platforms except for Wechat and Instagram. For Wechat, we requantize the images using its own quantization tables (Fig. 4.2). For Instagram, remember that for each image size there is a standard quantization table. Thus we first resize prepared images to 640×640 pixels and choose the quantization tables for image of 640×640 pixels to requantize the images (Fig. 4.1). We have verified that, in this way, these parameters of prepared images are not changed before and after publishing on each image-sharing platform.

$$\begin{bmatrix} 9 & 6 & 6 & 9 & 14 & 23 & 30 & 35 \\ 7 & 7 & 8 & 11 & 15 & 34 & 35 & 32 \\ 8 & 8 & 9 & 14 & 23 & 33 & 40 & 32 \\ 8 & 10 & 13 & 17 & 30 & 50 & 46 & 36 \\ 10 & 13 & 21 & 32 & 39 & 63 & 60 & 45 \\ 14 & 20 & 32 & 37 & 47 & 60 & 66 & 53 \\ 28 & 37 & 45 & 50 & 60 & 70 & 70 & 59 \\ 42 & 53 & 55 & 57 & 65 & 58 & 60 & 57 \end{bmatrix} \qquad \begin{bmatrix} 10 & 10 & 14 & 27 & 57 & 57 & 57 & 57 \\ 10 & 12 & 15 & 38 & 57 & 57 & 57 & 57 \\ 14 & 15 & 32 & 57 & 57 & 57 & 57 & 57 \\ 27 & 38 & 57 & 57 & 57 & 57 & 57 & 57 \\ 57 & 57 & 57 & 57 & 57 & 57 & 57 & 57 \\ 57 & 57 & 57 & 57 & 57 & 57 & 57 & 57 \\ 57 & 57 & 57 & 57 & 57 & 57 & 57 & 57 \\ 57 & 57 & 57 & 57 & 57 & 57 & 57 & 57 \end{bmatrix}$$

(a) Luminance table (b) Chrominance table

Figure 4.3: Quantization tables of $Q = 71$.

All the prepared images are encrypted by using the algorithm described in section 3.2.3. Next, we upload and download the encrypted images, as described in Sec. 4.1. Then the decryption is implemented on the laptop and we obtain the decrypted images. For comparing original images and decrypted images, we calculate their PSNR and UIQI.

According to the experiments, all of the encrypted images are completely unreadable, and when we upload the encrypted images, all of them are accepted by the image-sharing platforms as correct image format. But when we download and decrypt the images, we find that there are two types of results: positive results and negative results. As positive results, the recovered images have pretty high quality. As negative results, the quality of recovered images is extremely poor. In the next subsections, we present these two types of results.

4.2.1 Positive Results

If we publish the encrypted images on Flickr, Pinterest Google+ and Twitter, the recovered images have high quality. We summarize the results of "Lena" in Tab. 4.2, and the average results of other images in Tab. 4.3. We also present in these tables the PSNR and UIQI values of decryption without publishing the encrypted images. Then Fig. 4.4 shows the decrypted results of "Lena" after downloading from different image-sharing platforms. We find that the decryption algorithm can reconstruct the downloaded images with a high quality and there is no visible noise in images. Comparing with the decrypted images without publishing, they are very similar. For the other ten images, we have the same conclusion.

As shown in Tab. 4.2 and Tab. 4.3, on these four image-sharing platforms, the PSNR values are around 31∼35 dB, and the UIQI values are around 0.7∼0.9, which means the recovered images are similar to the original ones, although there are few losses. If we directly decrypt the encrypted images without publishing them, the PSNR values are around 31∼35 dB, and the UIQI values are around 0.7∼0.9. Comparing them with the values of PSNR and UIQI after publishing, the gap is very narrow and almost to vanishing point.

4.2.2 Negative Results

If we publish the encrypted images on Facebook, Instagram, Weibo and Wechat, the recovered images have very poor quality. We summarize the results of "Lena" in Tab. 4.4, and Fig. 4.5 shows

Platform	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Flickr	34.92	0.922	34.42	0.875
Pinterest	34.92	0.922	34.42	0.875
Google+	34.92	0.922	32.47	0.870
Twitter	34.92	0.922	34.26	0.877
Decryption without publishing	34.92	0.922	34.42	0.875

Table 4.2: Summary of the positive results of "Lena".

Platform	Grayscale image		Color image	
	Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Flickr	31.34	0.795	33.66	0.780
Pinterest	31.34	0.795	33.66	0.780
Google+	30.90	0.853	32.69	0.851
Twitter	30.43	0.871	31.05	0.870
Decryption without publishing	31.34	0.795	33.66	0.780

Table 4.3: Summary of the positive results of 10 other images.

(a) Flickr: grayscale image. (b) Flickr: color image. (c) Pinterest: grayscale image. (d) Pinterest: color image.

(e) Google+: grayscale image. (f) Google+: color image. (g) Twitter: grayscale image. (h) Twitter: color image.

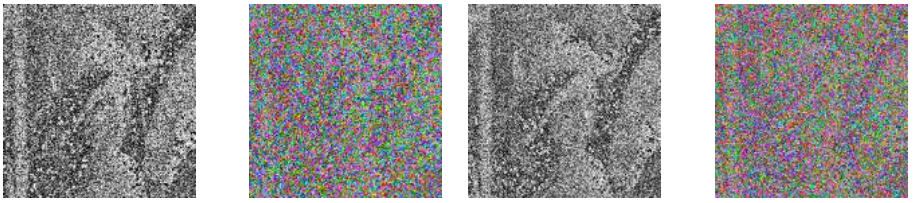
Figure 4.4: Positive experimental results of decrypted images after publishing on different platforms.

the decrypted results of "Lena" after downloading from the image-sharing platforms. We find that the decryption algorithm cannot reconstruct the downloaded images. For the other ten images, we have the same conclusion that the recovered images are still obscure and we only can recognize the outline.

As shown in Tab. 4.4, on these four image-sharing platforms, the values of PSNR and UIQI are smaller than 10 dB and 0.1, which means the quality of recovered images is extremely poor.

Platform	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Facebook	8.39	0.028	7.05	0.018
Instagram	8.31	0.022	7.17	0.014
Weibo	8.37	0.028	7.00	0.019
Wechat	8.35	0.026	7.20	0.016

Table 4.4: Summary of the negative results of "Lena".



(a) Facebook: grayscale image. (b) Facebook: color image. (c) Instagram: grayscale image. (d) Instagram: color image.

(e) Weibo: grayscale image. (f) Weibo: color image. (g) Wechat: grayscale image. (h) Wechat: color image.

Figure 4.5: Negative experimental results of decrypted images after publishing on different platforms.

4.3 Analysis of negative results

In this section, in order to understand why the images cannot be reconstructed after being published on some image-sharing platforms, we analyze the experimental results in section 4.2 and simulate the processes after uploading.

4.3.1 Analysis of the downloaded images

According to the experimental results in section 4.2, we find that the images cannot be reconstructed with a high quality after being published on Facebook, Instagram, Weibo and Wechat. However, verifying image size, quantization table and downsampling ratio of image, they are not changed before uploading and after downloading. So we want to know what changes happen to the image before uploading and after downloading.

First we take Facebook for example, and analyze the first 8×8 block of grayscale "Lena".

For a JPEG compression of $Q = 71$, the original first block before dequantization is:

$$\begin{bmatrix} 28 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.1}$$

After encryption, the block becomes:

$$\begin{bmatrix} 71 & 75 & 18 & 43 & 5 & 0 & 0 & 0 \\ 21 & -27 & 84 & 1 & 0 & 0 & 0 & 0 \\ -30 & 105 & -78 & 0 & 0 & 0 & 0 & 0 \\ 60 & -71 & 0 & 0 & 0 & 0 & 0 & 0 \\ -40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.2}$$

Then we upload grayscale "Lena" to Facebook and download it. The first 8×8 block becomes:

$$\begin{bmatrix} 46 & 32 & 21 & 19 & 1 & -2 & -1 & 0 \\ -4 & -15 & 24 & 3 & -6 & 0 & 0 & -1 \\ 8 & 42 & -36 & -6 & 1 & -1 & 0 & 1 \\ 24 & -16 & -8 & 6 & 1 & 0 & 1 & -1 \\ -12 & -5 & 4 & 2 & 1 & 0 & -1 & 0 \\ -2 & 2 & 0 & 1 & -1 & 0 & 0 & -1 \\ 1 & 0 & 1 & -1 & 1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \end{bmatrix} \tag{4.3}$$

Compared with the block (4.2), the coefficients are totally changed. After decryption, the block becomes:

$$\begin{bmatrix} 3 & -42 & 4 & -24 & -4 & 0 & 0 & 0 \\ -24 & 12 & -60 & 2 & 0 & 0 & 0 & 0 \\ 37 & -63 & 42 & 0 & 0 & 0 & 0 & 0 \\ -36 & 55 & 0 & 0 & 0 & 0 & 0 & 0 \\ 28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.4}$$

It is different from the original block (4.1), thus the image cannot be reconstructed correctly. We do the same test on Instagram, Weibo and Wechat, and we have the same results: the blocks before uploading and after downloading are different. Therefore, image-sharing platforms like Facebook process the images during publishing in a way we do not take into account.

Then we take Flick for example. After downloading the encrypted grayscale "Lena", the first 8×8 block is:

$$\begin{bmatrix} 71 & 75 & 18 & 43 & 5 & 0 & 0 & 0 \\ 21 & -27 & 84 & 1 & 0 & 0 & 0 & 0 \\ -30 & 105 & -78 & 0 & 0 & 0 & 0 & 0 \\ 60 & -71 & 0 & 0 & 0 & 0 & 0 & 0 \\ -40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.5}$$

Compared with the block (4.2), the coefficients have no changes. After decryption, the block becomes:

$$\begin{bmatrix} 28 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

It is the same as the original block (4.1), thus the reconstructed image has a high quality. We can conclude that if we set appropriate parameters for images to be uploaded, image-sharing platforms like Flickr do not process the image during publishing.

4.3.2 Upload Simulation

According to the examples above, we find that for the image-sharing platforms like Facebook, the coefficients of image are different before uploading and after downloading, but the image-sharing platforms like Flickr do not have this problem. Therefore, we try to simulate the process that image-sharing platforms like Facebook possibly do during publishing. We still take the first 8×8 block of grayscale "Lena" for example.

We suppose when we upload an image to image-sharing platforms like Facebook, even if all the parameters of image are the fixed ones which are mentioned in section 4.1, these image-sharing platforms carry out the complete processes of JPEG compression and decompression. Therefore, while the encrypted image is uploaded, the dequantization and reverse DCT are firstly applied, and the first block of "Lena" becomes:

$$\begin{bmatrix} 253 & 139 & 44 & 51 & 99 & 95 & 34 & -22 \\ 423 & 260 & 92 & 34 & 52 & 58 & 30 & 1 \\ 454 & 246 & 13 & -90 & -70 & -13 & 29 & 50 \\ 266 & 66 & -145 & -205 & -119 & 18 & 134 & 201 \\ 114 & -24 & -134 & -95 & 60 & 225 & 342 & 401 \\ 127 & 67 & 61 & 164 & 296 & 357 & 340 & 307 \\ 160 & 152 & 198 & 286 & 295 & 145 & -96 & -273 \\ 132 & 144 & 199 & 242 & 133 & -192 & -608 & -894 \end{bmatrix} \tag{4.7}$$

Remember that, after reverse DCT, the range of coefficients is $[-128, 127]$. But we find that in block (4.7), there are many values exceed this interval, they should be set to -128 or 127. Then this block becomes:

$$\begin{bmatrix} 127 & 127 & 44 & 51 & 99 & 95 & 34 & -22 \\ 127 & 127 & 92 & 34 & 52 & 58 & 30 & 1 \\ 127 & 127 & 13 & -90 & -70 & -13 & 29 & 50 \\ 127 & 66 & -128 & -128 & -119 & 18 & 127 & 127 \\ 114 & -24 & -128 & -95 & 60 & 127 & 127 & 127 \\ 127 & 67 & 61 & 127 & 127 & 127 & 127 & 127 \\ 127 & 127 & 127 & 127 & 127 & 127 & -96 & -128 \\ 127 & 127 & 127 & 127 & 127 & -128 & -128 & -128 \end{bmatrix} \tag{4.8}$$

Now we apply DCT and quantization with $Q = 71$, the block becomes:

$$\begin{bmatrix} 46 & 32 & 21 & 19 & 1 & -2 & -1 & 0 \\ -4 & -15 & 24 & 3 & -6 & 0 & 0 & -1 \\ 8 & 43 & -36 & -6 & 1 & -1 & 0 & 1 \\ 24 & -16 & -8 & 6 & 1 & 0 & 1 & -1 \\ -12 & -5 & 4 & 2 & 1 & 0 & -1 & 0 \\ -2 & 2 & 1 & 1 & -1 & 0 & 0 & -1 \\ 1 & 0 & 1 & -1 & 1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \end{bmatrix} \tag{4.9}$$

We can clearly see that this block is almost the same as (4.3) which is the first 8×8 block of the downloaded image from Facebook, except for certain values. That is because of the roundoff error when the DCT, reverse DCT and quantization are applied.

Due to the losses after reverse DCT, the block after downloading is different from the original block and the recovered image has a poor quality. Besides the resetting of coefficients after reverse DCT and the roundoff error of quantization, when color image is converted from YCbCr into RGB color space, results of conversion need to be rounded to the nearest integer value, so there is roundoff error as well. The range of RGB pixels is $[0, 255]$, so just like the resetting after reverse DCT, any values exceed this interval should be set to 0 or 255. In this way, while the color space is converted back, we cannot obtain the original values.

4.4 Conclusion

In this chapter, we first analyze eight most widely used image-sharing platforms, in order to provide their characteristics for our experiments. Next, we prepare some images according to the characteristics of the image-sharing platforms, and implement the scheme after quantization to encrypt the prepared images. We upload the encrypted images to the eight image-sharing platforms which are analyzed before, and then download and decrypt the images. According to the experiments, there are two types of results: positive results and negative results. On Flickr, Pinterest, Google+ and Twitter, the decryption algorithm can reconstruct the downloaded images with a high quality and also similar to the decrypted images without publishing. On Facebook, Instagram, Weibo and Wechat, the quality of recovered images is extremely poor. The values of PSNR and UIQI are smaller than 10dB and 0.1.

Through some analyses, we find that it is because image published on these image-sharing platforms are recompressed by using JPEG standard, and during decoding, once the coefficients exceed the setting ranges, they will be reset. Therefore, we cannot obtain the original values while the image is encoded again.

In the next chapter, to overcome the restriction, we improve the encryption algorithm and implement the improved algorithm on some of the image-sharing platforms.

Chapter 5

The improved encryption algorithm for published images

Contents

In this chapter, we improve the basic encryption algorithm to reduce or even avoid the losses of quality during the recompression and to decrypt the image with a high quality after publishing it on the image-sharing platforms. Then we details the experiments of this improved encryption algorithm on Facebook, Weibo and Wechat. In the end, we evaluate the performances of the improved encryption algorithm which is developed in C++.

5.1 Improved encryption algorithm for published images

After learning the limitation of the previous algorithm, we try to overcome the restriction. Thus, we improve the scheme of encryption after quantization to reduce or even avoid the losses of image information during JPEG recompression. We still use a pseudo-random function *prf* which takes value n as input and returns a pseudo-random value in the range $[0, n[$. This function is associated

with an initialization function prf_{init} that takes values *key* and *seed* as inputs, and allows to initialize the function *prf*. We add correcting code during encryption to tolerate the recompression, and after downloading from platforms, we correct the recovered coefficients.

There are six functions in the scheme:

Key Generation: $key = KeyGen(\lambda)$ takes λ as input and returns a random value *key* of λ bits.

Encryption: $CoefC = Enc(I, key, seed)$ uses *key* and a random value *seed* to initialize the pseudo-random function and generates a pseudo-random sequence. Then encrypts an image I using this sequence. The result is encrypted coefficients $CoefC$.

Correcting code: $C = Ccode(CoefC)$ takes encrypted coefficients $CoefC$ as input and generates the correcting code to embed in each block, then returns the encrypted image C as result.

Inverse correcting code: $CoefC' = ICcode(C')$ takes downloaded encrypted image C' as input and reconstructs the original encrypted coefficients $CoefC'$.

Decryption: $CoefI = Dec(CoefC', key, seed)$ decrypts the reconstructed coefficients $CoefC'$ using the pseudo-random sequence which is initialized by *key* and the random value *seed*.

Correction: $I' = Corr(CoefI)$ takes decrypted coefficient $CoefI$ as input and after fine-tuning, a high quality image I' can be obtained as the result.

In the following subsections, we detail these functions.

5.1.1 Encryption

We use the scheme of encryption after quantization. First of all, the function $prf_{\text{init}}(key, seed)$ initializes the pseudo-random function *prf* using random values *key* and *seed*.

According to the test of encrypting different coefficients after quantization in Sec. 3.2.3, we know that after encrypting DC and 9 AC coefficients, the recovered image is similar to the original image with enough details. Although the scheme choose to encrypt DC and the first 14 AC coefficients in the end, considering in the improved algorithm, we want to add the correcting code in each 8×8 block, so we prefer to encrypt fewer coefficients to provide sufficient space for the further correcting code.

Therefore, in Y component or grayscale image we only encrypt the DC coefficient and the first 9 AC coefficients according to "zigzag" order and set the remaining coefficients to 0 as described in matrix (5.1). We have already proved that in this way, the security, efficiency of algorithm and the quality of recovered image are satisfactory. Remember that, for a color image, the sensitivity of human eye to luminance (the Y component) intensity changes is greater than to chrominance (the Cb and Cr) intensity changes. Therefore, in Cb or Cr component we only encrypt the DC coefficient, but in order to keep some details, we retain the first 3 AC coefficients and move them to place $3, 5, 4$ (as shown in Fig 5.1(b)), and set the remaining coefficients to 0. If we set the DC coefficient to any constant, the first 3 AC coefficients cannot reveal enough information to allow the attackers to identify the original images, so retaining the first 3 AC coefficients is secure.

$$\begin{bmatrix} E_{00} & E_{01} & E_{02} & E_{03} & 0 & 0 & 0 & 0 \\ E_{10} & E_{11} & E_{12} & 0 & 0 & 0 & 0 & 0 \\ E_{20} & E_{21} & 0 & 0 & 0 & 0 & 0 & 0 \\ E_{30} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

5.1.2 Correcting code

We simulate the process that some image-sharing platforms possibly do during publishing an image, while the encrypted image is uploaded, the dequantization and reverse DCT are firstly applied. After reverse DCT, the range of coefficients should be in $[-128, 127]$, and all the values exceed this interval will be set to -128 or 127. So we should try to ensure that the range of uploaded coefficients is as close as possible to $[-128, 127]$ after reverse DCT. According to our analyses, we already know that the used quantization tables of different platforms, and depending on the elements of quantization tables, we know the range of encrypted coefficients, therefore, we can calculate the range of coefficients after dequantization and reverse DCT.

If we directly upload the encrypted coefficient, after dequantization and reverse DCT, some coefficients may be out of the range $[-128, 127]$. Therefore, we consider adding a correcting code, in order to deal with the encrypted values, and control the ranges. The main idea is to minimise the encrypted coefficient in the block.

One way to minimise the encrypted coefficient is to divide it by a constant d. The quotient is rounded to the nearest integer, denoted by $Rq_{i,j} = round(\frac{E_{i,j}}{d})$. This rounded quotient is multiplied by the constant divisor d and we calculate the difference between the multiplication result and the original coefficient, denoted by $Diff_{i,j} = E_{i,j} - Rq_{i,j} \times d$. We use $Rq_{i,j}$ and $Diff_{i,j}$ to replace the original encrypted coefficients in each block. Once we choose a suitable divisor, and reasonable allocation of storage location, the range of coefficients after reverse DCT can be controlled.

