REPUBLIQUE DU CAMEROUN
Paix - Travail – Patrie

UNIVERSITE DE YAOUNDE I

FACULTE DES SCIENCES

CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIE & GEOSCIENCES

UNITE DE RECHERCHE ET DE FORMATION DOCTORALE EN MATHEMATIQUES, INFORMATIQUES, BIOINFORMATIQUES ET APPLICATIONS
B.P. : 812 Yaoundé

REPUBLIC OF CAMEROON
Peace – Work – Fatherland

UNIVERSITY OF YAOUNDE I

FACULTY OF SCIENCES

POSTGRADUATE SCHOOL OF SCIENCE, TECHNOLOGY & GEOSCIENCES

RESEARCH AND TRAINING UNIT FOR DOCTORATE IN MATHEMATICS, COMPUTER SCIENCES AND APPLICATIONS
P.O. Box : 812 Yaoundé

LABORATOIRE D'INFORMATIQUE ET APPLICATIONS
LABORATORY OF COMPUTER SCIENCE AND APPLICATIONS

STEGANOGRAPHIE RESISTANTE A LA DETECTION DU SECRET PAR CONSERVATION DE L'INTEGRITE DES MEDIAS DE COUVERTURE

THESE

PRESENTEE ET SOUTENUE EN VUE DE L'OBTENTION DU GRADE DE

DOCTEUR/PhD EN INFORMATIQUE

PAR

MOYOU METCHEKA Léonel

MATRICULE : 06U236

SOUS LA DIRECTION DE

NDOUNDAM René

Maître de Conférences, Université de Yaoundé I

Année Académique 2023-2024

RÉPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
****
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
****
UNIVERSITÉ DE YAOUNDÉ I
****
CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TE*CHNOLOGIES* ET *GEOSCIENCES*
****

REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
****
MINISTRY OF HIGHER EDUCATION
****
THE UNIVERSITY OF YAOUNDE I
****
POSTGRADUATE SCHOOL OF SCIENCE, TECHNOLOGY AND GEOSCIENCES
****

DÉPARTEMENT D'INFORMATIQUE
DEPARTMENT OF COMPUTER SCIENCES

ATTESTATION DE CORRECTION DE LA THESE DE DOCTORAT / Ph.D

Nous soussignés, **TCHUENTE Maurice, Pr., UYI, Président du jury ; NDOUNDAM René, Pr., Rapporteur ; FOUDA DJODO Marcel, Pr., UYI, Examinateur ; YENKE Blaise Omer, Pr., UN, Examinateur ; TSOPZE Norbert, Pr., UYI, Examinateur,** membres du jury de la thèse de Doctorat / Ph.D présenté par **M. MOYOU METCHEKA Léonel, Matricule 06U236,** intitulée : **« Stéganographie résistante à la détection du secret par conservation de l'intégrité des médias de couverture »** et soutenue en vue de l'obtention du diplôme de Doctorat / Ph.D en **informatique,** Spécialité : **Sécurité informatique,** Option : **Stéganographie,** attestons que toutes les corrections demandées par le jury de soutenance en vue de l'amélioration de ce travail, ont été effectuées.

En foi de quoi la présente attestation lui est délivrée pour servir et valoir ce que de droit.

Président	**Rapporteur**	**Examinateurs**
Prof N. Tchuente	R. Ndoundam	Pr Yenke
		Tsopze N
		Fouda Djodo

Liste Protocolaire

Liste des enseignants permanents		
Année universitaire 2021-2022 (Par Département et par Grade)		
Date d'actualisation 22 Septembre 2021		
Administration		
DOYEN	OWONO OWONO Luc Calvin	Professeur
VICE-DOYEN DPSAA	ATCHADE A. T.	Maitre de Conférences
VICE-DOYEN DSSE	NYEGUE M. A.	Professeur
VICE-DOYEN DRC	ABOSSOLO Monique	Maitre de Conférences
Chef DAF	NDOYE FOE Marie C. F.	Maitre de Conférences
Chef DAASR	AJEAGAH G. A.	Professeur

1- DÉPARTEMENT DE BIOCHIMIE (BC) (40)		
NOMS ET PRÉNOMS	**GRADE**	**OBSERVATIONS**
1-BIGOGA DAIGA J.	Professeur	En poste
2-FEKAM BOYOM F.	Professeur	En poste
3-FOKOU Elie	Professeur	En poste
4-KANSCI Germain	Professeur	En poste
5-MBACHAM FON W.	Professeur	En poste
6-MOUNDIPA FEWOU P.	Professeur	Chef de Département
7-NINTCHOM P. V. épse BENG	Professeur	En poste
8-OBEN Julius E.	Professeur	En poste
9-ACHU Merci BIH	Maître de Conférences	En poste
10-ATOGHO Barbara Mma	Maître de Conférences	En poste
11-AZANTSA KINGUE G. B.	Maître de Conférences	En poste
12-BELINGA née NDOYE F. M. C. F.	Maître de Conférences	Chef DAF / FS
13-BOUDJEKO Thaddée	Maître de Conférences	En poste
14-DJUIDJE NGOUNOUE M.	Maître de Conférences	En poste
15-EFFA NNOMO Pierre	Maître de Conférences	En poste
16-EWANE Cécile Annie	Maître de Conférences	En poste
17-MOFOR née TEUGWA C.	Maître de Conférences	IS MINESUP
18-NANA L. épouse WAKAM	Maître de Conférences	En poste
19-NGONDI Judith L.	Maître de Conférences	En poste
20-NGUEFACK J.	Maître de Conférences	En poste
21-NJAYOU F. N.	Maître de Conférences	En poste
22-TCHANA KOUATCHOUA A.	Maître de Conférences	En poste
23-AKINDEH MBUH NJI	Chargé de Cours	En poste
24-BEBEE Fadimatou	Chargé de Cours	En poste
25-BEBOY EDJENGUELE S. N.	Chargé de Cours	En poste
26-DAKOLE DABOY C.	Chargé de Cours	En poste
27-DJUIKWO NKONGA R. V.	Chargée de Cours	En poste
28-DONGMO LEKAGNE J. B.	Chargé de Cours	En poste
29-FONKOUA M.	Chargé de Cours	En poste
30-KOTUE KAPTUE C.	Chargé de Cours	En poste
31-LUNGA Paul K.	Chargé de Cours	En poste
32-MANANGA M. J.	Chargée de Cours	En poste
33-MBONG ANGIE M. M. A.	Chargée de Cours	En poste
34-Palmer MASUMBE N.	Chargé de Cours	En poste
35-PECHANGOU NSANGOU S.	Chargé de Cours	En poste