For Y component or grayscale image, the encrypted coefficient $E_{i,j}$ is processed block by block as follow to generate the correcting code and the process is illustrated in Fig 5.1(a):

1. The encrypted coefficients in each 8×8 block are divided by a divisor d_1 and rounded to the nearest integer $Rq_{i,j}$.

2. The coefficients $Rq_{i,j}$ in each block are multiplied by d_1, and then we calculate the difference $Diff_{i,j}$ between the multiplication result and the original encrypted block. The range of $Diff_{i,j}$ falls in $[-\frac{d_1}{2}, \frac{d_1}{2})$, and according to this range, we can calculate the corresponding maximum or minimum values of $Rq_{i,j}$.

For Cb or Cr component, we note the divisor as d_2. The encrypted DC coefficient $E_{0,0}$ is processed block by block as follow to generate the correcting code and the process is illustrated in Fig 5.1(b):

1. The encrypted DC coefficients in each 8×8 block is divided by d_2 and rounded to the nearest integer $Rq_{0,0}$. We store it in place 1 according to zigzag order in each block.

2. The DC coefficient in this new block is multiplied by d_2, and then we calculate the difference $Diff_{0,0}$ between the multiplication result and the original encrypted block. We store it in place 2 according to zigzag order in each block.

   Finally, by performing the rest of encoding processes, the encrypted image C is obtained.

5.1.3 Inverse correcting code

After downloading, the downloaded encrypted image C' is in format JPEG, first we decode it just before the dequantization and it is represented by several non-overlapped 8×8 blocks. Each block is processed to recover the original encrypted coefficients as follow:

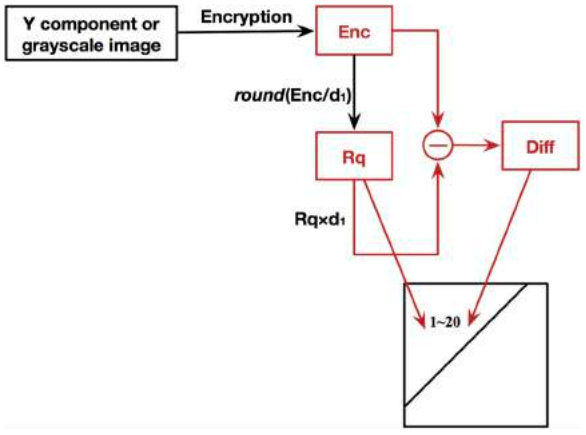
For Y component or grayscale image,

1. Extract the coefficients $Rq'_{i,j}$ in the block, multiply by d_1.

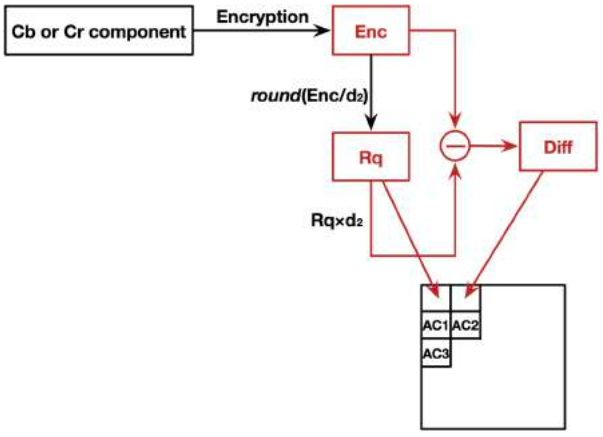
2. Extract the coefficients $Diff'_{i,j}$, and sequentially add them to the corresponding $Rq'_{i,j} \times d_1$. The results are stored in place 1 to 10 according to zigzag order.

For Cb or Cr component,

1. Extract the first coefficient $Rq'_{0,0}$ in the block, multiply by d_2.



(a) Correcting code in Y component or grayscale image



(b) Correcting code in Cb or Cr component

Figure 5.1: The process of generating the correcting code

2. Extract the second coefficient $Diff'_{0,0}$, and add it to $Rq'_{0,0} \times d_2$. The results are stored in place 1 according to zigzag order.

These coefficients are approximate to the original encrypted ones, and we can take them as the input of decryption algorithm.

5.1.4 Decryption

First of all, the function $prf_{\text{init}}(key, seed)$ initializes the pseudo-random function prf using random values key and $seed$.

We decrypt the result of previous step block by block. In Y component or grayscale image we only decrypt the DC coefficient and the first 9 AC coefficients according to "zigzag" order. In Cb or Cr component we only decrypt the DC coefficient and move back the first 3 AC coefficients from place $3, 5, 4$.

1. Suppose that the recovered encrypted coefficients $E'_{i,j}$ falls in the range $[Qmin_{i,j}, Qmax_{i,j}]$ as we explain above. We note $n_{i,j} = Qmax_{i,j} - Qmin_{i,j} + 1$.

2. Each recovered encrypted coefficient $E'_{i,j}$ is decrypted block by block as follow:

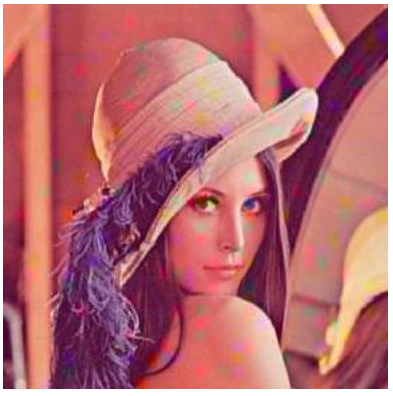
$$\begin{aligned} D_{i,j} =&(((E_{i,j} - Qmin_{i,j}) + (n_{i,j} - prf(n_{i,j}))) \\ &\mod n_{i,j}) + Qmin_{i,j} \end{aligned} \tag{5.2}$$

Finally, the decrypted block is entropy encoded and the decrypted coefficients $CoefI$ before fine-tuning are obtained.

5.1.5 Correction

Observing the decrypted image we obtain now, there are still some error pixels (Shown in Fig. 5.2). In order to make the result accurate, we propose to correct these errors. However, the correction is not a general one, it only applies to the errors which are generated by our decryption algorithm.

(a) An example of grayscale image



(b) An example of color image

Figure 5.2: The decrypted image without correction

First, we find that some blocks of the decrypted image are wrong, they have opposite color with the right one (Fig. 5.2(a)). However, the neighbours of the wrong block have the right color. Therefore, we try to use the information of the right neighbours to correct the wrong block. We found that the difference between the wrong block and its right neighbours is the sign of DC coefficient. After some test, we found that once we invert the sign of the DC coefficient of the wrong block, its color can be corrected. Therefore, in our algorithm, we get the DC coefficient in each block, and compare the sign of one DC coefficient and the signs of its neighbour DC coefficients. If its sign is inverse, we forcibly invert it.

For Cb or Cr component, we found that the color of some blocks have deviations (Fig. 5.2(b)), therefore an additional correction has to be implemented. Observing the deviated block, the deviation comes from the DC coefficient of the wrong block, and the difference DC value between the wrong block and its neighbours is usually ± 10. In order to detect which one is the wrong block, we compare the DC coefficient of its upper and lower neighbour and left and right neighbour, if the difference exceeds a threshold, we think that this block has deviation. After some tests, we set the threshold to 5 which can better find the wrong block. But we also find that sometimes the difference comes from the gap of color in the image, therefore, we need to ensure that both of its upper and lower neighbours or left and right neighbours satisfy this threshold. With the order of upper and lower or left and right, the results have some differences, therefore, for more accurate, we compare them in different order. In the first round, we compare upper and lower first and then left and right, in the second round, we compare left and right first and then upper and lower, then we calculate the average of these two rounds (Shown in Fig. 5.3). In the following, we detail the implementation process.

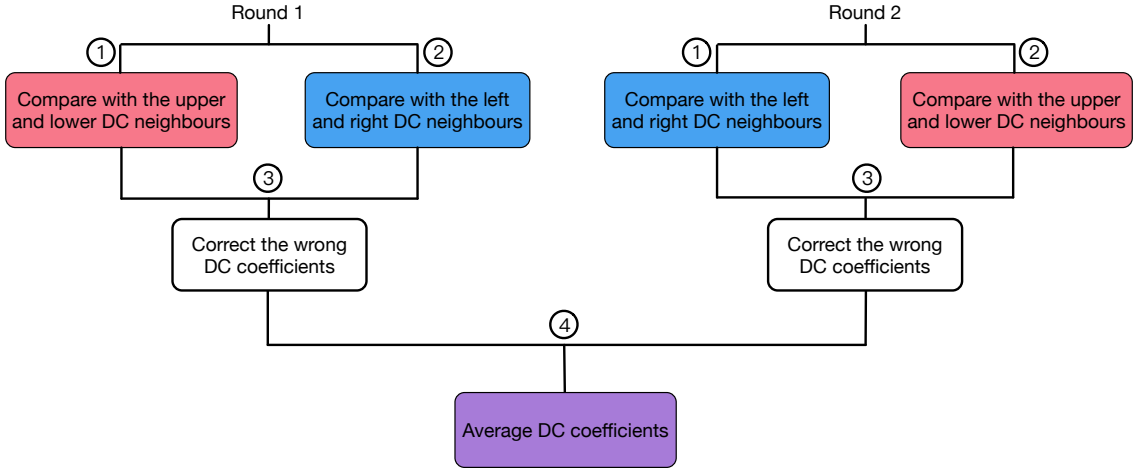


Figure 5.3: The correction process of Cb or Cr component.

We get the DC coefficient from each block, and put these DC coefficients together to form a new matrix. Take one of DC coefficients except the boundary ones, and define it as $dc(i,j)$.

1. We compare $dc(i,j)$ with its upper and lower DC coefficients $dc(i-1,j)$ and $dc(i+1,j)$. If the two differences are all more than 5, i.e. $dc(i,j)-dc(i-1,j)>5$ && $dc(i,j)-dc(i+1,j)>5$, then $dc(i,j)$ minus 10. If the two differences are all less than 5, i.e. $dc(i,j)-dc(i-1,j)<5$ && $dc(i,j)-dc(i+1,j)<5$, then $dc(i,j)$ plus 10.

2. $dc'(i,j)$ denotes one of DC coefficients except the boundary ones in the result of step 1, we compare it with its left and right DC coefficients $dc'(i,j-1)$ and $dc'(i,j+1)$. If the two differences are all more than 5, i.e. $dc'(i,j)-dc'(i,j-1)>5$ && $dc'(i,j)-dc'(i,j+1)>5$, then $dc'(i,j)$ minus 10. If the two differences are all less than 5, i.e. $dc'(i,j)-dc'(i,j-1)<5$ && $dc'(i,j)-dc'(i,j+1)<5$, then $dc'(i,j)$ plus 10.

3. The same operations are implemented once more for original $dc(i,j)$. But this time, we first compare it with its left and right DC coefficients, and then compare it with its upper and lower DC coefficients.

4. Average the results of step 2 and step 3 as a new DC matrix. Then the 2-D median filter using a window size of three is applied to this matrix. For the boundary coefficients, we replace them using the average value of their neighbours.

Finally, by performing the rest of encoding processes, the decrypted image I' is obtained.

5.1.6 Security of the algorithm

In chapter 3, we have already discussed about the security of the encryption scheme, that the proposed scheme is IND-CPA$ secure, and considering the two images have same size and same chroma, the proposed scheme can be IND-CPA secure.

In the improved algorithm, for the grayscale image, the scheme is still IND-CPA$ secure, and so as to the Y component of color image. For the Cb and Cr components, at first, we consider to encrypt DC and the first three AC coefficients, however, because of the roundoff error during the recompression on the image-sharing platforms, the chroma components cannot be recovered even with our correction mechanism (Shown in Fig. 5.4). Then we tried to only encrypt the DC coefficient, the chroma components can be recovered correctly but we lose all the details. Therefore, in order to ensure the quality of the decrypted image after publishing, we have to make a compromise and give up the security of the algorithm. For the Cb and Cr components, the first three AC coefficients are kept unencrypted, and we can recover the image without losing most of details. But in this way, the attacker can obtain some unencrypted coefficients, which help him to recognize the original image from the encrypted image. So the improved algorithm for color image is no longer IND-CPA$ secure.

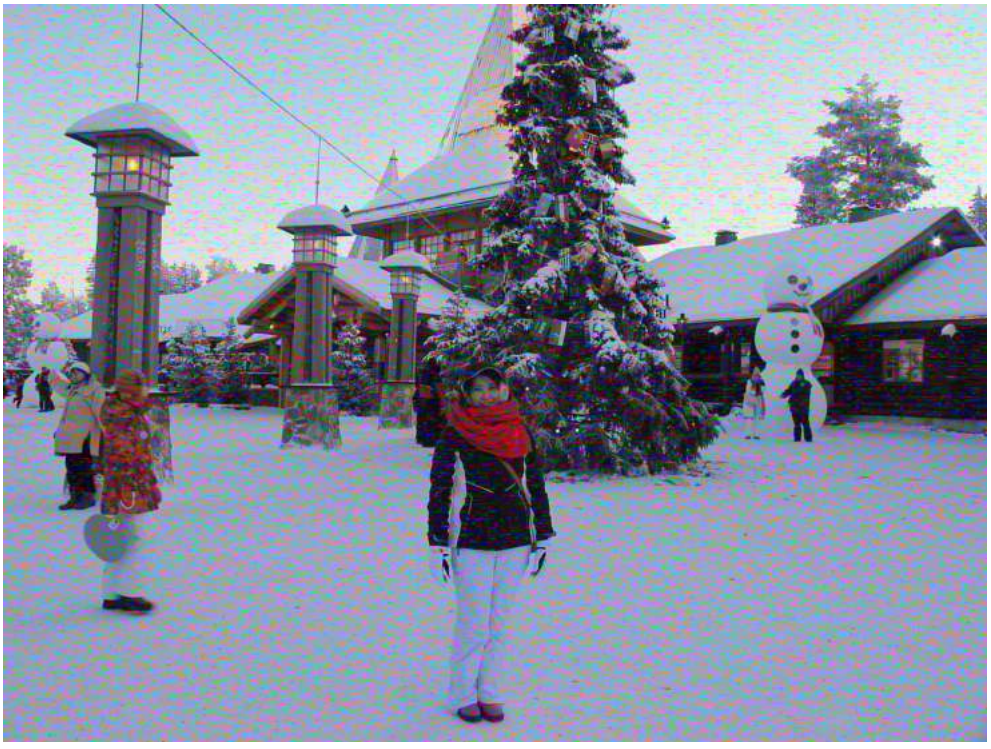


Figure 5.4: The correction process of Cb or Cr component.

5.2 The parameters selection

In the improved encryption algorithm, when we add the correcting code for each platforms, we need to find two suitable divisors for Y and UV components respectively and reasonable allocation of storage location. In the following subsections, we do some tests to determine these parameters.

5.2.1 Facebook

We have already known that platforms like Facebook carry out the complete processes of JPEG compression and decompression when we upload an image. That means, while the encrypted image is uploaded, the dequantization and reverse DCT are firstly applied. After reverse DCT, we need

to ensure that the range of coefficients falls in $[-128, 127]$, or at least close to this range. Therefore, we need to study the coefficients after reverse DCT when we choose the divisors.

We found that Facebook requantizes the published images using the quantization tables $Q = 71$ as shown in Fig. 4.3. It is worth mentioning that, the compression quality value we describe here represents the quantization tables calculated by the standard default tables that used by the IJG (Independent JPEG Group) code library.

The ranges of DCT coefficients in one block are shown in Fig. 3.4, then after being quantized by $Q = 71$, the ranges of coefficients are shown in Fig. 5.5.

$$\begin{bmatrix} -114 & -154 & -157 & -103 & -73 & -40 & -31 & -26 \\ -132 & -120 & -107 & -76 & -62 & -25 & -24 & -26 \\ -118 & -107 & -97 & -61 & -41 & -26 & -22 & -27 \\ -116 & -84 & -66 & -49 & -31 & -17 & -19 & -23 \\ -102 & -71 & -45 & -29 & -26 & -15 & -16 & -21 \\ -66 & -42 & -27 & -23 & -20 & -14 & -13 & -16 \\ -34 & -23 & -19 & -17 & -16 & -12 & -12 & -14 \\ -22 & -16 & -16 & -15 & -14 & -14 & -14 & -15 \end{bmatrix}$$

(a) The minimum values of luminance table

$$\begin{bmatrix} 113 & 154 & 157 & 103 & 73 & 40 & 31 & 26 \\ 132 & 120 & 107 & 76 & 62 & 25 & 24 & 26 \\ 118 & 107 & 97 & 61 & 41 & 26 & 22 & 27 \\ 116 & 84 & 66 & 49 & 31 & 17 & 19 & 23 \\ 102 & 71 & 45 & 29 & 26 & 15 & 16 & 21 \\ 66 & 42 & 27 & 23 & 20 & 14 & 13 & 16 \\ 34 & 23 & 19 & 17 & 16 & 12 & 12 & 14 \\ 22 & 16 & 16 & 15 & 14 & 14 & 14 & 15 \end{bmatrix}$$

(b) The maximum values of luminance table

$$\begin{bmatrix} -102 & -93 & -67 & -34 & -18 & -16 & -17 & -16 \\ -93 & -70 & -57 & -22 & -16 & -15 & -15 & -15 \\ -67 & -57 & -27 & -15 & -17 & -15 & -15 & -15 \\ -34 & -22 & -15 & -15 & -16 & -15 & -15 & -15 \\ -18 & -16 & -17 & -16 & -18 & -16 & -17 & -16 \\ -16 & -15 & -15 & -15 & -16 & -15 & -15 & -15 \\ -17 & -15 & -15 & -15 & -17 & -15 & -15 & -15 \\ -16 & -15 & -15 & -15 & -16 & -15 & -15 & -15 \end{bmatrix}$$

(c) The minimum values of chrominance table

$$\begin{bmatrix} 102 & 93 & 67 & 34 & 18 & 16 & 17 & 16 \\ 93 & 70 & 57 & 22 & 16 & 15 & 15 & 15 \\ 67 & 57 & 27 & 15 & 17 & 15 & 15 & 15 \\ 34 & 22 & 15 & 15 & 16 & 15 & 15 & 15 \\ 18 & 16 & 17 & 16 & 18 & 16 & 17 & 16 \\ 16 & 15 & 15 & 15 & 16 & 15 & 15 & 15 \\ 17 & 15 & 15 & 15 & 17 & 15 & 15 & 15 \\ 16 & 15 & 15 & 15 & 16 & 15 & 15 & 15 \end{bmatrix}$$

(d) The maximum values of chrominance table

Figure 5.5: The range of quantized DCT coefficients in one block (quantized by $Q = 71$).

For Y component or grayscale image, we test d_1 from 9 to 21, and the range of $Diff_{i,j}$ falls in $[-\frac{d_1}{2}, \frac{d_1}{2})$. Suppose that each $Diff_{i,j}$ can reach its maximum or minimum value, and then we can calculate the corresponding maximum or minimum values of $Rq_{i,j}$. According to the element of quantization table $q(i, j)$, we place the extremum values of $Rq_{i,j}$ and $Diff_{i,j}$ in each block. The principle is to store the larger value of $Rq_{i,j}$ and $Diff_{i,j}$ on the position where the smaller $q(i, j)$ is placed. For example, if $d_1 = 9$, the range of $Diff_{i,j}$ is $[-4, 4]$. In place 1, the maximum value of quantized DCT coefficients is 113. When $Diff_{0,0} = 4$, the maximum value that $Rq_{0,0}$ can reach is $floor[(113 - 4)/9] = 12$. In this way, we can get the first 10 maximum values of $Rq_{i,j}$:

$$\begin{bmatrix} 12 & 16 & 17 & 11 & 0 & 0 & 0 & 0 \\ 14 & 12 & 11 & 0 & 0 & 0 & 0 & 0 \\ 12 & 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

According to the quantization table in Fig. 4.3(a), we store $Rq_{i,j}$ and $Diff_{i,j}$:

$$\begin{bmatrix} 11 & 17 & 16 & 11 & 4 & 0 & 0 & 0 \\ 14 & 12 & 12 & 4 & 4 & 0 & 0 & 0 \\ 12 & 12 & 4 & 4 & 0 & 0 & 0 & 0 \\ 11 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.4}$$

Then we dequantize this block:

$$\begin{bmatrix} 99 & 102 & 96 & 99 & 56 & 0 & 0 & 0 \\ 98 & 84 & 96 & 44 & 60 & 0 & 0 & 0 \\ 96 & 96 & 36 & 56 & 0 & 0 & 0 & 0 \\ 88 & 40 & 52 & 0 & 0 & 0 & 0 & 0 \\ 40 & 52 & 0 & 0 & 0 & 0 & 0 & 0 \\ 56 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

and do reverse DCT to see whether the values exceed the range $[-128, 127]$:

$$\begin{bmatrix} 252 & 182 & 193 & 179 & 179 & 193 & 182 & 252 \\ 188 & 136 & 146 & 135 & 135 & 146 & 136 & 188 \\ 187 & 135 & 142 & 134 & 134 & 142 & 135 & 187 \\ 182 & 132 & 141 & 126 & 126 & 141 & 132 & 182 \\ 182 & 132 & 141 & 126 & 126 & 141 & 132 & 182 \\ 187 & 135 & 142 & 134 & 134 & 142 & 135 & 187 \\ 188 & 135 & 146 & 135 & 135 & 146 & 136 & 188 \\ 252 & 182 & 193 & 179 & 179 & 193 & 182 & 252 \end{bmatrix} \tag{5.6}$$

The maximum value is 252, and the minimum values can be calculated by using the same process. For other chose divisor, we do the same test, and calculate the extremum ranges, which are listed in Tab 5.2.

Divisor	Range
9	$[-252, 252]$
10	$[-252, 244]$
11	$[-238, 238]$
12	$[-248, 239]$
13	$[-235, 235]$
14	$[-246, 239]$
15	$[-237, 237]$
16	$[-241, 229]$
17	$[-232, 232]$
18	$[-240, 235]$
19	$[-230, 230]$
20	$[-239, 234]$
21	$[-234, 234]$

Table 5.1: The extremum values of coefficients after reverse DCT for different divisors (Facebook).

We find that when $d_1 = 19$, the ranges are closest to $[-128, 127]$ after reverse DCT. Even tough they exceed, considering not every coefficients can get the extremum values at the same time, the probability of falling into the interval $[-128, 127]$ is great.

Therefore, for Facebook we choose 19 as the divisor d_1 for Y component or grayscale image. The maximum number of $Diff_{i,j}$ is 9, and the maximum values of $Rq_{i,j}$ are:

$$\begin{bmatrix} 5 & 7 & 7 & 4 & 0 & 0 & 0 & 0 \\ 6 & 5 & 5 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.7}$$

The minimum values of $Rq_{i,j}$ are the opposite number of elements in matrix 5.7.
According to the quantization table in Fig. 4.3(a), we store $Rq_{i,j}$ and $Diff_{i,j}$:

$$\begin{bmatrix} 9 & 9 & 9 & 9 & 5 & 0 & 0 & 0 \\ 9 & 9 & 9 & 5 & 4 & 0 & 0 & 0 \\ 9 & 9 & 7 & 5 & 0 & 0 & 0 & 0 \\ 9 & 6 & 5 & 0 & 0 & 0 & 0 & 0 \\ 7 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.8}$$

In general, the way to store 10 $Diff_{i,j}$ and $Rq_{i,j}$ is shown in matrix 5.9

$$\begin{bmatrix} Diff_{00} & Diff_{01} & Diff_{02} & Diff_{03} & Rq_{20} & 0 & 0 & 0 \\ Diff_{10} & Diff_{11} & Diff_{12} & Rq_{00} & Rq_{03} & 0 & 0 & 0 \\ Diff_{20} & Diff_{21} & Rq_{02} & Rq_{11} & 0 & 0 & 0 & 0 \\ Diff_{30} & Rq_{10} & Rq_{12} & 0 & 0 & 0 & 0 & 0 \\ Rq_{01} & Rq_{21} & 0 & 0 & 0 & 0 & 0 & 0 \\ Rq_{30} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.9}$$

For Cb or Cr component, we test d_2 from 9 to 21 using the same process, and we choose 15 as the divisor d_2.

5.2.2 Weibo

As Facebook, Weibo also carries out the complete processes of JPEG compression and decompression when we upload an image. So we use the same method to protect published image on Weibo as on Facebook.

We found that Weibo requantizes the published images only if the quality is larger than $Q = 95$. Therefore, if we requantize the images using quantization tables of $Q = 71$, the compression quality will not be limited during publishing. That means, we can use the parameters which are applied on Facebook to prepare the correcting code for Weibo.

5.2.3 Wechat

Wechat requantized the published images using the quantization tables shown in Fig. 4.2

After being quantized by these quantization tables, the ranges of quantized DCT coefficients in one block are shown in Fig. 5.6.

For Wechat, we do the same test as we have done for Facebook. We calculate the extremum ranges for each chose divisor from 9 to 23, and list in Tab 5.2.

We find that when $d_1 = 19$, the ranges are closest to $[-128, 127]$ after reverse DCT.

Therefore, for Wechat we choose 19 as the divisor d_1 for Y component or grayscale image. The maximum number of $Diff_{i,j}$ is 9, and the maximum values of $Rq_{i,j}$ are:

$$\begin{bmatrix} 5 & 4 & 5 & 2 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.10}$$

The minimum values of $Rq_{i,j}$ are the opposite number of elements in matrix 5.10.
According to the quantization table in Fig. 4.2, we store $Rq_{i,j}$ and $Diff_{i,j}$:

$$\begin{bmatrix} -114 & -103 & -105 & -58 & -46 & -31 & -25 & -20 \\ -103 & -93 & -95 & -52 & -42 & -28 & -22 & -18 \\ -105 & -95 & -54 & -39 & -31 & -22 & -19 & -16 \\ -58 & -52 & -39 & -28 & -24 & -18 & -16 & -13 \\ -46 & -42 & -31 & -24 & -22 & -17 & -15 & -13 \\ -31 & -28 & -22 & -18 & -17 & -13 & -12 & -12 \\ -25 & -22 & -19 & -16 & -15 & -12 & -12 & -12 \\ -20 & -18 & -16 & -13 & -13 & -12 & -12 & -12 \end{bmatrix}$$

(a) The minimum values of luminance table

$$\begin{bmatrix} 113 & 103 & 105 & 58 & 46 & 31 & 25 & 20 \\ 103 & 93 & 95 & 52 & 42 & 28 & 22 & 18 \\ 105 & 95 & 54 & 39 & 31 & 22 & 19 & 16 \\ 58 & 52 & 39 & 28 & 24 & 18 & 16 & 13 \\ 46 & 42 & 31 & 24 & 22 & 17 & 15 & 13 \\ 31 & 28 & 22 & 18 & 17 & 13 & 12 & 12 \\ 25 & 22 & 19 & 16 & 15 & 12 & 12 & 12 \\ 20 & 18 & 16 & 13 & 13 & 12 & 12 & 12 \end{bmatrix}$$

(b) The maximum values of luminance table

$$\begin{bmatrix} -93 & -77 & -52 & -30 & -14 & -12 & -13 & -12 \\ -77 & -49 & -50 & -16 & -12 & -11 & -11 & -11 \\ -52 & -50 & -21 & -11 & -13 & -11 & -12 & -11 \\ -30 & -16 & -11 & -11 & -12 & -11 & -11 & -11 \\ -14 & -12 & -13 & -12 & -14 & -12 & -13 & -12 \\ -12 & -11 & -11 & -11 & -12 & -11 & -11 & -11 \\ -13 & -11 & -12 & -11 & -13 & -11 & -12 & -11 \\ -12 & -11 & -11 & -11 & -12 & -11 & -11 & -11 \end{bmatrix}$$

(c) The minimum values of chrominance table

$$\begin{bmatrix} 92 & 77 & 52 & 30 & 14 & 12 & 13 & 12 \\ 77 & 49 & 50 & 16 & 12 & 11 & 11 & 11 \\ 52 & 50 & 21 & 11 & 13 & 11 & 12 & 11 \\ 30 & 16 & 11 & 11 & 12 & 11 & 11 & 11 \\ 14 & 12 & 13 & 12 & 14 & 12 & 13 & 12 \\ 12 & 11 & 11 & 11 & 12 & 11 & 11 & 11 \\ 13 & 11 & 12 & 11 & 13 & 11 & 12 & 11 \\ 12 & 11 & 11 & 11 & 12 & 11 & 11 & 11 \end{bmatrix}$$

(d) The maximum values of chrominance table

Figure 5.6: The range of quantized DCT coefficients in one block (quantized by quantization tables of Wechat).

$$\begin{bmatrix} 9 & 9 & 9 & 9 & 4 & 2 & 0 & 0 \\ 9 & 9 & 9 & 5 & 4 & 0 & 0 & 0 \\ 9 & 9 & 5 & 4 & 0 & 0 & 0 & 0 \\ 9 & 5 & 2 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.11}$$

In general, the way to store 10 $Diff_{i,j}$ and $Rq_{i,j}$ is shown in matrix 5.12

$$\begin{bmatrix} Diff_{00} & Diff_{01} & Diff_{02} & Diff_{03} & Rq_{10} & Rq_{03} & 0 & 0 \\ Diff_{10} & Diff_{11} & Diff_{12} & Rq_{02} & Rq_{11} & 0 & 0 & 0 \\ Diff_{20} & Diff_{21} & Rq_{20} & Rq_{12} & 0 & 0 & 0 & 0 \\ Diff_{30} & Rq_{00} & Rq_{30} & 0 & 0 & 0 & 0 & 0 \\ Rq_{01} & Rq_{21} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.12}$$

For Cb or Cr component, we test d_2 from 7 to 23 using the same process, and we choose 11 as the divisor d_2.

5.3 Experimental results of improved encryption algorithm

In order to reduce the losses of recovered images in the previous chapter, we proposed an improved encryption algorithm in this chapter. The parameters of the improved algorithm depend on the characteristics of the different image-sharing platforms. We have done some test and determined the parameters for Facebook, Weibo and Wechat. In this section, we use the improved encryption algorithm to do the experiments on these three platforms. LibJPEG is used to encode and decode JPEG image, and all experiments were implemented in C/C++. We first take the grayscale image and color image "Lena" of 512×512 pixels as examples, and then we take ten grayscale images and ten color images of different sizes from 480×640 pixels to 2736×3648 pixels to do more tests.

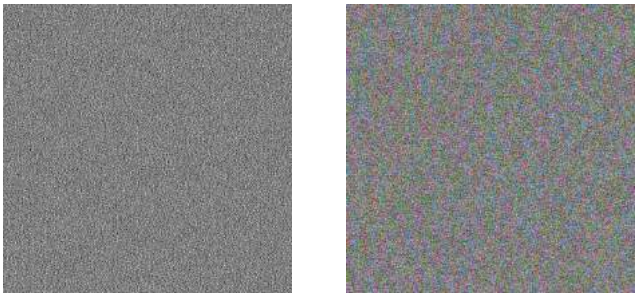
Divisor	Range
9	$[-307, 307]$
10	$[-319, 303]$
11	$[-301, 301]$
12	$[-324, 308]$
13	$[-311, 311]$
14	$[-323, 313]$
15	$[-304, 304]$
16	$[-314, 309]$
17	$[-300, 300]$
18	$[-303, 297]$
19	$[-297, 297]$
20	$[-309, 303]$
21	$[-301, 301]$
22	$[-311, 305]$
23	$[-311, 311]$

Table 5.2: The extremum values of coefficients after reverse DCT for different divisors (Wechat).

5.3.1 Facebook

Before encryption, we resize prepared images to the limitation image size of Facebook (2048 pixels) and change the downsampling ratio for color images to $4:2:0$ if they do not satisfy the conditions. Then we requantize the images using quantization tables of $Q = 71$. We encrypt the images using our algorithm on the laptop, the results are shown in Fig 5.7(a) and Fig 5.7(b). Then we upload the encrypted images to Facebook on OS X 10.10 with safari browser and use another account to download them. After download these images, the decryption is implemented on the laptop, and we get decrypted images as shown in Fig 5.9(a) and Fig 5.9(b).

We summarize the results of "Lena" in Tab. 5.3, and the average results of other 20 images in Tab. 5.4. As shown in Tab. 5.3, the decryption algorithm can reconstruct the plaintext grayscale "Lena" with PSNR of 32.41dB and with UIQI of 0.8602, and reconstruct the plaintext color "Lena" with PSNR of 29.23dB and with UIQI of 0.7965. If we decrypt these two encrypted images directly without publishing on Facebook, the results are shown in Fig 5.8(a) and Fig 5.8(b). The PSNR values are 32.41dB and 29.42dB, and UIQI values are 0.8602 and 0.7965. Comparing them with the PSNR and UIQI values after publishing the images on Facebook, the difference is very small. That means, the recompression has only a little impact on the results, our algorithm can reconstruct an image with a high quality even after publishing it on Facebook and our algorithm can be compatible with JPEG recompression. For the other 20 images, as shown in Tab. 5.4, we have the same conclusion.



(a) Encrypted grayscale image. (b) Encrypted color image.

Figure 5.7: Experiment results of encryption($d_1 = 19$, $d_2 = 15$).

(a) Decrypted grayscale image.

(b) Decrypted color image.

Figure 5.8: Experiment results of decryption without publishing ($d_1 = 19$, $d_2 = 15$).

(a) Decrypted grayscale image from Facebook.

(b) Decrypted color image from Facebook.

Figure 5.9: Experiment results of decryption on Facebook.

5.3.2 Weibo

Before encryption, we resize prepared images to the limitation image size of Weibo (1024 pixels) if they are not satisfy the conditions and we use the quantization tables of $Q = 71$ to requantize the images. We encrypt the images using our algorithm on the laptop. Then we upload the encrypted images to Weibo on OS X 10.10 with safari browser and use another account to download them. After download these images, the decryption is implemented on the laptop, and we get decrypted images as shown in Fig 5.10(a) and Fig 5.10(b).

From the decrypted images, we find that Weibo adds a logo in the lower right corner of the published image. After we download and decrypt the image, it is impossible to recover this region of the plaintext image. We summarize the results of "Lena" in Tab. 5.5, and the average results of other 20 images in Tab. 5.6. As shown in Tab. 5.5, the decryption algorithm can reconstruct the plaintext grayscale "Lena" with PSNR of 23.63dB and with UIQI of 0.8360, and reconstruct the plaintext color "Lena" with PSNR of 23.90dB and with UIQI of 0.7739. If we decrypt these two encrypted images directly without publishing on Weibo, the results are shown in Fig 5.8(a) and

	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Decryption after publishing	32.41	0.8602	29.23	0.7965
Decryption without publishing	32.41	0.8602	29.42	0.7965

Table 5.3: Summary of the results for "Lena" (Facebook).

	Grayscale image		Color image	
	Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Decryption after publishing	28.94	0.7994	30.16	0.7936
Decryption without publishing	28.94	0.7994	30.29	0.7945

Table 5.4: Summary of the average results for 20 other images (Facebook).

(a) Decrypted grayscale image from Weibo.

(b) Decrypted color image from Weibo.

Figure 5.10: Experimental results of decryption on Weibo.

Fig 5.8(b). The PSNR values are 32.41dB and 29.42dB, and UIQI values are 0.8602 and 0.7965. Comparing encrypted images without publishing with the images after publishing on Weibo, the difference is a little large, because of the unrecovered logo in the lower right corner. Therefore, we try to calculate the similarity between the decrypted image in addition to the logo part and the original one. The results are summarized in Tab. 5.7 and Tab. 5.8. The PSNR values are around 28∼33 dB and the UIQI values are around 0.7∼0.9. Comparing decrypted images without publishing with the images after publishing on Weibo, the difference is very small. That means, the recompression has only a little impact on the results, our algorithm can reconstruct an image with a high quality even after publishing it on Weibo and our algorithm can be compatible with JPEG recompression except for the generated logo.

	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Decryption after publishing	23.63	0.8360	23.90	0.7739
Decryption without publishing	32.41	0.8602	29.42	0.7965

Table 5.5: Summary of the results for “Lena” (Weibo with logo).

5.3.3 Wechat

For Wechat, we can only use their applications for smartphone to publish images. The test environment for smartphone is iOS 9.3.1. The version of Wechat is 6.3.15. Before encryption, we resize prepared images to the limitation image size of Wechat (1280 pixels) and change the downsampling ratio for color images to 4:2:0 if they are not satisfy the conditions. And we requantize the images

	Grayscale image		Color image	
	Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Decryption after publishing	21.19	0.7844	21.76	0.7800
Decryption without publishing	28.87	0.8053	30.00	0.8049

Table 5.6: Summary of the average results for 20 other images (Weibo with logo).

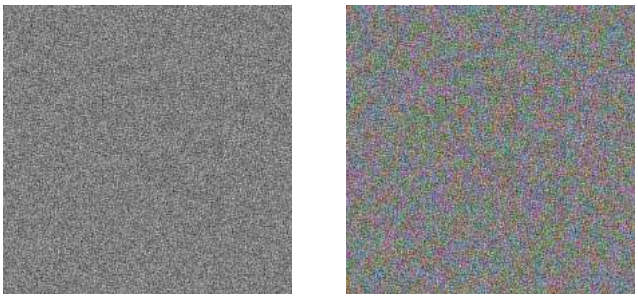
	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Decryption after publishing	32.46	0.8664	29.39	0.7935
Decryption without publishing	32.46	0.8664	29.62	0.7943

Table 5.7: Summary of the results for "Lena" (Weibo without logo).

	Grayscale image		Color image	
	Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Decryption after publishing	28.99	0.8325	29.59	0.8352
Decryption without publishing	29.01	0.8328	30.22	0.8376

Table 5.8: Summary of the average results for 20 other images (Weibo without logo).

using its own quantization tables. We encrypt the images using our algorithm on the laptop, the results are shown in Fig 5.11(a) and Fig 5.11(b). Then we send the encrypted images via a lossless way, i.e. e-mail or "Dropbox", to a smartphone. In this way, images are not processed before being published. Next we upload images to Wechat using their native smartphone applications. When we download the images, we use another account on smartphone and send the images back to laptop via a lossless way. Then the decryption is implemented on the laptop, and we get decrypted images as shown in Fig 5.13(a) and Fig 5.13(b).



(a) Encrypted grayscale image. (b) Encrypted color image.

Figure 5.11: Experiment results of encryption($d_1 = 19$, $d_2 = 11$).

(a) Decrypted grayscale image. (b) Decrypted color image.

Figure 5.12: Experiment results of decryption without publishing ($d_1 = 19$, $d_2 = 11$).

(a) Decrypted grayscale image from Wechat. (b) Decrypted color image from Wechat.

Figure 5.13: Experimental results of decryption on Wechat.

We summarize the results of "Lena" in Tab. 5.9, and the average results of other 20 images in Tab. 5.10. As shown in Tab. 5.9, the decryption algorithm can reconstruct the plaintext grayscale "Lena" with PSNR of 32.13dB and with UIQI of 0.8335, and reconstruct the plaintext color "Lena" with PSNR of 29.23dB and with UIQI of 0.7773. If we decrypt these two encrypted images directly

without publishing on Wechat, the results are shown in Fig 5.12(a) and Fig 5.12(b). The PSNR values are 32.30dB and 29.34dB, and UIQI values are 0.8366 and 0.7810. Comparing them with the PSNR and UIQI values after publishing the images on Wechat, the difference is very small. That means, the recompression has only a little impact on the results, our algorithm can reconstruct an image with a high quality even after publishing it on Wechat and our algorithm can be compatible with JPEG recompression. For the other 20 images, as shown in Tab. 5.10, we have the same conclusion.