36-FOUPOUAPOUOGNIGNI Y.	Assistant	En poste
37-KOUOH ELOMBO F.	Assistant	En poste
38-MBOUCHE FANMOE M. J.	Assistante	En poste
39-OWONA AYISSI V. B.	Assistant	En poste
40-WILFRIED A. A.	Assistante	En poste
2- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE ANIMALES (BPA) (51)		
1-AJEAGAH Gideon A.	Professeur	DAARS/FS
2-BILONG BILONG C. F.	Professeur	Chef de Département
3-DIMO Théophile	Professeur	En Poste
4-DJIETO LORDON C.	Professeur	En Poste
5-DZEUFIET DJOMENI P. D.	Professeur	En poste
6-ESSOMBA née NTSAMA M.	Professeur	VD/FMSB/UYI
7-FOMENA A.	Professeur	En Poste
8-KAMTCHOUING P.	Professeur	En poste
9-KEKEUNOU Sévilor	Maître de Conférences	En poste
10-NJAMEN D.	Professeur	En poste
11-NJIOKOU Flobert	Professeur	En Poste
12-NOLA Moïse	Professeur	En poste
13-TAN Paul V.	Professeur	En poste
14-TCHUEM TCHUENTE L. A.	Professeur	I. S. CP/MINSANTE
15-ZEBAZE TOGOUET S. H.	Professeur	En poste
16-BILANDA Danielle C.	Maître de Conférences	En poste
17-DJIOGUE Séfirin	Maître de Conférences	En poste
18-JATSA BOUKENG H. épse M.	Maître de Conférences	En Poste
19-LEKEUFACK FOLEFACK G. B.	Maître de Conférences	En poste
20-MEGNEKOU Rosette	Maître de Conférences	En poste
21-MONY R. épse NTONE	Maître de Conférences	En Poste
22-NGUEGUIM TSOFACK F.	Maître de Conférences	En poste
23-TOMBI Jeannette	Maître de Conférences	En poste
24-ALENE Désirée C.	Chargée de Cours	En poste
25-ATSAMO Albert D.	Chargé de Cours	En poste
26-BELLET EDIMO Oscar R.	Chargé de Cours	En poste
27-DONFACK M.	Chargée de Cours	En poste
28-ETEME ENAMA S.	Chargé de Cours	En poste
29-GOUNOUE KAMKUMO R.	Chargée de Cours	En poste
30-KANDEDA KAVAYE A.	Chargé de Cours	En poste
31-MAHOB Raymond J.	Chargé de Cours	En poste
32-MBENOUN MASSE P. S.	Chargé de Cours	En poste
33-MOUNGANG L. M.	Chargée de Cours	En poste
34-MVEYO NDANKEU Y. P.	Chargé de Cours	En poste
35-NGOUATEU KENFACK O. B.	Chargé de Cours	En poste
36-NGUEMBOK	Chargé de Cours	En poste
37NJUA Clarisse Yafi	Chargée de Cours	CD. UBA
38-NOAH EWOTI Olive V.	Chargée de Cours	En poste

39-TADU Zephyrin	Chargé de Cours	En poste
40-TAMSA ARFAO Antoine	Chargé de Cours	En poste
41-YEDE	Chargé de Cours	En poste
42-AMPON NSANGOU Indou	Assistant	En poste
43-BASSOCK BAYIHA E. D.	Assistant	En poste
44-ESSAMA MBIDA D. S.	Assistante	En poste
45-FEUGANG YOUMSSI F.	Assistant	En poste
46-FOKAM A. C. Epse KEGNE	Assistant	En poste
47-GONWOUO NONO L.	Assistant	En poste
48-KOGA MANG DOBARA	Assistant	En poste
49-LEME BANOCK Lucie	Assistante	En poste
50-NWANE Philippe Bienvenu	Assistante	En poste
51-YOUNOUSSA LAME	Assistant	En poste
3- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE VÉGÉTALES (BPV) (31)		
1-AMBANG Zachée	Professeur	CD/UYII
2-BELL Joseph M.	Professeur	En poste
3-DJOCGOUE Pierre F.	Professeur	En poste
4-MBOLO Marie	Professeur	En poste
5-MOSSEBO Dominique C.	Professeur	En poste
6-YOUMBI Emmanuel	Professeur	Chef de Département
7-ZAPFACK Louis	Professeur	En poste
8-ANGONI Hyacinthe	Maître de Conférences	En poste
9-BIYE Elvire H.	Maître de Conférences	En poste
10-MALA Armand W.	Maître de Conférences	En poste
11-MBARGA BINDZI Marie A.	Maître de Conférences	CT/ MINESUP
12-NDONGO BEKOLO	Maître de Conférences	CE / MINRESI
13-NGODO MELINGUI Jean B.	Maître de Conférences	En poste
14-NGONKEU MAGAPTCHE Eddy L.	Maître de Conférences	En poste
15-TONFACK Libert B.	Maître de Conférences	En poste
16-TSOATA Esaïe	Maître de Conférences	En poste
17-DJEUANI Astride C.	Chargé de Cours	En poste
18-GOMANDJE Christelle	Chargée de Cours	En poste
19-MAFFO MAFFO Nicole L.	Chargé de Cours	En poste
20-MAHBOU SOMO TOUKAM. G.	Chargé de Cours	En poste
21-NGALLE Hermine B.	Chargée de Cours	En poste
22-NNANGA MEBENGA Ruth L.	Chargé de Cours	En poste
23-NOUKEU KOUAKAM A.	Chargé de Cours	En poste
24-ONANA JEAN MICHEL	Chargé de Cours	En poste
25-GODSWILL NTSOMBAH N.	Assistant	En poste
26-KABELONG BANAHO L.P.R.	Assistant	En poste
27-KONO Léon Dieudonné	Assistant	En poste
28-LIBALAH Moses B.	Assistant	En poste
29-LIKENG-LI-NGUE Benoit C	Assistant	En poste
30-TAEDOUNG Evariste H.	Assistant	En poste
31-TEMEGNE NONO C.	Assistant	En poste