	Grayscale image		Color image	
	PSNR (dB)	UIQI	PSNR (dB)	UIQI
Decryption after publishing	32.13	0.8335	29.23	0.7773
Decryption without publishing	32.30	0.8366	29.34	0.7810

Table 5.9: Summary of the results for "Lena" (Wechat).

	Grayscale image		Color image	
	Average PSNR (dB)	Average UIQI	Average PSNR (dB)	Average UIQI
Decryption after publishing	28.87	0.7783	29.97	0.7851
Decryption without publishing	29.09	0.7823	30.67	0.7920

Table 5.10: Summary of the average results for 20 other images (Wechat).

5.4 Performance Evaluation

In this section, we evaluate the performances of the improved encryption algorithm through the analyses of the complexity of the algorithm and its execution time.

Twenty grayscale images of different sizes from 256×256 pixels to 15408×8184 pixels were prepared for the tests. For color images, since our algorithm is used to protect the images published on Facebook, their downsampling ratio may not be the limitation ratio of Facebook, i.e. 4:2:0. In this case, our algorithm will change the ratio during encryption, and the performances may be affected. Therefore, we prepared 20 color images with downsampling ratio of 4:2:0 and 20 color images with other downsampling ratios (e.g. 4:4:4, 4:2:2, 4:1:1, 4:1:0) of different sizes, from 256×256 pixels to 15408×8184 pixels.

We experimented with an Intel architecture: tests were performed on an Intel Core i7-3615QM CPU with four cores running at 2.30 GHz. The size of memory is 4 GB. The operating system is MAC OS X 10.10.5. Our tests were on a completely unloaded system (executing only the operating system and the encryption and decryption programs).

5.4.1 The complexity of algorithm

The complexity of the algorithm is a function of the length of the string representing the input, which quantitatively describes the running time of the algorithm [Sip06]. We estimate the complexity of algorithm by counting the number of elementary operations performed by our code, and we get to know that our algorithm only has a complexity of $O(n^2)$ to encrypt or decrypt an image, where n is the image width or length. For most of the traditional cryptosystems (i.e. DES, Triple

DES, AES, RC4, Blowfish), if we use them to encrypt an image, we usually get an $O(n^2)$ complexity as well, where n is the image width or length. Therefore, the complexity of our algorithm is not high, and it is a relatively simple algorithm.

5.4.2 Execution time

We encrypt and decrypt all the prepared images using C++, and calculate their execution times. In the following subsections, the illustrations show the relationship between image size and execution time of encryption and decryption.

In each Figure, we analyze the relationship between the image size and the execution time. The abscissa values are based on the image sizes in ascending order.

5.4.2.1 Gray image

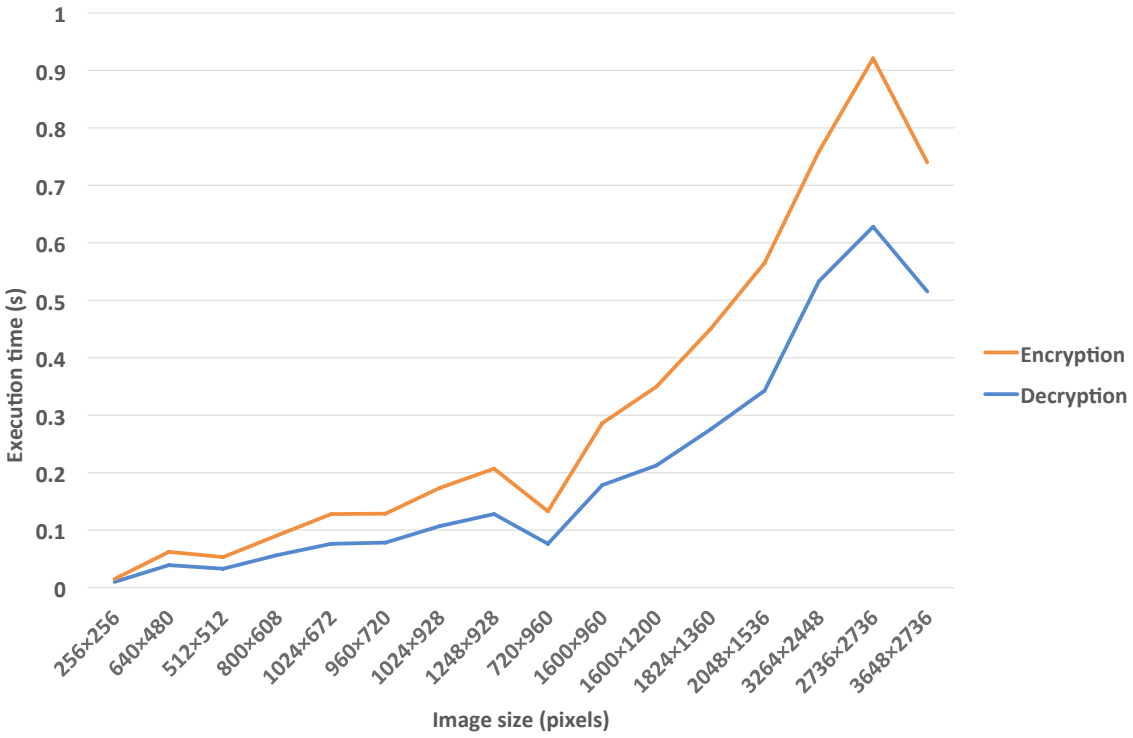


Figure 5.14: The encryption and decryption execution times of gray images of different sizes.

The image size is resized during encryption in the program C++ if its longer side exceeds 2048. But there is an limitation in the program C++, the resizing factor cannot exceed 0.5. So if the longer side of image is greater than 4096, we cannot get the test results. Generally speaking, the decryption takes less time than the encryption, from 5ms to 293 ms, and the encryption takes 9ms to 628 ms.

As shown in Fig. 5.14, we find that when the image size is less than 2048, as image size increases, the execution time grows accordingly, and they are linear relationship. But if the image size is greater than 2048, the execution time depends on the image size after resizing.

5.4.2.2 Color image (with downsampling ratio 4:2:0)

As explained above, the limitation of program C++ is the same for processing the color image. If the longer side of image is greater than 4096, we cannot get the test results. Generally speaking, the decryption takes less time than the encryption, from 8ms to 548ms, and the encryption takes 14 ms to 1.1 s. It takes more time to encrypt and decrypt the color images than to encrypt and decrypt the gray images.

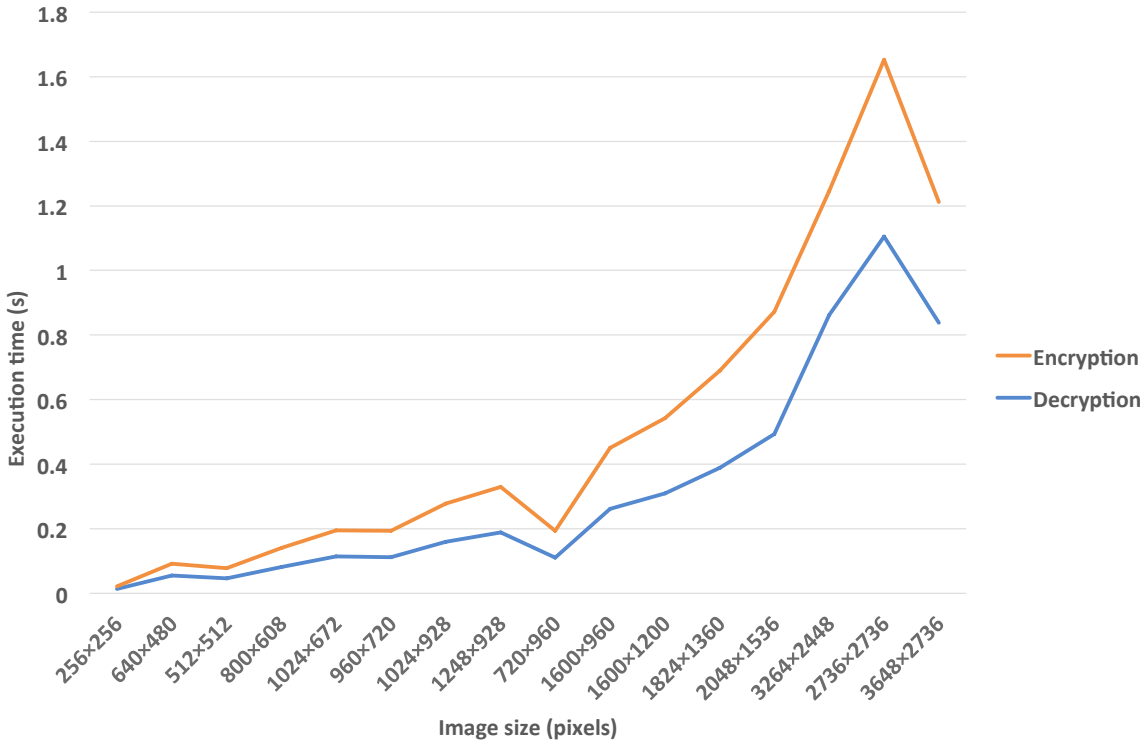


Figure 5.15: The encryption and decryption execution times of color images with downsampling ratio 4:2:0 of different sizes.

As shown in Fig. 5.15, we find that when the image size is less than 2048, as image size increases, the execution time grows accordingly, and they are linear relationship. But if the image size is greater than 2048, the execution time depends on the image size after resizing.

5.4.2.3 Color image (with other downsampling ratios)

Generally speaking, the decryption takes less time than the encryption, from 8 ms to 568 ms, and the encryption takes 32 ms to 1.5s. Changing the downsampling ratio only takes a little more time in hundreds of milliseconds maximally to encrypt and decrypt the color images.

As shown in Fig. 5.16, we find that when the image size is less than 2048, as image size increases, the execution time grows accordingly, and they are linear relationship. But if the image size is greater than 2048, the execution time depends on the image size after resizing.

We implement this encryption algorithm in our application to protect the images on Facebook. Users can use our application to encrypt the images and the encrypted images can be decrypted directly on the webpage. Therefore, the execution time to display the encrypted images and recovered images is very important for user's experience. Huawei [1] studied the perception of people when they wait a text or a picture display in the process of web browsing. They use "zero waiting time" to describe that user cannot detect the presence of waiting for page display if the display time is less than a threshold. After some experiments, they determined that the average value of this threshold is 270 ms.

According to the analyses above, for a grayscale image, the decryption takes maximally 293 ms, which only exceeds the threshold 23 ms. The encryption takes maximally 628 ms, but if the image size is smaller than 1824×1360 pixels, the encryption time is within the threshold. For a color image, the decryption takes maximally 568 ms, but if the image size is smaller than 1824×1360 pixels, the decryption time is within the threshold. The encryption takes maximally 1.5 s, but if the image size is smaller than 960×720 pixels, the encryption time is within the threshold.

[1] http://www.huawei.com/mediafiles/CBG/PDF/Files/hw_364998.pdf

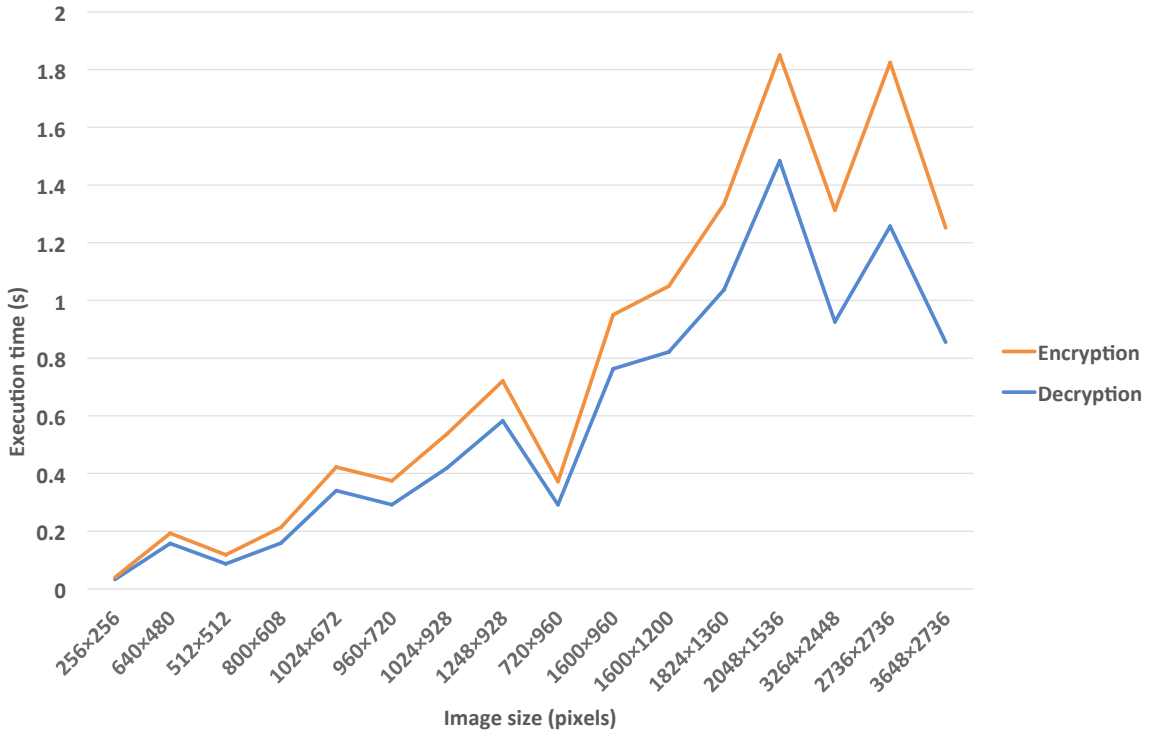


Figure 5.16: The encryption and decryption execution times of color images with other downsampling ratio of different sizes.

Therefore, in addition to encrypt or decrypt biggest images, users can have a "zero waiting time" experience using our algorithm.

5.5 Conclusion

In this chapter, the improved encryption algorithm is proposed to overcome the restriction of basic encryption algorithm. A correcting code is added during encryption based on the characteristic of different platforms. The improved algorithm can ensure that the range of uploaded coefficients is close to $[-128, 127]$ after reverse DCT. In this way, we can minimize the losses of image information during JPEG recompression. After introducing the general algorithm, we determine the most appropriate parameters for three image-sharing platforms (Facebook, Weibo, Wechat) according to their characteristics.

Next, we take some images to do the experiments by using the improved algorithm on Facebook, Weibo and Wechat. The results show that the improved algorithm can reconstruct the downloaded images with a high quality and also similar to the decrypted images without publishing. We can conclude that, although the improved algorithm should depend on the characteristics of each platforms, which is not very general, it can overcome the restriction at least for Facebook, Weibo and Wechat.

In the end, we evaluate the performances of the improved algorithm. Firstly we analyze the complexity of encryption and decryption algorithms for gray and color images. Next we calculate the execution time of program C++ for different sizes images and find the relationship between the execution time and the image size. From the results, we know that our algorithm only has a complexity of $O(n^2)$, which is not high by comparing with other traditional cryptosystems, so this is a relatively simple algorithm. When we use the C++ program to realize our algorithm, it only takes 1.5s maximally to encrypt a color image, the decryption even takes less time. It can be accepted by user's experience. When we decrypt the images of small size until the plaintext images display on the webpage, the users can have a "zero waiting time" experience.

Chapter 6

Application: PixGuardian

Contents

6.1 The scenario

We aim to protect user's image on the image-sharing platforms. Depending on our encryption algorithm, we build an application named PixGuardien to encrypt the images. It also offers users

to define the access conditions of the image and to manage the secret keys. Then the users can publish the encrypted image on Facebook (until now, the image encrypted by this application can be only published on Facebook). For the people who are allowed to access the image, they can use our plugin which is installed in the browser Chrome to decrypt and see the plaintext image.

6.1.1 The inscription and generation of keys

6.1.1.1 General principle

Our principle is to share an image JPEG securely between users through Internet. Each user can use the installed application to have the access to the server which has the authority role of certification. The server distributes the encrypted session keys to decrypt the encrypted image from the Internet. One user (we call him Bob) can decrypt the image if all the access conditions of the image which is defined by the sender (the person who publishes the encrypted image, we call her Alice) are permitted.

To make it simple, in the following paragraphs, we use the same pair of public key / private key to encrypt, sign and authenticate. However, normally each entity have two pairs of keys: one for encrypting and signing, another one for authentication.

6.1.1.2 Generation of the public / private keys

Each participator must register to the server and generate the keys at the beginning. After installing the application, an SSL link is established between the server and the application of user Alice by using the public key K_S of server. Alice is invited to choose a username and to provide an email address. Alice's application generates a pair of public key / private key randomly, and sends the username, email address and Alice's public key to the server.

The server then creates an account for Alice with her username, email address and certificate $Cert_A$. This certificate contains Alice's username, email address and her public key. The server sends $Cert_A$ to Alice's application.

For the private key, either Alice stores it and uses her pair of private key / public key to authenticate to the server, or a password is created to encrypt and store the private key locally. In the second case, Alice must type in her username and password in order to decrypt the private key and thus be able to authenticate.

6.1.1.3 User's certification

Where, S is the serve, App is the application, A is the user Alice. $Cert_A$ is the certificate of A created by the server, $Cert_A = (usernameA, a@mail.com, Ka)$. After the step 4, the application generates a pair of public key (Ka) / private key for A. The server also stores A's certificate. The step 5 and 6, represented by the red arrows, which means the exchanges are performed by SSL channel. This representation works in the following paragraphes as well.

6.1.1.4 Sharing the certificate and generation of symmetric keys

To share an image with specific people, Alice must obtain the certificate of these people. From the interface, it is possible for Alice to search by using the username or email address to get the certificate she needs.

To encrypt an image, Alice drops the image on the interface and selects who are allowed to see the image. Then she defines the access conditions of the image, general and specific for the selected people. The application generates a random symmetric session key to encrypt the image .

6.1.1.5 Obtain the certificates

After a mutual authentication between Alice's application and the server, Alice types in the username or email address of a friend (e.g. Bob) in the search bar on the interface and selects Bob's profile. The server searches the account of Bob in its database. If find, the server informs Bob

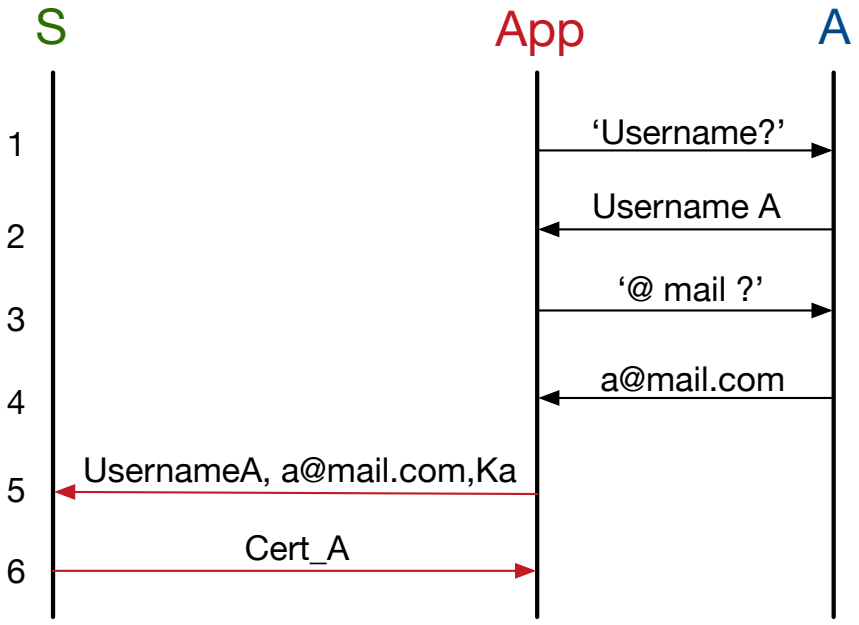


Figure 6.1: The certification of A and generation/distribution of key

about Alice's request. When Bob is connected, his application and the server authenticate mutually. Bob receives Alice's request sent by the server, and Bob accepts to give or not to give his certificate to Alice. If he accepts, the server sends to Alice $Cert_B$ and sends to Bob $Cert_A$ (Shown in Fig. 6.2). Otherwise, nobody can receive the certificate, and Alice receives the negative response from Bob sent by the server.