4- DÉPARTEMENT DE CHIMIE INORGANIQUE (CI) (32)		
1-AGWARA ONDOH M.	Professeur	Chef de Département
2-DJOUFAC WOUMFO E.	Maître de Conférences	En poste
3-Florence UFI C. épouse MELO	Professeur	R. U.Ngaoundere
4-GHOGOMU Paul M.	Professeur	M. C deMiss.PR
5-NANSEU NjikiCharles P.	Professeur	En poste
6-NDIFON Peter T.	Professeur	CT MINRESI
7-NDIKONTAR M. KOR	Professeur	VD Univ. Bamenda
8-NENWA Justin	Professeur	En poste
9-NGAMENI Emmanuel	Professeur	D FS UDs
10-NGOMO Horace M.	Professeur	V Chancelor/UB
11-ACAYANKA Elie	Maître de Conférences	En poste
12-EMADACK Alphonse	Maître de Conférences	En poste
13-KAMGANG YOUBI G.	Maître de Conférences	En poste
14-KEMMEGNE MBOUGUEM J. C.	Maître de Conférences	En poste
15-KONG SAKEO	Maître de Conférences	En poste
16-NDI NSAMI J.	Maître de Conférences	En poste
17-NJIOMOU C. épse D.	Maître de Conférences	En poste
18-NJOYA Dayirou	Maître de Conférences	En poste
19-TCHAKOUTE KOUAMO H.	Chargé de Cours	En poste
20-BELIBI BELIBI P. D.	Chargé de Cours	CS/ ENS Bertoua
21-CHEUMANI YONA A. M.	Chargé de Cours	En poste
22-KENNE DEDZO G.	Chargé de Cours	En poste
23-KOUOTOU DAOUDA	Chargé de Cours	En poste
24-MAKON Thomas B.	Chargé de Cours	En poste
25-MBEY Jean Aime	Chargé de Cours	En poste
26-NCHIMI NONO KATIA	Chargé de Cours	En poste
27-NEBA nee NDOSIRI B. N.	Chargée de Cours	CT/ MINFEM
28-NYAMEN Linda D.	Chargée de Cours	En poste
29-PABOUDAM GBAMBIE A.	Chargée de Cours	En poste
30-NJANKWA NJABONG N. E.	Assistant	En poste
31-PATOUOSSA ISSOFA	Assistant	En poste
32-SIEWE Jean Mermoz	Assistant	En Poste
5- DÉPARTEMENT DE CHIMIE ORGANIQUE (CO) (40)		
1-DONGO Etienne	Professeur	Vice-Doyen
2-GHOGOMU TIH Robert R.	Professeur	Dir. IBAF/UDA
3-NGOUELA Silvère A.	Professeur	Chef de Departement UDS
4-NYASSE Barthélemy	Professeur	En poste
5-PEGNYEMB Dieudonné E.	Professeur	D/ MINESUP
6-WANDJI Jean	Professeur	En poste
7-Alex de Théodore A.	Maître de Conférences	VD / DPSAA
8-AMBASSA Pantaléon	Chargé de Cours	En poste
9-EYONG Kenneth OBEN	Maître de Conférences	En poste

10-FOLEFOC Gabriel N.	Maître de Conférences	En poste
11-FOTSO WABO G.	Maître de Conférences	En poste
12-KEUMEDJIO Félix	Maître de Conférences	En poste
13-KEUMOGNE Marguerite	Maître de Conférences	En poste
14-KOUAM Jacques	Maître de Conférences	En poste
15-MBAZOA née DJAMA C.	Maître de Conférences	En poste
16-MKOUNGA Pierre	Maître de Conférences	En poste
17-MVOT AKAK CARINE	Chargé de Cours	En poste
18-NGO MBING Joséphine	Maître de Conférences	S/D. MINERESI
19-NGONO BIKOBO Dominique S.	Maître de Conférences	En poste
20-NOTE LOUGBOT Olivier P.	Maître de Conférences	CS/MINESUP
21-NOUNGOUE TCHAMO D.	Maître de Conférences	En poste
22-TABOPDA KUATE Turibio	Maître de Conférences	En poste
23-TAGATSING FOTSING M.	Maître de Conférences	D /FS/ UYI
24-TCHOUANKEU Jean-Claude	Maître de Conférences	D /FS/ UYI
25-TIH née NGO BILONG E. A.	Maître de Conférences	En poste
26-YANKEP Emmanuel	Maître de Conférences	En poste
27-ZONDENDEGOUMBA Ernestine	Chargée de Cours	En poste
28-KAMTO Eutrophe Le Doux	Chargé de Cours	En poste
29-NGNINTEDO Dominique	Chargé de Cours	En poste
30-NGOMO Orléans	Chargée de Cours	En poste
31-OUAHOUO WACHE B. M.	Chargée de Cours	En poste
32-SIELINOU TEDJON V.	Chargé de Cours	En poste
33-MESSI Angélique N.	Assistant	En poste
34-MUNVERA MFIFEN Aristide	Assistant	En poste
35-NONO NONO Éric C.	Assistant	En poste
36-OUETE NANTCHOUANG J. L.	Assistant	En poste
37-TCHAMGOUE Joseph	Assistant	En poste
38-TSAFFACK Maurice	Assistant	En poste
39-TSAMO TONTSA A.	Assistant	En poste
40-TSEMEUGNE Joseph	Assistant	En poste
6- DÉPARTEMENT D'INFORMATIQUE (IN) (22)		
1-FOUDA NDJODO Marcel L.	Professeur	CD ENS/C IGA.SUP
2-ATSA ETOUNDI Roger	Professeur	C Div.MINESUP
3-NDOUNDAM Réné	Maître de Conférences	En poste
4-TSOPZE Norbert	Maître de Conférences	En poste
5-ABESSOLO ALO'O Gislain	Chargé de Cours	En poste
6-AMINOU Halidou	Chargé de Cours	Chef de Département
7-DJAM Xaviera YOUH - KIMBI	Chargé de Cours	En Poste
8-DOMGA KOMGUEM Rodrigue	Chargé de Cours	En poste
9-EBELE Serge Alain	Chargé de Cours	En poste
10-EKODECK Stéphane Gaël R.	Chargé de Cours	En poste
11-HAMZA Adamou	Chargé de Cours	En poste
12-JIOMEKONG AZANZI F.	Chargé de Cours	En poste
13-KOUOKAM KOUOKAM E. A.	Chargé de Cours	En poste

14-MELATAGIA YONTA Paulin	Chargé de Cours	En poste
15-MESSI NGUELE Thomas	Chargé de Cours	En poste
16-MONTHE DJIADEU V. M.	Chargé de Cours	En poste
17-NZEKON NZEKO'O ARMEL C.	Chargé de Cours	En poste
18-OLLE OLLE Daniel C. D.	Chargé de Cours	S/D ENSET Ebolowa
19-TAPAMO Hyppolite	Chargé de Cours	En poste
20-BAYEM Jacques Narcisse	Assistant	En poste
21-MAKEMBE. S . Oswald	Assistant	Directeur CUTI
22-NKONDOCK. MI. BAHANACK.N.	Assistant	En poste
7- DÉPARTEMENT DE MATHÉMATIQUES (MA)		
1-AYISSI Raoult Domingo	Professeur	Chef de Département
2-EMVUDU WONO Yves S.	Professeur	Inspecteur MINESUP
3-KIANPI Maurice	Maître de Conférences	En poste
4-MBANG Joseph	Chargé de Cours	En poste
5-MBEHOU Mohamed	Maître de Conférences	En poste
6-MBELE BIDIMA Martin L.	Chargé de Cours	En poste
7-NKUIMI JUGNIA Célestin	Maître de Conférences	En poste
8-NOUNDJEU Pierre	Maître de Conférences	CS P & Diplômes
9-TCHAPNDA NJABO Sophonie B.	Maître de Conférences	D/AIMS Rwanda
10-TCHOUNDJA Edgar Landry	Maître de Conférences	En poste
11-BOGSO ANTOINE MARIE	Chargé de Cours	En poste
12-AGHOUKENG JIOFACK Jean G.	Chargé de Cours	CC MINPLAMAT
13-CHENDJOU Gilbert	Chargé de Cours	En poste
14-DJIADEU NGAHA Michel	Chargé de Cours	En poste
15-DOUANLA YONTA Herman	Chargé de Cours	En poste
16-FOMEKONG Christophe	Chargé de Cours	En poste
17-KIKI Maxime Armand	Chargé de Cours	En poste
18-MBAKOP Guy Merlin	Chargé de Cours	En poste
19-MENGUE MENGUE David J.	Chargé de Cours	En poste
20-NGUEFACK Bernard	Chargé de Cours	En poste
21-NIMPA PEFOUKEU R.	Chargée de Cours	En poste
22-POLA DOUNDOU E.	Chargé de Cours	En poste
23-TAKAM SOH Patrice	Chargé de Cours	En poste
24-TCHANGANG Roger Duclos	Chargé de Cours	En poste
25-TETSADJIO TCHILEPECK M. E.	Chargée de Cours	En poste
26-TIAYA TSAGUE N. Anne-Marie	Chargée de Cours	En poste
27-BITYE MVONDO Esther C.	Assistante	En poste
28-FOKAM Jean Marcel	Assistante	En poste
29-LOUMNGAM KAMGA Victor	Assistante	En poste
30-MBATAKOU Salomon Joseph	Assistante	En poste
31-MBIAKOP Hilaire G.	Assistant	En poste

32-MEFENZA NOUNTU T.	Assistant	En poste
33-OGADOA AMASSAYOGA	Assistant	En poste
34-TCHEUTIA Daniel Duviol	Assistant	En poste
35-TENKEU JEUFACK Y. L.	Assistant	En poste
8- DÉPARTEMENT DE MICROBIOLOGIE (MIB) (21)		
1-ESSIA NGANG Jean J.	Professeur	Chef de Département
2-NYEGUE Maximilienne A.	Professeur	VICE-DOYEN / DSSE
3-NWAGA Dieudonné M.	Professeur	En poste
4-ASSAM ASSAM Jean Paul	Maître de Conférences	En poste
5-BOYOMO ONANA	Maître de Conférences	En poste
6-KOUITCHEU M. Epse KOUAM L. B.	Maître de Conférences	En poste
7-RIWOM Sara Honorine	Maître de Conférences	En poste
8-SADO KAMDEM Sylvain Leroy	Maître de Conférences	En poste
9-BODA Maurice	Chargé de Cours	En poste
10-BOUGNOM Blaise Pascal	Chargé de Cours	En poste
11-ESSONO OBOUGOU Germain G.	Chargé de Cours	En poste
12-NJIKI BIKOÏ Jacky	Chargée de Cours	En poste
13-TCHIKOUA Roger	Chargé de Cours	En poste
14-ESSONO Damien Marie	Assistant	En poste
15-LAMYE Glory MOH	Assistant	En poste
16-MEYIN A EBONG Solange	Assistante	En poste
17-MONI NDEDI E. Del F.	Assistante	En poste
18-NKOUDOU ZE Nardis	Assistant	En poste
19-SAKE NGANE Carole Stéphanie	Assistante	En poste
20-TAMATCHO KWEYANG B. P.	Assistante	En poste
21-TOBOLBAÏ Richard	Assistant	En poste
9- DEPARTEMENT DE PYSIQUE(PHY) (44)		
1-BEN- BOLIE Germain Hubert	Professeur	En poste
2-DJUIDJE KENMOE épouse A.	Professeur	En poste
3-EKOBENA FOUDA Henri Paul	Professeur	C D. UN
4-ESSIMBI ZOBO Bernard	Professeur	En poste
5-KOFANE Timoléon Crépin	Professeur	En poste
6-NANA ENGO Serge Guy	Professeur	En poste
7-NANA NBENDJO Blaise	Maître de Conférences	En poste
8-NDJAKA Jean Marie B.	Professeur	Chef de Département
9-NJANDJOCK NOUCK Philippe	Professeur	SD/ MINRESI
10-NOUAYOU Robert	Professeur	En poste
11-PEMHA Elkana	Professeur	En poste
12-TABOD Charles TABOD	Professeur	Doyen Univ/Bda
13-TCHAWOUA Clément	Professeur	En poste
14-WOAFO Paul	Professeur	En poste
15-ZEKENG Serge Sylvain	Professeur	En poste