Where A is Alice, S is the server, B is Bob.

In step 2, Alice can find Bob according to his username or his email address,. In step 5, Bob also can refuse the request and then the server informs Alice Bob's refusal instead of sending the certificates.

6.1.2 General architecture

The scheme (Fig. 6.3) represents a general architecture about sharing an image with a specified person (here is Bob).

Where,

- k_P: the private key of participator P
- K_P: the public key of participator P
- $\{M\}_K$: the encryption of the object M with the key K
- K_{ses}: the symmetric session key to encrypt image
- $Cert_P$: the certificat of participator P
- $AC_{P \to U}$: the access conditions of an image defined by P for U
- $Sign\_{}_U$: the signature of $\{id, U, \{K_{ses}\}_{K_U}, AC_{P \to U}\}$ of P for U

The step 2 and step 6 are realised after the mutual authentification between the server and the applications. After the step 2, if the verification is failed, the server returns an error message. The same as the step 7, if Bob is not authorised to see the image, the server returns an error message.

The exchanges between the applications/plugins and the server are secure by using the channel SSL.

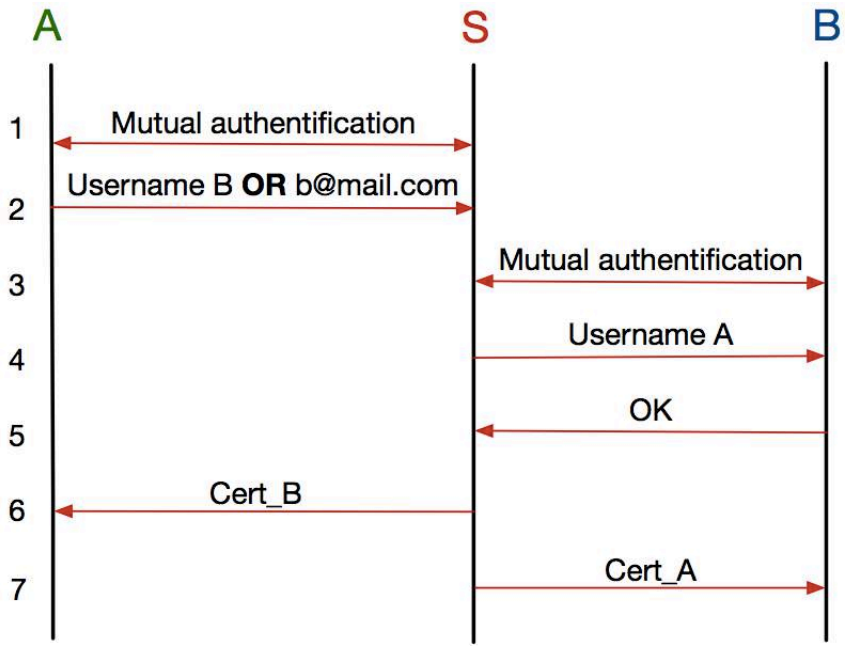


Figure 6.2: Certificates exchange

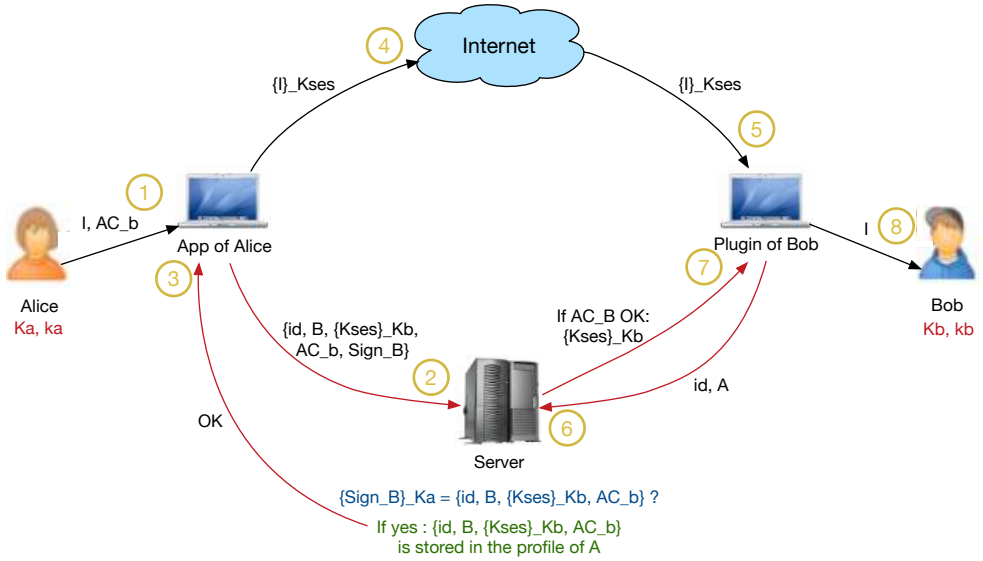


Figure 6.3: General architecture for sharing with a person

Each participator P installs the application to carry out the encryptions / decryptions, generates the identity of an image id and the session keys. The maximum size of the identity is 256 bits and it is obtained by the application.

For each participator P, the server stores his certificate $Cert_P$, his e-mail address and his public key K_P. The server also stores all the quadruplets representing an image for each receiver. In each quadruplet, we have: the image identity id, the person identity U, the session key encrypted by a public key K_U of U and the access conditions of the image $AC_{P \to U}$ of P for U.

The application/plugin works like that:

- generate a session key K_{ses}.
- get the identity id of image I.
- encrypt the image I with the session key using the encryption function C, i.e. $C(I, K_{ses}) = \{I\}_{K_{ses}}$.
- decrypt the encrypted images with the public key K using the decryption fonction $InvC$, i.e. $InvC(C(I, K_{ses}), K_{ses}) = I$.
- exchange with the server by using a secure channel SSL.

The application allows to register (certification) to the server, drop the image to encrypt and generate the encrypted image which can be used to published on the image-sharing platforms. After dropping an image, the user needs to choose the access conditions of the image.

6.1.3 Upload the image

There are two cases when we upload the image. The first one is that the access to images is restricted to few people, the second one is that everyone can access the image (during a defined period or only with numbers of views). Indeed, the exchange and storage of data are slightly different depending on the cases, in the following paragraphs, we detail the distinctions.

6.1.3.1 The case of access to images for restricted people

First of all, Alice and Bob have their own certificate and are registered to the server who knows their usernames, their email addresses and their public keys. And they have their public keys respectively. Alice can exchange her certificate with friends, such as Bob, and get their certificates.

Alice wants to share an image I with Bob. After being connected, she drops the image I on the interface of application. Then the application generates a session key and calculates identity of the image id. Then Alice defines the access conditions of this image.

An access condition can be authorised and/or refused to display the image:

- by some people.
- during a limited time.
- with a limited numbers of view.
- OR a combination of the three conditions above.

The application knows the public keys of authorised persons according to their certificate. It encrypts the session key using the public key of each person respectively.

Example: If Alice allows Bob and Charlie to see her image, then the application creates $\{K_{ses}\}_{K_B}$ and $\{K_{ses}\}_{K_C}$.

Also, each authorised person is associated with the access conditions of the image. Using the example above, the access conditions of the image for B and C are respectively created and represented by $AC_{A \to B}$ and $AC_{A \to C}$.

After opening a SSL channel to the server (and mutual authentication between the application of Alice and the server), for each authorised person U, the application sends to the server the quintuplet: $(id, U, \{K_{ses}\}_{K_U}, AC_{A \to U}, Sign_U)$.

Where $Sign_U$ represents the signature of the other four elements with the private key of A. That is : $Sign_U = \{id, U, \{K_{ses}\}_{K_U}, AC_{A\to U}\}_{K_A}$.

After receiving all quintuplets, the server checks each of them by ensuring that $\{Sign_U\}_{K_A} = \{id, U, \{K_{ses}\}_{K_U}, AC_{A\to U}\}$. The server can do this verification because it owns the public key of Alice K_A in her certificate $Cert_A$. After the verification, the server stores the quintuplets $\{id, U, \{K_{ses}\}_{K_U}, AC_{A\to U}\}$ in the profile of Alice and confirms the storage of quintuplets to Alice, otherwise it returns an error message.

Then Alice gets her encrypted image $\{I\}_{K_{ses}}$ and shares it on the Internet, such as, on the social networks (Explained in Fig. 6.4).

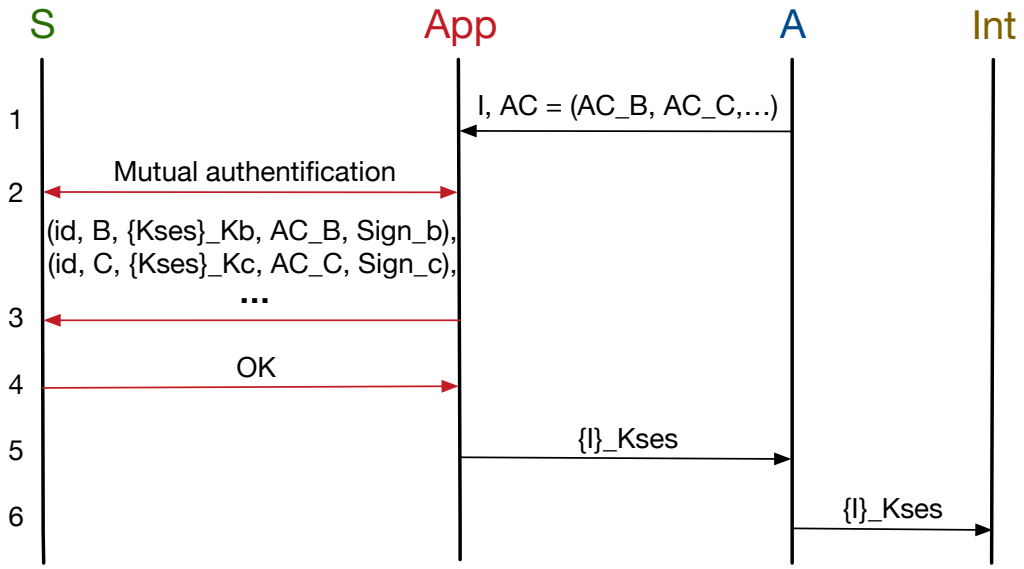


Figure 6.4: Encryption of the image I

Where S is the server, App is the application, A is the entity of Alice and Int is the internet. AC represents all the access conditions of the image. Each AC_icorresponds to the specific access conditions for the individual i. Note that: $Sign\_i = \{id, i, \{K_{ses}\}_{K_i}, AC_{A\to i}\}_{k_A}$.

After step 1, App generates the session key, calculates the id, encrypts the session key with the public keys of the persons who are allowed to see the image and to create signatures.

Step 2 represents the mutual authentication via SSL channel. Finally, in step 4, the server can also return an error if there is one during the signature verification. In this case, either the previous step is repeated, or the protocol stops.

6.1.3.2 The case of access to images for everyone

This is the case that the user wants everyone to see his images but during a limited time and/or with a limited numbers of view.

In this case, as everyone can see the image, the server can as well. The session key is stored in the server without being encrypted. The application only sends the session key K_{ses}, the access conditions AC (the general conditions for the image) and $Sign_S = \{K_{ses}, AC\}_{K_A}$ (Explained in Fig. 6.5). Where S is the server, App is the application, A is the entity of Alice and Int is the internet. AC represents all the access conditions of the image. Note that: $Sign\_s = \{K_{ses}, AC\}_{k_A}$.

As before, in step 4, the server may return an error if the signature verification is not success. In this case, either the application renews the previous sending, or the protocol stops.

Note that even if the access conditions of image are expired, the server always owns the session key and can still decrypt the image.

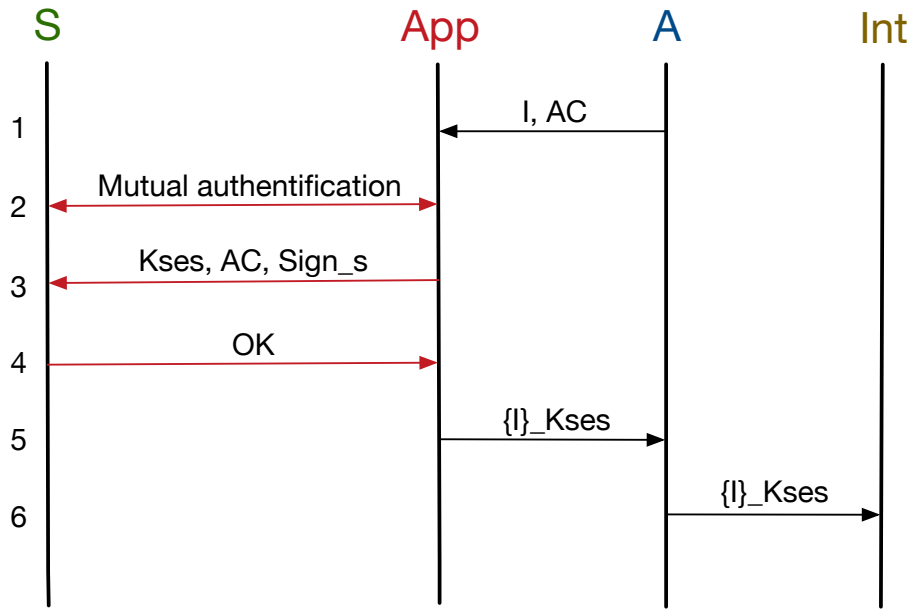


Figure 6.5: Encryption of the image I, version 2

6.1.4 Download the image $\{I\}_{K_{ses}}$ with decryption

6.1.4.1 The case of access to images for restricted people

In this case, Bob sees the shared image $\{I\}_{K_{ses}}$ by Alice on the website. His plugin opens a SSL channel to the server with a mutual authentification.

The plugin calculates the identity of the image *id* and sends it with the username of Alice to the server. The server looks for this *id* under the username of Alice. If found, the server checks the access conditions for Bob. If the access conditions allow Bob to decrypt the image, then the server returns $\{K_{ses}\}_{K_B}$ to Bob. Otherwise, the server returns an error message. In the positive case, the plugin decrypts $\{K_{ses}\}_{K_B}$ with the private key of Bob K_B to obtain the session key. Finally, $\{I\}_{K_{ses}}$ is decrypted and Bob views the image I (Explained in Fig. 6.6).

Each authorised user who wants to see the image must do the same procedure to decrypt $\{I\}_{K_{ses}}$ until the general access conditions are expired (for example: time expired, the numbers of view expired).

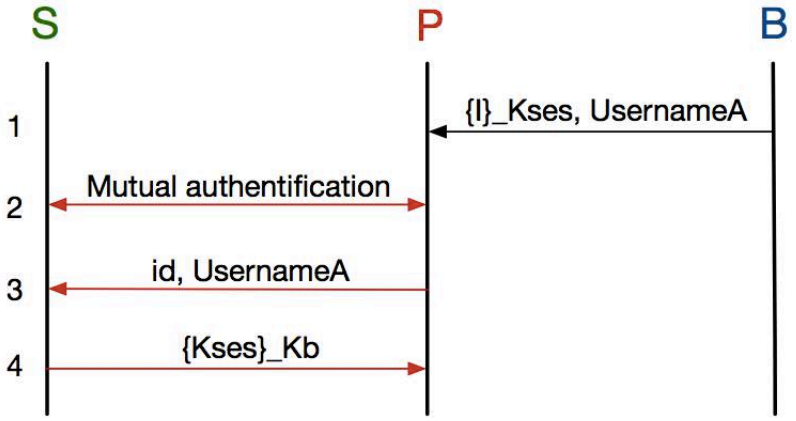


Figure 6.6: Decryption of the image I

Where: S is the server, P is the plugin, B is the entity of Bob.
After step 3, the server verifies the access conditions of the image for B. If Bob can see the image

then step 4 is performed. Otherwise, the server returns an error message instead of $\{K_{ses}\}\_K_b$. After Step 4, the plugin finds the session key and decrypts the image.

Here, Bob knows the image publisher when sending the username. After decryption, the session keys are not be saved by the plugin.

6.1.4.2 The case of access to images for everyone

In this case, after having the identity of the image and the username of Alice, the server verifies the access conditions. If it is still possible to see, the server returns to the plugin the unencrypted session key, but via the secure SSL channel (Explained in Fig. 6.7).

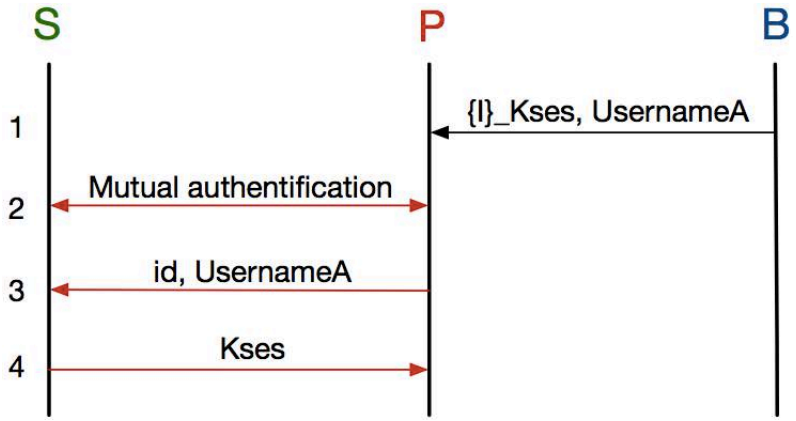


Figure 6.7: Decryption of the image I, version 2

Where: S is the server, P is the plugin, B is the entity of Bob.

After step 3, the server verifies the access conditions of the image. If the image is still visible, then step 4 is performed. Otherwise, the server returns an error message instead of K_{ses}. After step 4, the plugin decrypts the image.

6.2 The application to publish images

In this section, we show how to turn the scenario into the real application/plugin and use them to share image on Facebook. In order to fully protect user's privacy, we choose to realise the case of access to images for restricted people, because considering in the case of access to images for everyone, the server always owns the session key and can always decrypt the image, so the server cannot be trusted by the users.

6.2.1 Registration and login

First of all, users need to register in the server (shown in Fig. 6.8) by using the application. The authentication is established between the server and the users, and then the application helps users generate their own pairs of keys. The public keys are sent to and stored in the server.

When the user who wants to share his images with restricted people, he needs to ask the server for the public key of each authorised user. The server then asks these users for the authorisation of providing the public key. (shown in Fig. 6.9).

The interaction of users' login and registration in our application is shown in Fig. 6.10. The registration information are stored in our database. The pairs of public key and private key are stored in the cryptographic API.