16-BIYA MOTTO Frédéric	Maître de Conférences	DG/HYDRO Mekin
17-BODO Bertrand	Maître de Conférences	En poste
18-ENYEGUE A NYAM épse B.	Chargée de Cours	En poste
19-EYEBE FOUDA Jean sire	Maître de Conférences	En poste
20-FEWO Serge Ibraïd	Maître de Conférences	En poste
21-HONA Jacques	Maître de Conférences	En poste
22-MBANE BIOUELE César	Maître de Conférences	En poste
23-MBINACK Clément	Maître de Conférences	En poste
24-NDOP Joseph	Maître de Conférences	En poste
25-SAIDOU	Maître de Conférences	CC/IRGM/MINRESI
26-SIEWE SIEWE Martin	Maître de Conférences	En poste
27-SIMO Elie	Maître de Conférences	En poste
28-VONDOU Derbetini A.	Maître de Conférences	En poste
29-WAKATA née BEYA Annie	Maître de Conférences	Directeur/ENS/UYI
30-ABDOURAHIMI	Chargé de Cours	En poste
31-CHAMANI Roméo	Chargé de Cours	En poste
32-EDONGUE HERVAIS	Chargé de Cours	En poste
33-FOUEDJIO David	Chargé de Cours	Chef Cell. MINADER
34-MBONO SAMBA Yves C. U.	Chargé de Cours	En poste
35-MELI'I Joelle L.	Chargée de Cours	En poste
36-MVOGO ALAIN	Chargé de Cours	En poste
37-OBOUNOU Marcel	Chargé de Cours	DA/U I E/Sangmalima
38-WOULACHE Rosalie L.	Chargée de Cours	En poste
39-AYISSI EYEBE Guy F. V.	Assistant	En poste
40-DJIOTANG TCHOTCHOU L. A.	Assistant	En poste
41-LAMARA Maurice	Assistant	En poste
42-OTTOU ABE M. T.	Assistant	En poste
43-TEYOU NGOUPOU Ariel	Assistant	En poste
44-WANDJI NYAMSI W.	Assistant	En poste
10- DÉPARTEMENT DE SCIENCES DE LA TERRE (ST) (42)		
1-BITOM Dieudonné	Professeur	D / FASA / UDs
2-FOUATEU Rose épse YONGUE	Professeur	En poste
3-NDAM NGOUPAYOU J.R.	Professeur	En poste
4-NDJIGUI Paul Désiré	Professeur	Chef de Département
5-NGOS III Simon	Professeur	DAAC/Uma
6-NKOUMBOU Charles	Professeur	En poste
7-NZENTI Jean-Paul	Professeur	En poste
8-ABOSSOLO née ANGUE M.	Maître de Conférences	VD / DRC
9-BISSO Dieudonné	Maître de Conférences	D/P B Memve'ele
10-EKOMANE Emile	Maître de Conférences	En poste
11-GANNO Sylvestre	Maître de Conférences	En poste
12-GHOGOMU Richard TANWI	Maître de Conférences	CD/Uma
13-MOUNDI Amidou	Maître de Conférences	CT/ MINIMDT
14-NGUEUTCHOUA Gabriel	Maître de Conférences	CEA/MINRESI
15-NJILAH Isaac KONFOR	Maître de Conférences	En poste

16-NYECK Bruno	Maître de Conférences	En poste
17-ONANA Vincent Laurent	Maître de Conférences	CS M & du Matériel
18-TCHAKOUNTE J. épse NUMBEM	Maître de Conférences	Chef.cell / MINRESI
19-TCHOUANKOUE Jean-Pierre	Maître de Conférences	En poste
20-TEMDJIM Robert	Maître de Conférences	En poste
21-YENE ATANGANA Joseph Q.	Maître de Conférences	CD. /MINTP
22-ZO'O ZAME Philémon	Maître de Conférences	DG/ART
23-ANABA ONANA Achille B.	Chargé de Cours	En poste
24-BEKOA Etienne	Chargé de Cours	En poste
25-ELISE SABABA	Chargé de Cours	En poste
26-ESSONO Jean	Chargé de Cours	En poste
27-EYONG JOHN TAKEM	Chargé de Cours	En poste
28-FUH Calistus Gentry	Chargé de Cours	S. E/MINMIDT
29-LAMILEN BILLA Daniel	Chargé de Cours	En poste
30-MBESSE CECILE OLIVE	Chargée de Cours	En poste
31-MBIDA YEM	Chargé de Cours	En poste
32-METANG Victor	Chargé de Cours	En poste
33-MINYEM Dieudonné-Lucien	Chargé de Cours	CD/Uma
34-NGO BELNOUN Rose N.	Chargée de Cours	En poste
35-NGO BIDJECK Louise M.	Chargée de Cours	En poste
36-NOMO NEGUE Emmanuel	Chargé de Cours	En poste
37-NTSAMA ATANGANA J.	Chargé de Cours	En poste
38-TCHAPTCHET TCHATO De P.	Chargé de Cours	En poste
39-TEHNA Nathanaël	Chargé de Cours	En poste
40-TEMGA Jean P.	Chargé de Cours	En poste
41-FEUMBA Roger	Assistant	En poste
42-MBANGA NYOBE Jules	Assistant	En poste

Répartition chiffrée des Enseignants de la Faculté des Sciences de l'Université de Yaoundé I

NOMBRE D'ENSEIGNANTS					
DÉPARTEMENT	**Professeurs**	**Maîtres de Conférences**	**Chargés de Cours**	**Assistants**	**Total**
BCH	8 (01)	14 (10)	13 (05)	05 (02)	**40 (18)**
BPA	15 (01)	8 (06)	18 (05)	10 (03)	**51 (15)**
BPV	07 (01)	9 (01)	8 (06)	07 (01)	**31 (9)**
CI	10 (01)	09 (02)	10 (02)	03 (0)	**32 (5)**
CO	6 (0)	21 (05)	05 (02)	08 (02)	**40 (9)**
IN	2 (0)	2 (0)	15 (01)	03 (00)	**22 (1)**
MAT	2 (0)	8 (0)	15 (01)	09 (02)	**34 (7)**
MIB	3 (0)	5 (03)	06 (01)	06 (02)	**20 (6)**
PHY	15 (0)	14 (02)	09 (03)	08 (03)	**46 (8)**
ST	7 (1)	15 (01)	18 (05)	02 (0)	**42 (7)**
Total	**75 (5)**	**105 (30)**	**117 (31)**	**61 (15)**	**358 (85)**

Soit un total de **358 (85)** dont :

- Professeurs **75 (5)**
- Maîtres de Conférences **105 (30)**
- Chargés de Cours **117 (31)**
- Assistants **61 (15)**

( ) = Nombre de Femmes **85**

Dédicace

A

ma maman Odette Chebou

Remerciements

Au Professeur Maurice TCHUENTE

Pour avoir accepté de présider mon jury de soutenance de thèse malgré ses multiples occupations. Également pour ses remarques et suggestions qui ont été capitales pour la finalisation de ce travail.

Au Professeur René Ndoundam

Pour avoir joué un rôle déterminant pour l'encadrement et l'aboutissement de cette thèse. J'ai été honoré de partager son bureau durant ces années de dur labeur. A plus d'un titre je remercie le Professeur René Ndoundam pour son abnégation au travail, ses innombrables conseils, son expertise poussée et surtout sa disponibilité. Je n'oublierais pas de mentionner la passion de son métier et la patience dont il a fait preuve vis à vis de moi.

Au Professeur Marcel FOUDA NDJODO

Pour avoir accepté d'examiner cette thèse au regard de ses multiples occupations. De plus, pour l'intérêt qu'il a bien voulu porter sur ce travail.

Au Professeur Blaise YENKE

Pour avoir accepté d'expertiser cette thèse. Aussi pour ses remarques et observations qui ont été nécessaire à l'amélioration de la qualité de ce travail.

Au Professeur Norbert TSOPZE

Pour avoir accepté d'expertiser cette thèse. De même pour la pertinence de ses remarques qui ont permis de produire un travail de meilleur qualité.

Au Professeur Alain Tchana

A qui je témoigne toute ma gratitude au regard de l'encadrement exceptionnel reçu et ayant contribué à améliorer la qualité de la présentation et surtout à solidifier les travaux à l'aide d'expérimentations.

A tout le Corps Enseignant du Département d'Informatique de L'Université de Yaoundé I

Pour tous les enseignements reçus.

A tout mes camarades du département

Avec qui tout au long des années nous avons eu à travailler dur pour arriver à nos fins.

A ma famille

Pour le soutien sans faille et la patience qu'ils ont su me montrer :

Ma sœur et son époux

Docteur MOYOU christelle epse Malong et M. Malong Walter, pour leur soutien multiforme durant ces longues années de doutes et d'incertitudes.

A mon grand frère

Professeur Tchakoutio Alain, qui m'a tenu la main et m'a orienté durant ces années. Je te suis infiniment reconnaissant pour avoir cru en moi et pour tes multiples conseils.

A mes frères

M. Ngandjon Carrel, M. Woudje Alex, M. Tindom Gervais et M. Kemgouo Francis, pour tous les encouragements et la bienveillance dont ils ont fait preuve.

A mon épouse et mes princesses

Mme Tapidji Félicia epse MOYOU, Chebou Eunice Roxane et Tapidji Yaelle Belgrade, pour le tout soutien et le réconfort pendant ces années de dur labeur.

A mes oncles et tantes

M. & Mme Sikapin ainsi que Mme Djuidje Hortense, pour leurs encouragements et leurs soutiens.

A mes parents

Mme Chebou Odette epse MOYOU ainsi qu'à M. MOYOU Samuel, pour tous les sacrifices investis en ma modeste personne.

Je remercie toutes les personnes dont j'aurai oublié le nom et qui aurait contribué d'une manière ou d'une autre à la réalisation de ce travail.

Table des matières

Résumé

La stéganographie est une branche de la cybersécurité qui vise à dissimuler l'existence d'une communication en incorporant un message secret à l'intérieur des supports de couverture de type texte, image, audio ou vidéo. Les approches traditionnelles de stéganographie reposent sur la modification du support représentant un risque de détection croissant face aux techniques d'analyse forensique des supports. Cependant, l'approche coverless évite toute modification du support en sélectionnant le support dont le contenu est similaire au message secret après application d'une fonction de hachage. Cette approche offre une alternative prometteuse mais encore limitée en termes de capacité, de robustesse et de flexibilité (adaptation difficile à différents contextes de message secret ou à divers types de supports). Cette thèse propose un nouveau paradigme de stéganographie sans modification du support basé sur la dissociation entre le contenu du support et l'information secrète, en utilisant un système d'indexation des fichiers dans des répertoires et un mécanisme d'indirection pour le stockage et le partage de fichiers dans les environnements multi-cloud. Le modèle repose sur quatre propriétés fondamentales : l'utilisation de tout type de fichiers(texte, image, vidéo, etc.) comme support potentiel, une indexation des supports à partir des descripteurs numériques de fichiers (date de modification, type de fichiers, nom de fichiers, taille des fichiers, etc.), une indirection entre les deux parties en communication assurée par la distribution des fichiers et leur partage à travers les services de stockage des clouds publics et enfin la reconstruction du message fondée sur l'ordre des fichiers sélectionnés et des clouds intervenants. Les résultats expérimentaux montrent une meilleure résistance à l'altération du support et une grande flexibilité par rapport aux méthodes basée sur le coverless. De plus l'analyse du coût de l'usage des espaces de stockage des clouds est nul. Les travaux futurs consisteront à diminuer la taille de la base de données de fichiers nécessaire à l'encodage du secret en proposant une méthode qui diminue à la fois ce nombre de fichiers par répertoire et augmente la capacité d'embarquement. Puis de diminuer le nombre de fichiers déplacés vers l'environnement de stockage multi-cloud lors de la dissimulation pour augmenter les performances de la méthode.