6.2.2 Sharing the images

After succeeding registration and login, user who wants to share the images with restricted people needs the public keys of authorised users to encrypt the key which is used to encrypt the image and

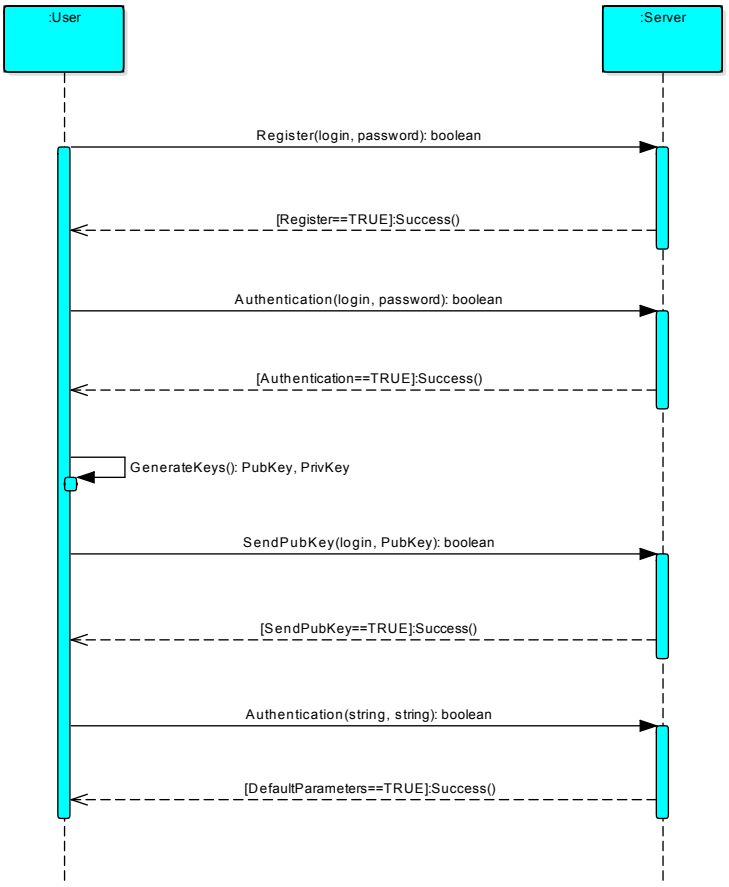


Figure 6.8: The general case of registration

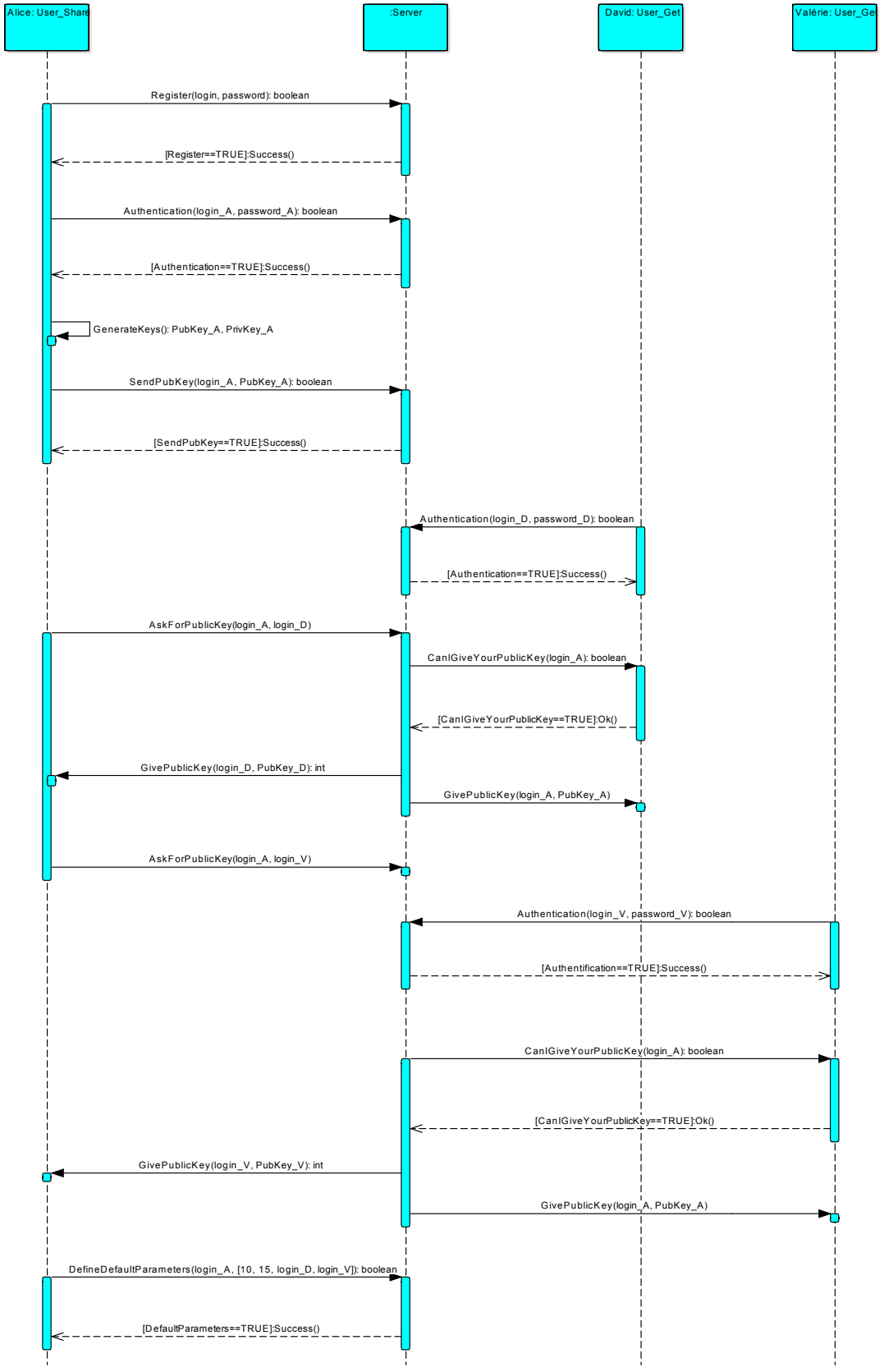


Figure 6.9: The registration of restricted users

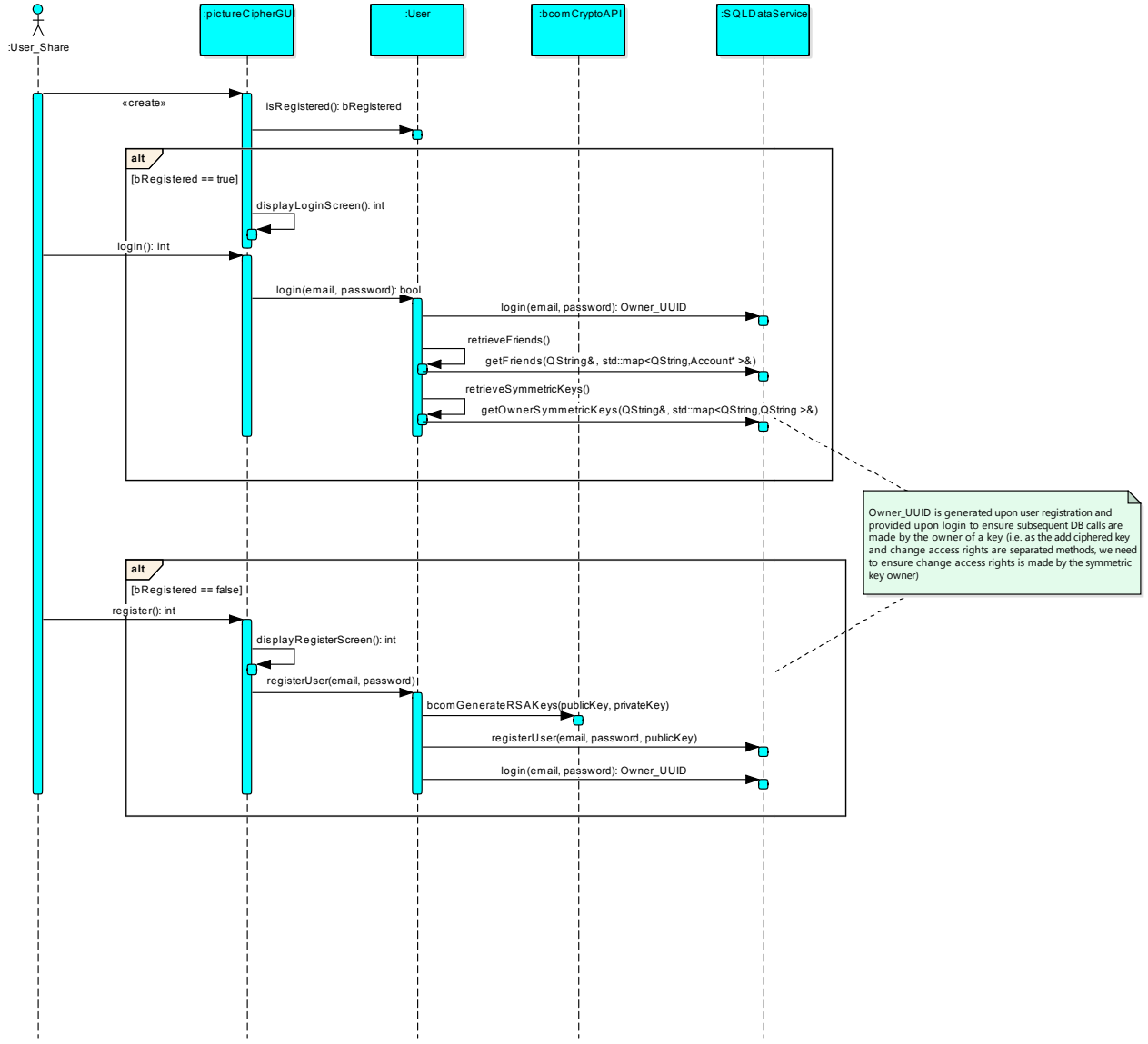


Figure 6.10: The registration in our server

store the encrypted keys in the server. He also needs to define the access conditions of his images and send them to the server. Then the user publishes his encrypted image on Facebook. On the other side, for the user who wants to see this image, our plugin in the browser verifies his rights with the server. After verifying the access conditions, the authorised user can get the encrypted key, he uses his private key to decrypt it and then decrypts the image, The unauthorised user can only get an error message. The access conditions can be modified at anytime by the user sharing image (Shown in Fig. 6.11).

The interaction of users sharing image in our application is shown in Fig. 6.12. The image ID and access conditions are stored in our database. All the keys are stored in the cryptographic API.

6.2.3 The secure in the server

In the process of sharing images, our server is trustworthy to the user. The key used to encrypt the image is encrypted by public key of each authorised user before being stored in the server. These keys are only decrypted by the corresponding private keys of authorised users. Therefore, user sharing images can trust the server will not obtain the decipher key and know the images he wants to share, even all the encrypted keys are stored in it.

The encrypted image is published on Facebook by user himself through his own computer. The authorised user on the other side only uses the plugin in the browser to connect the server. After getting the encrypted key from the server, the plugin helps user to decrypt the key and then to decrypt the image without storing the key. The decrypted image is only displayed on this user's computer, the server cannot know any information about the plaintext image.

6.2.4 Demonstration

In this section, we introduce how to use this application and plugin to share images on Facebook and we show some screenshots of the demonstration.

First of all, the users register themselves in our application (shown in Fig. 6.13(a)), they type all the asked information, if any information does not meet the requirement, an error message then is given back (shown in Fig. 6.13(b)).

Next, user types the username and password to login the application (shown in Fig. 6.14(a)), but if any of them is wrong, an error message is given back (shown in Fig. 6.14(b))

When user logins successfully, he can see the main interface(Fig. 6.16(a)), where he can drop his image, and he can check all the images he dropped (Fig. 6.16).

In the "parameters" interface (Fig. 6.16(a)), user can set the default information such as the duration of image validation, the number of view and shared contacts. For each image, in addition to use the default parameters, user can also select other authorised users and set other access conditions (Fig. 6.16(b)). Then user obtains the encrypted image and publishes it on Facebook. In the "contacts rights" interface (Fig. 6.16(c)), user can review the rights he assigns to each image. It is worth mentioning that, the identity of image is stored in the "QR code" which is embedded in the middle of the encrypted image. It is used to realise the connection between image and the server.

After publishing the image on Facebook, users can use the plugin in browser Chrome to decrypt the image (Fig. 6.17(a)). However, only the authorised users can successfully see the plaintext image in the validity date (Fig. 6.17(b)), while the others will be told that they do not have the right to see the image (Fig. 6.17(c)).

6.3 Studies of using PixGuardian

The PixGuardian is designed specifically to meet the need of authenticity control and data diffusion. In order to know if it is adapted to certain level of requirements of the users, we established an experiment to evaluate the user experiences of PixGuardian. The main object of this evaluation is to understand users' needs and to evaluate the conformity of conception solution to users' requirements. The improvement ideas proposed during the evaluation will be used to realise a more satisfying solution to meet users' needs and will be designed in the next version.

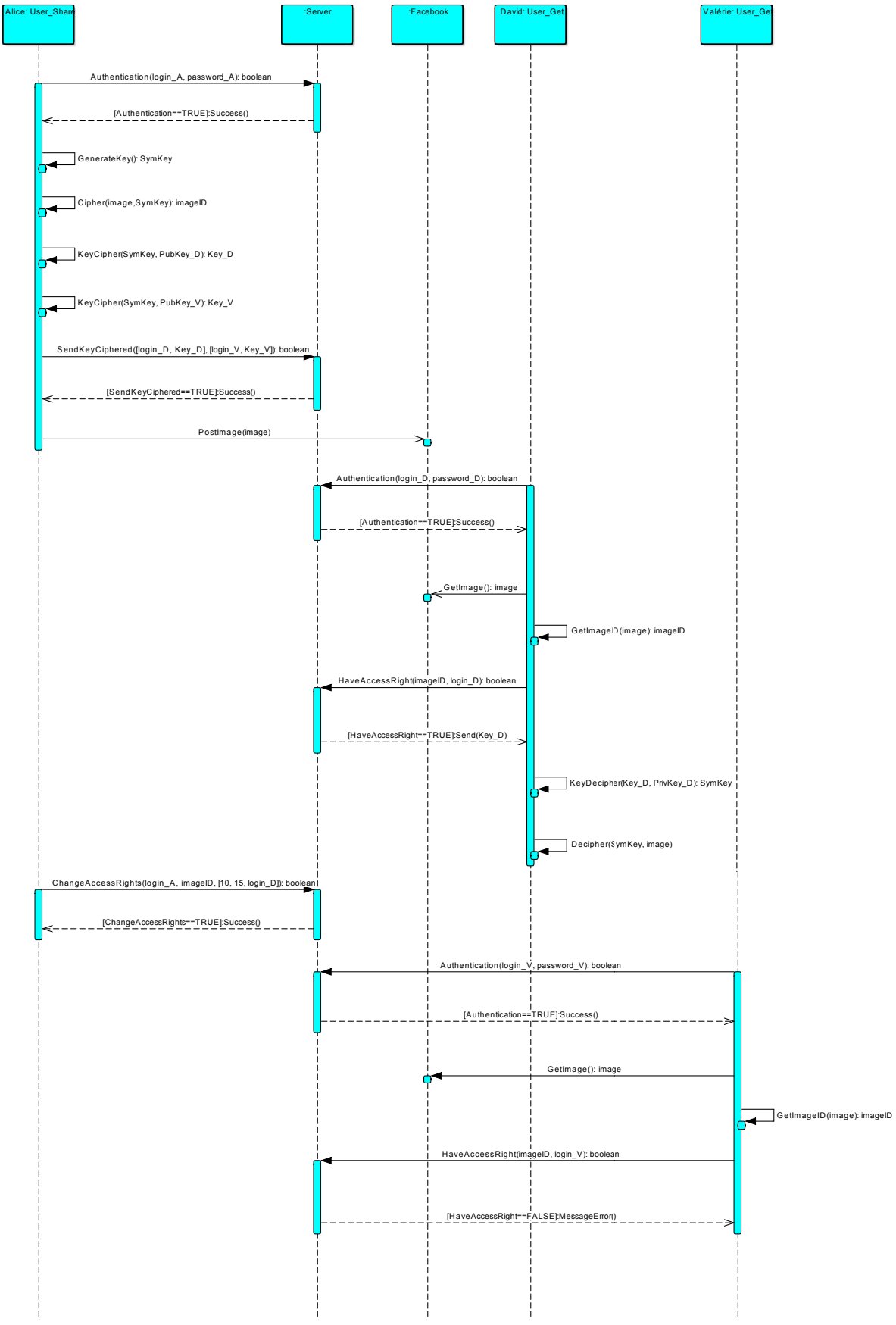


Figure 6.11: The sharing images between restricted users

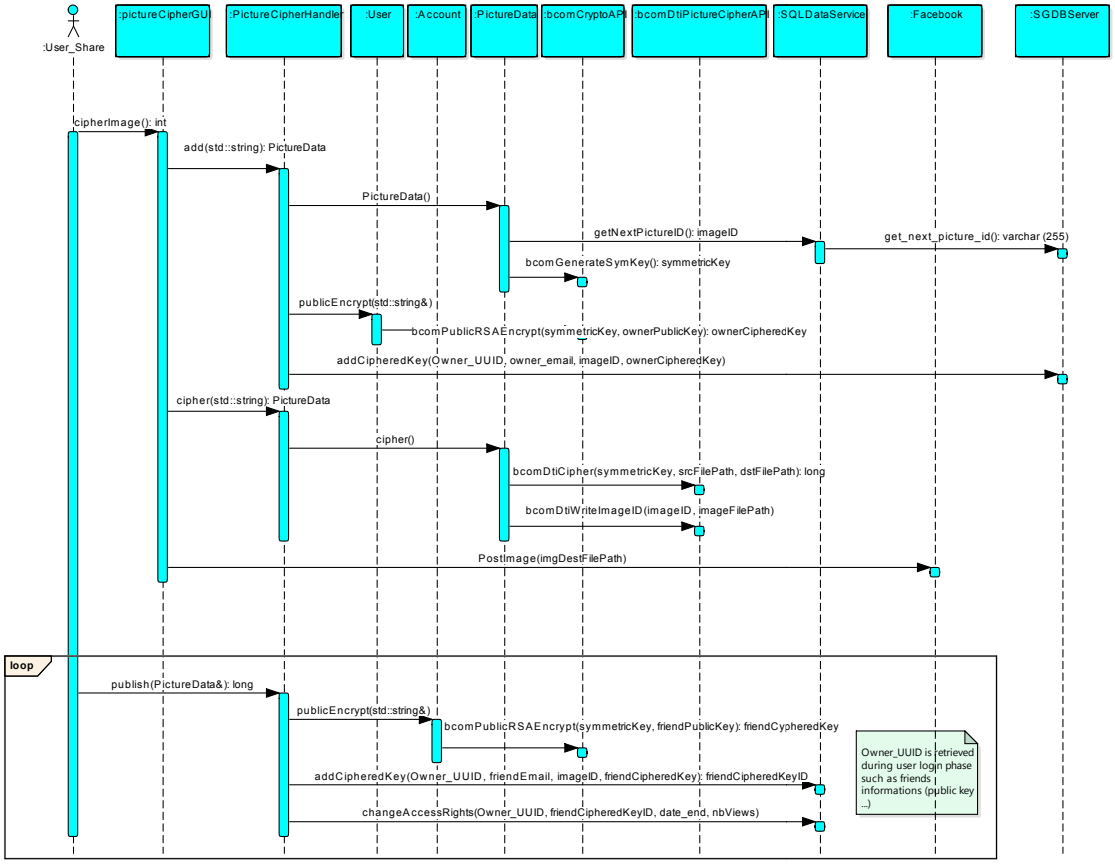
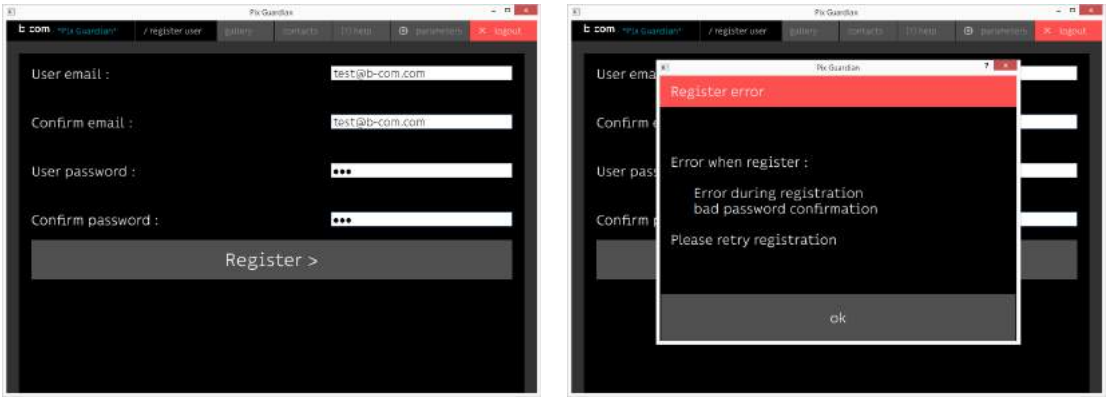