Mots clés : *Stéganographie, Coverless, Indexation des fichiers, Indirection, Environnement de stockage multi-cloud.*

Abstract

Steganography is a branch of cybersecurity that aims to conceal the existence of a communication by embedding a secret message within covert media such as text, images, audio, or video. Traditional steganography approaches rely on modifying the media, which poses an increasing risk of detection when faced with media forensic analysis techniques. However, the coverless approach avoids any modification of the media by selecting media whose content is similar to the secret message after applying a hash function. This approach offers a promising alternative but is still limited in terms of capacity, robustness, and flexibility (difficult to adapt to different secret message contexts or various media types). This thesis proposes a new paradigm of steganography without modification of the medium based on the dissociation between the content of the medium and the secret information, using a system of indexing files in directories and an indirection mechanism for storing and sharing files in multi-cloud environments. The model is based on four fundamental properties : the use of any type of files (text, image, video, etc.) as potential medium, an indexing of the media from the digital descriptors of files (modification date, file type, file name, file size, etc.), an indirection between the two communicating parties ensured by the distribution of the files and their sharing through the storage services of public clouds and finally the reconstruction of the message based on the order of the selected files and the intervening clouds. The experimental results show a better resistance to the alteration of the medium and a great flexibility compared to methods based on the coverless. In addition, the analysis of the cost of using cloud storage spaces is null. Future work will involve reducing the size of the file database required to encrypt the secret by proposing a method that both reduces the number of files per directory and increases the insertion capacity. Then, optimize the method by reducing the number of files moved within the multi-cloud storage environment during concealment.

Keywords : *Steganography, Coverless, File Indexing, Indirection, Multi-Cloud Storage Environment.*

Table des figures

Liste des tableaux

Liste des Abréviations

ADN	Acide Désoxyribonucléique
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
BMP	BitMaP
BMS	Bits les Moins Significatifs
CU	Charge Utile
DCT	Discrete Cosine Transformation
EQM	Erreur Quadratique Moyenne
FBI	Federal Bureau of Investigation
FPGA	Field Programmable Gate Array
HRA	Habileté de Résistance aux Attaques
IPv6	Internet Protocol version 6
JPEG	Joint Photographic Experts Group
LSBS	Least Significant bit Substitution
LTE	Long-Term Evolution
LZW	Lempel Ziv Welch
MBR	Master Boot Record
MPEG	Moving Picture Experts Group
OSI	Open System Interconnexion
PDF	Portable Document File
PDU	Unité de Données de Protocole
PSNR	Peak Signal to Noise Ratio
RSA	Rivest Shamir Adleman

SCTP	Stream Control Transmission Protocol
SRS	Success Rate of Secret
TCD	Transformation en Cosinus Discret
TCP/IP	Transmission Control Protocol/Internet Protocol
TOD	Transformation en Ondelettes Discrète
WLAN	Wireless Local Area Network

Introduction Générale

Contexte

Le besoin de communication a favorisé l'essor de nouvelles technologies de l'information et de la communication à l'instar de l'Internet. Ce réseau à l'échelle mondiale offre de nombreux services aux internautes notamment la consultation des pages web, le téléchargement des fichiers, la messagerie électronique, la vidéo à la demande, la communication instantanée et bien d'autres. Au même moment, les utilisateurs sont en permanence exposés à des attaques visant les données personnelles, les infrastructures et les services déployés. Les délits enregistrés sur le cyberespace sont néfastes et variés. Il s'agit des intrusions non autorisées dans les systèmes, l'infection des machines par des programmes malveillants, la prise de contrôle à distance des ordinateurs, le téléchargement illégal, l'indisponibilité des services par une attaque par déni de service...

La sécurité des systèmes et de l'interaction des machines et des applications sont désormais un enjeu sérieux pour tous : citoyens, petites et grandes entreprises, organes administratifs et étatiques. La sécurité informatique également appelée cybersécurité [1] est la protection des systèmes informatiques contre le vol et l'endommagement du matériel, du logiciel ou des informations ainsi que la protection contre toute perturbation ou mauvaise direction des services qu'ils fournissent. La cybersécurité consiste à garantir trois propriétés fondamentales et essentielles concernant les informations, les services et les infrastructures : la confidentialité, l'intégrité et la disponibilité. La confidentialité assure que les informations ne sont divulguées qu'aux personnes autorisées. L'intégrité assure que le système ou les données ne sont modifiées que par une action volontaire et légitime. Tandis que la disponibilité assure qu'un système ou une information est accessible en temps opportun aux utilisateurs. Ainsi, sécuriser un système d'information signifie empêcher une entité non autorisée (utilisateur, processus, service, machine) d'accéder, de modifier ou de rendre inaccessibles des données informatiques, des services et infrastructures informatiques. Notons que d'autres propriétés élémentaires telles que l'authenticité (preuve de l'origine des informations) et la non répudiation (garantie qu'une transaction ne peut être niée) peuvent également être citées.

Trois principaux mécanismes [2] sont mis en œuvre pour garantir les propriétés fondamentales de la sécurité informatique. Premièrement, les mécanismes de prévention

pour empêcher toute violation de ces propriétés. Puis les mécanismes de détection pour identifier toute tentative réussie. Enfin les mécanismes de réaction pour déployer des moyens spécifiques pour atténuer ou inhiber toute tentative réussie. Multiples sont les techniques employées pour chacun de ces mécanismes de sécurité. La prévention fait intervenir des techniques telles que la cryptographie, la stéganographie et le watermarking. En ce qui concerne la détection, des techniques de détection d'intrusions et d'anomalies ou de corrélation d'alertes sont implémentées. Le blocage de l'attaque, la reconfiguration du système et la contre-attaque sont les techniques de sécurité réactive en réponse aux attaques réussies.