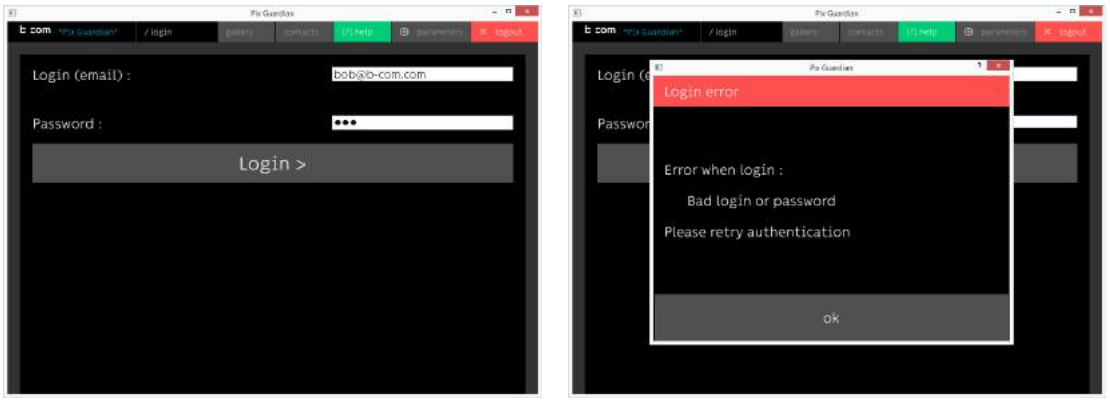
Figure 6.12: The sharing images in our server



(a) The registration interface

(b) The error message of registration interface

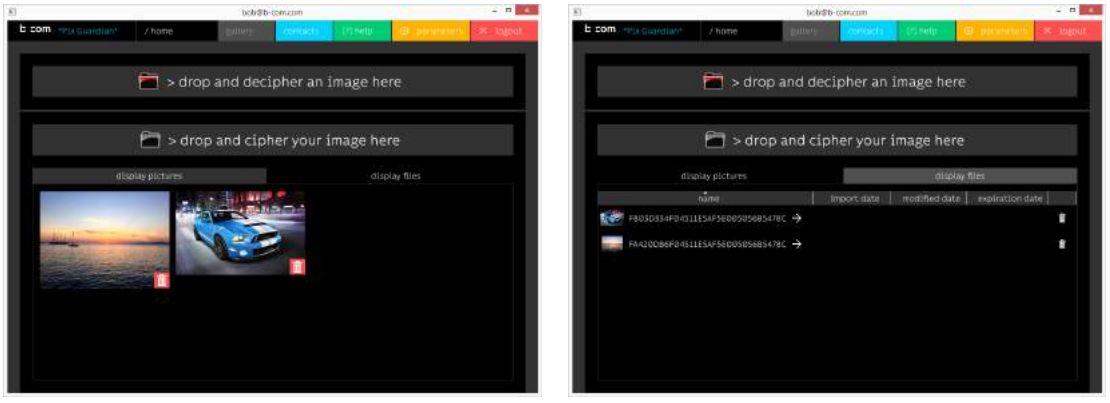
Figure 6.13: The process of registration



(a) The login interface

(b) The error message of login interface

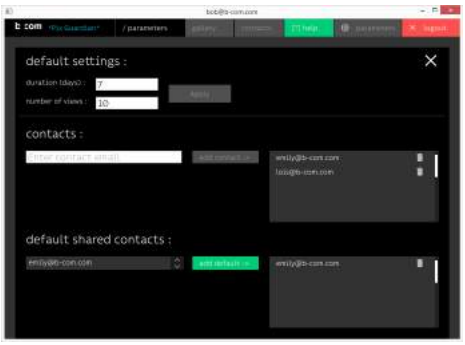
Figure 6.14: The process of login



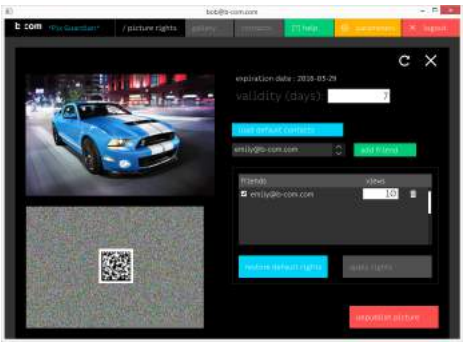
(a) The main interface (show images)

(b) The main interface (show files)

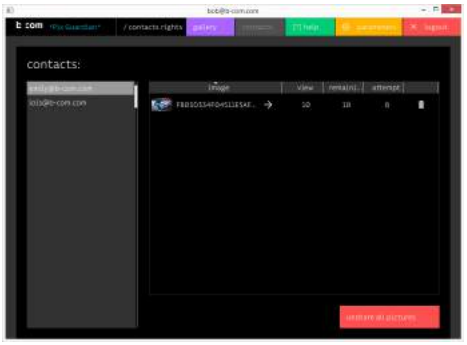
Figure 6.15: The main interface of application



(a) The interface to set default parameters



(b) The interface to set parameters for each image



(c) The interface to review the setting rights

Figure 6.16: Interfaces of PixGuardian

(a) The plugin option for the image

(b) The authorised result

(c) The unauthorised result

Figure 6.17: The plugin decrypting image on Facebook

We recruited 33 users participate this experiment during six weeks to collect the quantitative and qualitative data. The questionnaires submitted by users can help us to evaluate the acceptability and their experiences according to their positive and negative feedback.

6.3.1 Experiment description

In this evaluation experiment, users are placed in their natural environment, precisely which is expected by the system studied: they are at home, with their materials and use the system when they need.

This evaluation experiment is use to meet several objectives:

- **study the experiences** in a large, natural contexte. The context of use is a factor that impacts user experience. We want to collect real user experience (UX) instead of the test in a laboratory, which can reduce the limit of laboratory environment.
- **Study UX and its temporal dynamics.** With the questionnaire, we want to collect the data about various phases of the experiment. Before the use, we evaluate the anticipated UX (the users imagine), and after the use, we evaluate the UX momentary (their feelings during using our application).
- **Determine correlates and consequences of experience with the system.** Through interviews with users, we want to deepen the questionnaires and collect their expectations, their mood, or their use background.

The inconvenience is that the distance between the users and the observers makes it more difficult to record the observations and the validity of the analysis. These elements should be payed special attention during the study.

6.3.1.1 Statistics of uses

Anonymous statistics on the use and behaviour of participants is recorded when using the service. Among the traces of uses, we focus in particular on:

- The information of login account (frequency, success, fail).
- The used features (encryption, decryption, adding image, setting rights, etc.).
- All the manipulations of service that led to failures (connection, adding friends, decryption, etc.).

These traces help us to obtain an global view on the functional qualities of service, and also on its ergonomic qualities (usefulness, usability, efficiency). They decide what should we do during the interview. And they are also allow us to obtain the results on different computer configurations of participants.

6.3.1.2 Questionnaires

In this experiment, there are two parts of the questionnaires: the evaluation before use and the evaluation after use.

Questionnaire pre-use

The first part of the questionnaires lets the users understand a description of the service which allows them to imagine an use situation. Their answers reveal their practices, attitudes, values. It is an anticipated UX to be evaluated, and it includes the following dimensions:

- The acceptability is evaluated by: the global judgement and the intends of use.
- The affective-moticational factors that are measured by: intrinsic motivation and social influence.
- The anticipated UX which is measured by:

* The utility and usability as a measure of functional qualities for UX.
* The simulation, the global attractiveness, the aesthetics and the confidence as a scale of nonfunctional qualities for UX.
* The absorption, Self image and the emotion are used as an indicator of the affective-motivational reactions.

Questionnaire post-use

For the questionnaire post-use, the acceptability is evaluated by the global judgement and the intention of use. It is also measured by the factors and affective-motivational reactions in the following dimensions:

- Intrinsic motivation
- Social influence
- Self-image
- Their emotions

Effective UX is measured by:

- The utility and usability as a measure of functional qualities.
- The simulation, the global attractiveness, the aesthetics and the confidence as a scale of nonfunctional qualities.

These two parts questionnaires are managed by email during15 days and there are 35 and 57 questions respectively. All the questions are in the form of affirmative sentences. Participants will be asked to evaluate which degree they agree or disagree, by positioning spontaneously on an 11-point scale (from 0 to 10): 0 corresponds to "Strongly disagree" and 10 corresponds to "strongly agree"

6.3.1.3 Interviews

The questionnaire data and traces of use do not allow us to have a deep understanding of the users' activity, and of the context of these activities. Therefore, interviews are in progress in addition. The dialogue with potential users allow us to deeply explore the attitudes, opinions, preferences, beliefs, or mental representations of the interviewee.

In the interview, three main themes are addressed:

- The individual profile of user: we seek to know who they are, what are their technological hobbies, their habits on Facebook in terms of image sharing.
- The customs and attitudes of the users with PixGuardian: we focus on their experience with the service.
- Users' expectations: we ask if the service meets their global needs and what aspects should be improved in priority.

We also discuss the business model of the application: it determines if users are willing to pay for this service and in which form.

These points are addressed in the form of conversation, and the participants are not necessarily all geographically close. The interviews may also be conducted by videoconference (Skype, Hangouts).

6.3.1.4 Recruitment of participants

The target participants for this service are the general public, especially for the people who are interested in control their digital images on the Internet, and concerned with the right to be forgotten about their contents.

The current version of the application is intended for use on Facebook. The recruitment of participants in the experiment is taken into account as following:

- M/F 18 years old or more.
- Having a computer with Windows systems (XP, 7, 8, 10) or Mac OS.
- Having an active Facebook account (i.e. which is used at least once a week).
- Having shared, at least once, the images through Facebook.

The principal use case is sharing images with family/friends, the sampling technique called "snowball" (Shown in Fig. 6.18). We select 10 participants whom we can easily access as the internal ones. These people then be responsible for recruiting based on the same criteria. Then 10 people of their close entourage test the service. The internal participant has a close relationship with 10 external participants and these 10 external participants can also have the links between them.

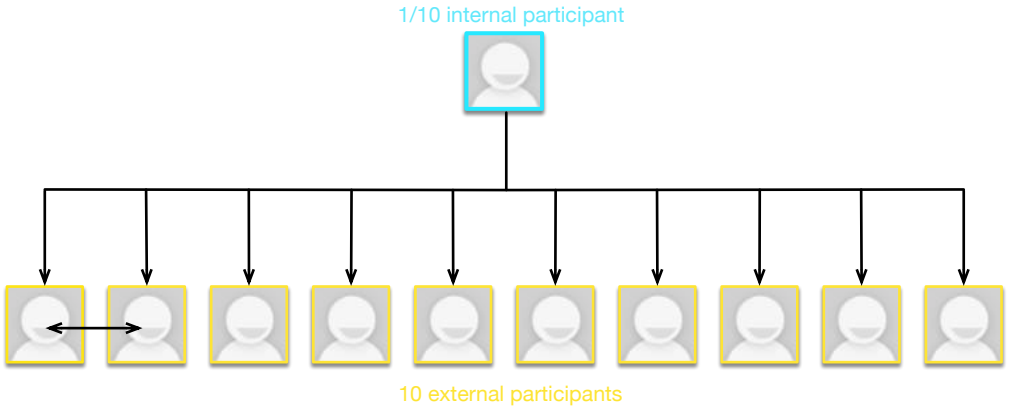


Figure 6.18: The interface to review the setting rights

6.3.2 Results and analyses

6.3.2.1 Participants

We recruited 33 participants, among them:

- 3 users do not meet the criteria (1 participant uses Linux, 1 user does not have a Facebook account, 1 user does not have a personal computer). Therefore, they were excluded from the base of testers.
- According to the statistics of uses, only 11 of them used the application at least once.
- 20 users responded the pre-use questionnaire, and 5 users responded to the post-use questionnaire.

These reduces the amount of data to analyse.

6.3.2.2 User Experience

Overall, the use of PixGuardian generates a positive experience for four-fifths of the people interviewed. The participants think the idea is interesting and the service is useful.

On the contrary, users do not plan to continue using it, since the service is considered too limited and does not seem finished. However, the participants perceive the potential of the service, and therefore they have some expectations of this service.

The experience global is average. They found that the service is useful for a professional use, but it was complicated, took a long time to implement, that demotivates.

The functional qualities

☹ The results about the **usability** of the service are weak. For most participants, it is complicated to use PixGuardia: it is relatively long and difficult to understand how to use. But with the description of the service, users had thought it is a simple service to use. Some users they it did not reach the goals we had set, with effectiveness, efficiency and satisfaction.

😐 There is also a decrease in the evaluation of the **utility** after use of the service. Users thought that PixGuardian only meet their needs moderately, and the functionalities offered by the service were not very important, so only not bad to use.

Non-instrumental qualities

☺ The answers about the factor **stimulation** are very positive. Users think that this service is original and innovative. The proposed functionalities are innovative and are not common. The service stimulates the curiosity of users, and allows them to learn new things.

😐 There is a decrease in the evaluation of the **global attractiveness** of the service. Using PixGuardian seems moderately pleasant, comfortable, and friendly. After using, the service becomes less attractive and less interesting for users and their friends.

😐 The results for the **aesthetics** of the application are heterogeneous. Users generally like the visual appearance of the application, however it has not particularly pleasant look, nor aesthetically attractive.

The affective-motivational reactions

☹ We notice that the effect of **absorption** is weak. Users are never truly absorbed by the interaction of the service. They do not lose the sense of time, nor forget anything else when using PixGuardian.

😐 The evaluation of the **confidence** of the service is average. For the question "I have the confidence in the information provided by this application", a majority of users tend to answer "disagree". Therefore, the information and functionalities provided by PixGuardian seem moderately week.

☹ The results about the **identification** and self-image are weak. Users feel they did not perceive positively or negatively for the others when they use the service. Using PixGuardian seems a little valuable, and users think their contacts do not envy to user the service.

😐 Majority of users have **positive emotion** during the interaction. They indicated that they are excited, happy and satisfied moderately. Only 3 users have **negative emotions** to use the service, they indicated that they felt annoyed and frustrated when using the application.

6.3.2.3 Acceptability of service

😐 The results related to the **intrinsic motivation** remain neutral. Using the service generates the amusement, satisfies the curiosity of users to discover new things, and gives pleasure when interacting, but only in a moderately way.

☹ Comparing the responses related to **social influence** pre-use and post-use. The latter one is negative. Users indicated that they would not encourage their relatives to use the service. And they do not think that people will be grateful to convince them to use the service.

😐 After use, there is also a decrease in the **global jugement** of the service. For the question 'How to evaluate PixGuardian globally". User gave a note of 610 (notice that the pre-use note is 810).

☺ Contrarily, **the intentions of use** increase. Users indicated particularly that they want to continue using the service in their daily lives.

6.3.2.4 User Expectations

The improvements and the evolutions

In terms of improvement, users are waiting for more simplicity in the implementation of the service. They believe that for the first use, a video or a animated tutorial is need to accompany the user to execute the necessary initial configurations.

They also want to make it easier to create an account and to manage the contacts. Two users suggest connecting with the Facebook ID, which could then help to identify friends who use the service. Considering the management of contacts, users think it should be expanded, such as the personalisation of contacts, automatic detection of a new user via importing a list of mail addresses, creation of groups.

In terms of evolution, they express the desire of a multi-support service. Indeed, the shared images are mostly from their smartphone, and users do not want to transfer the images to their computer before sharing them.

Users also desire an encryption tool for multi-format images (.png, .bmp, .tiff, .gif, etc.). The JPEG format are well prepared for the personal photos, but the illustrations are always in PNG.

An extension on the other web browsers is also desired. Indeed, among the 5 interviewers, only one of them use the Chrome browser.

Finally, users want a service that would allow them to know about: “Who saw the shared image?”, “Who stored it?”, “Who re-shared it, and where?”.

Business model

The participants all said they would not use the existing service if it has to pay. For the use on the social network, PixGuardian does not seem to bring a real additional value compared to the existing features, such as setting the audience.

However, for the business use, a participant is willing to acquire it after some evolution of service. He mentioned that the service should apply for different image formats, and more simplicity to implement for the clients. He insisted that the clients do not need the encryption interface, but just plugin decryption. So they want something easy to install. This participant imagined a license can be acquired 10 to 15 euros per year.

6.3.3 Conclusion

After studies of using PixGuardian, the results indicate that the users have a strong interest in the service, however, the current version does not meet all their needs. Nevertheless, users still foresee the potential of the service and wait the development, which contributes to maintain their strong intentions to use the service.

The improvements on pragmatic level of the main service interface would be needed (guiding, preventing and handling errors, feedback, etc.). The evolutions of some new functions are also expected. So in the next version of service, we need to consider these aspects to create a better user experience and a more positive acceptability.

Chapter 7

Conclusion and remarks on future research

Contents

7.1 Conclusion of the thesis

In recent years, with the rapid development of social networks, more and more users prefer to share their photos, videos, status, etc. through these platforms than using e-mail or personal webpages. It makes sharing become easier, and most of the social networks provider try to help users to control the access to their contents to protect their privacy, most of the social networks allow users to group their friends according to the relations they have with, such as "Friends", "Family", "Acquaintances", etc. Thus, specifying the friends that can access to each content, users may think that their privacy is guaranteed. However, from our point of view, the privacy of the users is not guaranteed with respect to the providers of the social networks since they can clearly know any published contents. Therefore, in this thesis, we aim to provide a method to protect user's privacy completely, that only the user himself and the authorised friends can know the contents.

According to some existing solutions and some analysis, in this thesis, we limited our study to JPEG images, and we assert that the published images should not be readable while uploading. Therefore, the main idea is to allow users to encrypt their images, and publish the encrypted images on any existing image-sharing platforms (i.e., Facebook, Flickr, Twitter, Google+, ...). The encryption keys as well as the access control rules of encrypted images are stored in our independent infrastructure. When someone accesses some published encrypted image, he/she has to request the encryption key to our independent infrastructure, the encryption key being sent back only if the access control rules authorise it. Notice that the enforcement of the access control rules can also be used to implement the right to be forgotten.

After studying the related works, we found there are three challenges when researchers propose the image encryption algorithm.

- The first challenge is the definition of an encryption algorithm that preserves the image format. Many image formats are accepted by existing image-sharing platforms, such as JPEG, PNG, GIF, etc. However, according to our experimentation (mentioned in §4.1), in most of platforms, the uploaded images are systematically compressed in JPEG format, no matter what are their original formats. In this context, the proposed encryption algorithm has to preserve the JPEG format [1].

[1] Notice that JPEG is the most widely used standards for storing and compressing images.

- The second challenge is that the encryption algorithm must be secure. Especially, we require that the adversary should not get any information of the plaintext image from the encrypted image. In particular, he/she cannot distinguish the encryption of a brighter image from a darker image. Therefore, IND- CPA secure is our judgment criteria for the security of image encryption algorithm.

- The third challenge is that the encryption algorithm has to be compatible with the majority of existing image-sharing platforms (i.e., Facebook, Flickr, Twitter, Google+, ...). However, according to our experimentation (mentioned in §4.1), some of the existing sharing platforms performs a post-processing of the published images. Thus, the proposed encryption algorithm has to allow to decrypt the encrypted image even if it has been post-processed.

In this thesis, we tried to overcome all these three challenges. Our first contribution is the definition of an encryption algorithm that preserves the JPEG format and that is IND-CPA secure. More specifically, we have proposed three JPEG image encryption schemes that use the same encryption algorithm but are integrated into different steps of JPEG compression process. We have proved the security of these encryption schemes and the comparison of the decrypted images allowed us to select the best scheme with respect to the quality of the resulted images.

Our second contribution is that all of the encrypted images are accepted as JPEG images by any image-sharing platform. We have used our encryption algorithm to encrypt JPEG images and we have uploaded the encrypted images on eight widely used image-sharing platforms (Facebook, Flickr, Pinterest, Google+, Twitter, Instagram and two Chinese image-sharing platforms: Weibo and Wechat). Thus, we have checked that all of the encrypted images have been accepted by these image-sharing platforms as having a correct image format.

Our third contribution is that our encryption scheme is compatible with the most of images-sharing platforms. We have downloaded and decrypted the images, and we have compared the decrypted images with the original ones. We have thus observed that for Flickr, Pinterest, Google+ and Twitter, the recovered images had a high quality. On the opposite, for Facebook, Instagram, Weibo and Wechat, we have noticed that the recovered images had an extremely poor quality. By analysing the results, we have discovered that these image-sharing platforms performs a post-processing for any uploaded images. Given that knowledge, we have been able to improve the encryption algorithm so as to obtain a new encryption scheme that is compatible with Facebook, Weibo and Wechat.

Our fourth contribution is an independent content management infrastructure that allows users to use our encryption algorithm to publish their images on Facebook while specifying the access control rules for these images. We have developed an application named PixGuardian to help the users to encrypt images. It also allows to generate, distribute and manage the encryption keys for the users, and permits users to define the access conditions for their images. Then, with our plugin in browser Chrome, the authorised users can decrypt images. Finally, we have evaluated the acceptability of PixGuardian by studying its use by 33 users during six weeks.

7.2 The future work

Before proposing the algorithm, in order to choose the most appropriate parameters for different image-sharing platforms, we did some tests on several widely used image-sharing platforms and obtained their characteristics. But we found that some researches gave the different test results about the characteristics of some image-sharing platforms [MPBS15, GPMB16]. Their tests showed that in some special case, the characteristics maybe not the same as the ones we found, but in normal case they are the same. It is probably because we used the different devices and did the tests in different periods of time (the tests probably were carried out on different version of platforms). We may need to improve our analyses results combining the other different research results. However, in our algorithm, before we upload the encrypted image, we prepare the images to change their characteristics to the fixed one, and these characteristics are proved will not be changed after being uploaded. So even if the other researches have the different analyse results, our algorithm will not be influenced and the experimental results will not be changed.

In our improved algorithm, for some image-sharing platforms, we need to prepare different parameters for each of them, since these parameters depend on the characteristics of each platforms, and each platform has different characteristics. This is a drawback of our algorithm which is not general for all the image-sharing platforms, although it can reconstruct the images with a high quality for most of platforms. Therefore, we want to continue to improve the algorithm to make it more general. For example, we can prepare parameters for a small quantization ratio, and for all the bigger quantization ratio, the parameters can be used but maybe lose some quality. In this way, we can use our application to provide more image-sharing platforms without changing the parameters.

We notice that, our algorithm can protect user's privacy among the authorised contacts. By using our application, users can define the access conditions, and their friends can use our plugin to access the plaintext images on Facebook in the browser Chromo directly. However, we cannot trust the authorised contacts completely. If they screenshots the plaintext images, and retransmit the images, we have no way to stop them and track them. For this problem, we may embed the watermark into the images, which help us to trace the user who retransmits the images and remove his/her authority immediately.

At the beginning, we aim to protect the content published on the social networks. According to the research on eMarketer [soc], we know the percentage the different types of content posted by Facebook pages worldwide. We propose to use encryption to protect user's privacy, and should make sure that the encrypted results can be published on the social network as a right format. Among these main types of contents posted, the images are more widely published on the social networks. Therefore, in this thesis, we propose how to protect the published images as a priority. In the next step, we will propose the algorithm which can protect the user's privacy when they publish the video on the social networks or some video-sharing platforms. Nowadays, many digital watermarking technologies are proposed to ensure that the copyright of the users who publish the video is not infringed. But even if the copyright of the users is protected, the privacy cannot be protected since the published contents can be always seen. The algorithm used to protect the video, it should adapt different format of video, different movie resolution, etc., especially the lossy compression and commonly used video format such as MPEG, HEVC. We will test to embed the encryption in different steps of compression process either in spacial domain or in frequency domain, then get the best one. If necessary, we will add the correcting mechanisme as well during the decryption after downloading the encrypted video.

According to the feedback of the users when we evaluated the acceptability of PixGuardian, we collect some improvement ideas. One of them is that the users prefer to share images from their smartphone directly instead of transferring the images to their computer before sharing them. We think this proposition is really valuable, because people like to take pictures by using their smartphone, it is convenience to publish them directly. And most of image-sharing platforms already have their application version on the smartphone, even more, some of them only have the smartphone version to publish images (Instagram, Snapchat, ...). Therefore, find a solution to encrypt images and publish them directly via the smartphone can be a good topic. But we need to consider the screen size of different models of smartphone, do some tests to know how they compress the published image and the degree of cropping, then we can propose a suitable algorithm and even develop another application which can be used on smartphone for publishing the encrypted images on different image-sharing platforms.

Bibliography

[3GP16] 3GPP. Security related network functions. Technical Report 43.020, 3rd Generation Partnership Project (3GPP), 2016.

[AGA12] Shoaib Ansari, Neelesh Gupta, and Sudhir Agrawal. An image encryption approach using chaotic map in frequency domain. *International Journal of Emerging Technology and Advanced Engineering*, 2(8):287–91, 2012.

[ASY97] K.T. Alligood, T.D. Sauer, and J.A. Yorke. *Chaos: An Introduction to Dynamical Systems*. Chaos: An Introduction to Dynamical Systems. New York, NY, 1997.

[BGLL14] Michael Backes, Sebastian Gerling, Stefan Lorenz, and Stephan Lukas. X-pire 2.0: a user-controlled expiration date and copy protection mechanism. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1633–1640. ACM, 2014.

[BHG+12] Ames Bielenberg, Lara Helm, Anthony Gentilucci, Dan Stefanescu, and Honggang Zhang. The growth of diaspora-a decentralized online social network in the wild. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 13–18. IEEE, 2012.

[BSVD09] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52. ACM, 2009.

[BYJ08] Mohammad Ali Bani Younes and Aman Jantan. Image encryption using block based transformation algorithm. 2008.

[CMS09] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12), 2009.

[DC00] Philip P Dang and Paul M Chau. Image encryption for secure internet multimedia applications. *Consumer Electronics, IEEE Transactions on*, 46(3):395–403, 2000.

[Den82] Dorothy E Denning. Cryptography and data security. 1982.

[Dey12] Shuvashis Dey. Sd-aei: An advanced encryption technique for images. In *Digital Information Processing and Communications (ICDIPC), 2012 Second International Conference on*, pages 68–73. IEEE, 2012.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[DR99] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[Dwo] Morris Dworkin. Recommendation for block cipher modes of operation : Methods and techniques. Nist specifications.

[EH06] D. Eastlake and T. Hansen. US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634, RFC Editor, July 2006.

[FMBD08] Cyril Fonteneau, Jean Motsch, Marie Babel, and Olivier Déforges. A hierarchical selective encryption technique in a scalable image codec. In *International Conference in Communications*, pages 1–4, 2008.

[G+98] Independent JPEG Group et al. Libjpeg 6b. *URL http://www. ijg. org March*, 1998.

[GBA] M. Peeters G. Bertoni, J. Daemen and G. Van Assche. Keccak Specification. NIST Specifications.

[GC08] Tiegang Gao and Zengqiang Chen. Image encryption based on a new total shuffling algorithm. *Chaos, Solitons & Fractals*, 38(1):213–220, 2008.

[GHG05] Zhi-Hong Guan, Fangjun Huang, and Wenjie Guan. Chaos-based image encryption algorithm. *Physics Letters A*, 346(1):153–157, 2005.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, Usa*, pages 365–377, 1982.

[GPMB16] Oliver Giudice, Antonino Paratore, Marco Moltisanti, and Sebastiano Battiato. A classification engine for image ballistics of social data. *arXiv preprint arXiv:1610.06347*, 2016.

[ILL89] Russell Impagliazzo, Leonid A Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 12–24. ACM, 1989.

[KE14] Pavel Korshunov and Touradj Ebrahimi. Scrambling-based tool for secure protection of jpeg images. In *IEEE International Conference on Image Processing (ICIP)*, number EPFL-CONF-200907, 2014.

[Ker05] Douglas A Kerr. Chrominance subsampling in digital images. *The Pumpkin,(1), November*, 2005.

[KJK10] Muhammad Imran Khan, Varun Jeoti, and Muhammad Asif Khan. Perceptual encryption of jpeg compressed images using dct coefficients and splitting of dc coefficients into bitplanes. In *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*, pages 1–6. IEEE, 2010.

[KR] Thomas Kunkelmann and Rolf Reinema. A scalable security architecture for multimedia communication standards. In *Proceedings of the International Conference on Multimedia Computing and Systems, ICMCS 1997, Ottawa, Ontario, Canada, June 3-6, 1997*, pages 660–662. IEEE Computer Society.

[KSHR10] Seyed Hossein Kamali, Reza Shakerian, Maysam Hedayati, and Mohsen Rahmani. A new modified version of advanced encryption standard based algorithm for image encryption. In *Electronics and Information Engineering (ICEIE), 2010 International Conference On*, volume 1, pages V1–141. IEEE, 2010.

[LDL12] Yuling Luo, Minghui Du, and Dong Liu. Jpeg image encryption algorithm based on spatiotemporal chaos. In *Chaos-Fractals Theories and Applications (IWCFTA), 2012 Fifth International Workshop on*, pages 191–195. IEEE, 2012.

[MO14] Antonio Marcedone and Claudio Orlandi. Obfuscation ⇒ (IND-CPA security !⇒ circular security). In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2014.

[MPBS15] Marco Moltisanti, Antonino Paratore, Sebastiano Battiato, and Luigi Saravo. Image manipulation on facebook for forensics evidence. In *International Conference on Image Analysis and Processing*, pages 506–517. Springer, 2015.

[PM93] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1993.

[PP12] Prashan Premaratne and Malin Premaratne. Key-based scrambling for secure image communication. In *Emerging Intelligent Computing Technology and Applications*, pages 259–263. Springer, 2012.

[PR+05] William Puech, Jose Marconi Rodrigues, et al. Crypto-compression of medical images by selective encryption of dct. In *EUSIPCO'05: European Signal Processing Conference*, 2005.

[PSU02] Martina Podesser, Hans-Peter Schmidt, and Andreas Uhl. Selective bitplane encryption for secure transmission of image data in mobile environments. In *Proceedings of the 5th IEEE Nordic Signal Processing Symposium (NORSIG?02)*, pages 4–6, 2002.

[PU02] Andreas Pommer and Andreas Uhl. Selective encryption of wavelet packet subband structures for obscured transmission of visual data. In *Proceedings of the 3rd IEEE Benelux Signal Processing Symposium (SPS 2002)*, pages 25–28, 2002.

[QN98] Lintian Qiao and Klara Nahrstedt. Comparison of mpeg encryption algorithms. *Computers & Graphics*, 22(4):437–448, 1998.

[RGO13] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward privacy-preserving photo sharing. In *NSDI*, pages 515–528, 2013.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sch96] Bruce Schneier. *Applied cryptography - protocols, algorithms, and source code in C, 2nd Edition*. Wiley, 1996.

[SD12] Rajesh Sharma and Anwitaman Datta. Supernova: Super-peers based architecture for decentralized online social networks. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–10. IEEE, 2012.

[Sha49] Claude E Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[soc] http://www.socialmediaexaminer.com/photos-generate-engagement-research/.

[Sta74] Fred Alan Stahl. On computational security. *On Computational Security*, 1974.

[Str01] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology and chemistry*. Perseus publishing, 2001.

[SZS04] Yang Shuangyuan, Lu Zhengding, and Han Shuihua. An asymmetric image encryption based on matrix transformation. In *Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on*, volume 1, pages 66–69. IEEE, 2004.

[TSBS13] Matt Tierney, Ian Spiro, Christoph Bregler, and Lakshminarayanan Subramanian. Cryptagram: Photo privacy for online social media. In *Proceedings of the first ACM conference on Online social networks*, pages 75–88. ACM, 2013.

[UJN+07] Koredianto Usman, Hiroshi Juzoji, I Nakajima, Soegijardjo Soegidjoko, Mohamad Ramdhani, Toshihiro Hori, and S Igi. Medical image encryption based on pixel arrangement and random permutation for transmission security. In *e-Health Networking, Application and Services, 2007 9th International Conference on*, pages 244–247. IEEE, 2007.

[VDB02] Marc Van Droogenbroeck and Raphaël Benedett. Techniques for a selective encryption of uncompressed and compressed images. *ACIVS Advanced Concepts for Intelligent Vision Systems, Proceedings*, 2002.

[WB02] Zhou Wang and Alan C Bovik. A universal image quality index. *Signal Processing Letters, IEEE*, 9(3):81–84, 2002.

[WFL15] Charles V Wright, Wu-chi Feng, and Feng Liu. Thumbnail preserving encryption for jpeg. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, pages 141–146. ACM, 2015.

[WWLC11] Yong Wang, Kwok-Wo Wong, Xiaofeng Liao, and Guanrong Chen. A new chaos-based fast image encryption algorithm. *Applied soft computing*, 11(1):514–522, 2011.

[YKE15] Lin Yuan, Pavel Korshunov, and Touradj Ebrahimi. Privacy-preserving photo sharing based on a secure JPEG. In *2015 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Hong Kong, China, April 26 - May 1, 2015*, pages 185–190. IEEE, 2015.

[YPWSp+09] Zhang Yun-Peng, Liu Wei, Cao Shui-ping, Zhai Zheng-jun, Nie Xuan, and Dai Weidi. Digital image encryption algorithm based on chaos and improved des. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 474–479. IEEE, 2009.

[ZMK+07] Medien Zeghid, Mohsen Machhout, Lazhar Khriji, Adel Baganne, Rached Tourki, et al. A modified aes based algorithm for image encryption. *International Journal of Computer Science and Engineering*, 1(1):70–75, 2007.

[ZZWY11] Zhi-liang Zhu, Wei Zhang, Kwok-wo Wong, and Hai Yu. A chaos-based symmetric image encryption scheme using a bit-level permutation. *Information Sciences*, 181(6):1171–1186, 2011.

Résumé

Au cours de ces dernières années, avec le développement rapide des plateformes de partage d'images, de plus en plus d'utilisateurs choisissent de diffuser leurs photos sur ces services notamment grâce à leur facilité d'utilisation. La plupart des plateformes permettent aux utilisateurs d'autoriser l'accès aux images uniquement à un groupe restreint de personnes. Une telle capacité donne un sentiment de confiance aux utilisateurs vis-à-vis de la confidentialité de ces images. Malheureusement, cela reste uniquement un sentiment d'illusion, car une fois que les utilisateurs ont mis en ligne leurs images, la confidentialité ne peut être garantie sachant que le fournisseur de la plateforme, qui n'est pas nécessairement une entité de confiance, peut connaitre clairement le contenu de n'importe quelle image publiée sur sa plateforme. Par contre, si les images ne sont plus mises en ligne en claire mais chiffrées, i.e. qu'elles ont été rendu illisibles par un algorithme de chiffrement adapté, alors la confidentialité est assurée. Ainsi, il sera fourni aux personnes autorisées à visionner ces images un moyen pour déchiffrer, et ainsi accéder à leurs contenus en clair.

En étudiant les différentes méthodes existantes pour chiffrer des images publiées sur des plateformes de partage d'images, nous en avons conclu qu'à notre connaissance, il y a trois principales spécificités à prendre en compte lors du chiffrement d'une image, qui restreigne le choix de l'algorithme de chiffrement. Tout d'abord, le chiffrement doit être effectué en respectant le format de l'image. En effet, une image chiffrée doit être aussi une image (e.g. le format JPEG), afin de pouvoir être mise en ligne sur la plateforme de partage. Deuxièmement, l'algorithme de chiffrement doit garantir l'indistinguabilité (propriété, IND-CPA) des messages chiffrés. En effet, l'adversaire ne doit obtenir de l'information sur le contenu de l'image à partir de la version chiffrée de l'image. Dans un troisième temps, l'algorithme doit être compatible avec les traitements des images spécifiques à la plateforme de partage d'images. Les éventuelles détériorations de la qualité de l'image liée à de tels traitements de l'image ne doivent pas pour autant empêcher le déchiffrement de l'image.

L'objectif principal de cette thèse a été de proposer un nouveau schéma de chiffrement des images JPEG permettant à la fois de garantir la conservation du format, la confidentialité du contenu et de la qualité de l'image JPEG. Dans un premier temps, nous avons proposé un schéma de chiffrement garantissant les deux premières problématiques. Ce schéma a ensuite été implémenté sur différentes plateformes de partage d'images (Facebook, Flickr, Pinterest, Google+, Twitter, Instagram, Weibo, Wechat). Malheureusement, nous avons montré que sur certaines plateformes, (Facebook, Instagram, Weibo and Wechat) notre solution ne permettait de maintenir une qualité d'images suffisantes après déchiffrement. Par conséquent, des codes correcteurs ont été ajoutés au sein de l'algorithme de chiffrement et déchiffrement, afin de maintenir la bonne qualité des images déchiffrées.

Mots-clés: Chiffrement d'images, Respect de la vie privée, Plateformes de partages d'images, Réseaux sociaux.

Abstract

In recent years, with the rapid development of image-sharing platforms, more and more users prefer to share their photos through these platforms than using e-mail or personal webpages, which makes image-sharing become easier. Most of the platforms allow the users to specify who can access to the images, it may result a feeling of safety and privacy. However, once users upload their images, privacy is not guaranteed, since at least the provider of the image-sharing platform can clearly know the contents of any published images. Therefore, uploading an unreadable image is a good solution to protect user's privacy. According to some existing researches, encrypting images before publishing them should be a top priority. In this way, only the authorised users who can decrypt the encrypted image can access to the contents of the published images.

By studying the existing methods, we found that there are three challenges when proposing an encryption algorithm for the images published on image-sharing platforms. Firstly the encrypted result has to be viewed as a correct image format (often a JPEG image) by the image-sharing platforms. Therefore, an encryption algorithm which preserves image format after encryption is required. Secondly, the algorithm should be secure, i.e. the adversary cannot get any information of plaintext image from the encrypted image. We aim to provide IND-CPA property for the encryption algorithm. Thirdly, the algorithm has to be compatible with basic processing of the digital data in each image-sharing platforms, that means the decryption algorithm can recover the plaintext image with a high quality even the encrypted image is processed by the platforms.

In this thesis, our main goal is to propose an encryption algorithm to protect JPEG image privacy on different image-sharing platforms and overcome the three challenges (image format preserving, security guarantee, processing compatibility). We first propose an encryption algorithm which can meet the requirements of the first two points. We then implement this algorithm on several widely used image-sharing platforms (Facebook, Flickr, Pinterest, Google+, Twitter, Instagram, Weibo, Wechat). However, the results show that it cannot recover the plaintext image with a high quality after downloading the image from some of the image-sharing platforms we tested (Facebook, Instagram, Weibo and Wechat). This encryption algorithm can be only used to protect the privacy on the other four platforms. Therefore, we improve this algorithm to add the correcting mechanism, in order to reduce the losses of image information during uploading the encrypted image on each platforms and reconstruct the downloaded images with a high quality.

Keywords: Image encryption, Privacy protection, image-sharing platforms, social networks.