La cryptographie [3] est l'art de l'écriture sécrète en convertissant l'information confidentielle sous une forme inintelligible. L'algorithme de chiffrement prend en entrée le message secret et la clé puis produit en sortie le message chiffré. Tandis que l'algorithme de déchiffrement prend en entrée le texte chiffré et la clé pour produire le message secret initial. Les clés doivent être solides et doivent se conserver dans un endroit sûr, afin de rendre le décryptage incassable. Les clés sont regroupées en clés secrètes ou symétriques et en clés publiques ou asymétriques. Ainsi, les clés secrètes sont celles qui sont connues uniquement de l'expéditeur et du destinataire. A l'inverse, les clés publiques sont connues de tous et sont nécessaires uniquement pour le chiffrement des données secrètes. Le déchiffrement nécessite une clé privée détenue exclusivement par le destinataire. Lorsque les données confidentielles sont extraites à partir du message chiffré : on parle de cryptanalyse.

Le watermarking ou tatouage numérique [4] est une technique permettant d'ajouter des informations de copyright à des documents numériques pour faciliter leur identification ou protéger les droits d'auteur. Le message inclut dans le document appelé marque, est un ensemble de bits dont le contenu dépend de l'application. La marque peut être l'identifiant du créateur, du propriétaire, de l'acheteur ou alors une forme de signature pour détecter des contrefaçons. En général, nous distinguons deux classes de marquage : ceux dits fragile et semi-fragile. Avec le marquage fragile, il est impossible d'extraire le message après une modification du document. Cela permet de détecter toute modification par un tiers conduisant à des contrefaçons. En outre, le marquage semi-fragile est capable d'extraire le message avec précision, même après modification du document. Cette technique est généralement utilisée pour la protection des droits d'auteur.

Enfin, la stéganographie [5] qui est un art de communication secrète dans lequel les informations confidentielles sont embarquées dans un média de couverture. Ainsi, seuls l'émetteur et le destinataire sont les seuls ayant connaissance de l'existence du message secret. Plusieurs supports peuvent être utilisés pour dissimuler l'information confidentielle : les textes, les images, les sons et les vidéos. Notons que la stéganographie a un avantage sur la cryptographie, en ce sens que le support stéganographique après incorporation du secret semble être le même que le support de couverture initial. Par conséquent, ce support modifié transite sur le réseau vers le destinataire de manière anodine sans attirer l'attention. Un couplage de cryptographie et stéganographie améliore considérablement la sécurité des données confidentielles. Dans ce cas, le secret est tout

d'abord chiffré avant d'être dissimulé dans le support de couverture.

De manière générale, les techniques de sécurité des données telles que la cryptographie, le watermarking et la stéganographie ont suscité un immense intérêt des chercheurs. En effet, ces trois techniques de sécurité sont si proches les unes des autres que leur objectif fondamental est d'assurer la confidentialité des données pendant la transmission. Cependant, les principes de fonctionnement sont différents les uns des autres.

Nous nous intéressons dans cette thèse aux mécanismes de prévention pour empêcher toutes violations d'accès aux données confidentielles, plus particulièrement par l'usage des techniques de stéganographie qui assurent la sécurité du secret par dissimulation à l'intérieur des supports de communication. Ce mode d'échanges d'informations confidentielles à l'aide de la stéganographie considère que la communication établie est imperceptible par quiconque et connue uniquement de l'émetteur et du destinataire, à la différence de la cryptographie où l'on sait qu'il y a une communication chiffrée en cours.

Problématique de la thèse

En stéganographie, la dissimulation du secret dans les supports doit se faire de la manière la pus imperceptible possible. C'est pourquoi les supports sont généralement modifiés pour intégrer les données confidentielles. Mais cette approche est aujourd'hui dépassée car la stéganalyse est capable de détecter les messages secrets dissimulés à l'intérieur de ces supports. Un nouveau courant de stéganographie a vu le jour pour évader les outils de stéganalyse appelé **coverless** ou **zero-steganography** (sans modification du support). L'idée est de sélectionner le support qui contient déjà tout le secret ou une partie du secret à transférer. Ainsi, le support original n'est pas modifié et ce dernier est envoyé au destinataire qui pourra utiliser une fonction de hachage robuste pour extraire le secret.

Après analyse, les méthodes basées sur le coverless [6],[7],[8],[9] sont limitées par :

- une faible capacité d'embarquement ;
- la construction préalable de la base de données de fichiers permettant de transférer le secret. Les fichiers sélectionnés doivent avoir un lien avec le secret à envoyer ;
- un nombre élevé de fichiers stégo à transférer au destinataire ;
- le type de support utilisé est le même à chaque fois ;
- une communication directe entre les parties qui communiquent ce qui peut attirer l'attention ;
- la vulnérabilité à la distorsion (altération) du support.

La problématique de cette thèse est la suivante :

Comment concevoir un modèle de communication robuste aux attaques par distorsion avec sélection du support n'ayant aucune corrélation avec le secret à transférer ?

Questions de recherche

La question de recherche principale se formule ainsi : Comment dissimuler un secret sans dépendre du support à l'aide d'une communication indirecte avec le destinataire sans éveiller de soupçon ?

Les questions de recherche secondaires sont les suivantes :

- comment pallier à la vulnérabilité des attaques par distorsion ?
- comment mettre en œuvre l'indépendance du support ?

- comment transférer un grand volume de données sans attirer l'attention ?

Hypothèses de recherche

Nous formulons les hypothèses suivantes qui doivent être vérifiées à la fin de ce travail :

Hypothèse 1 : La méthode de dissimulation utilisée permet de récupérer la totalité du secret même en cas d'altération du support ;

Hypothèse 2 : Tout type de fichiers multimédia peut être utilisé lors de la dissimulation ;

Méthodologie de recherche

La méthodologie employée dans cette thèse est décrite de la manière suivante :

- la construction de la base de données utilise des pointeurs ou référencement local sur les fichiers ;

- la dissimulation du secret est basée sur l'approche par sélection du support(Coverless) ;

– la sélection des supports se fait sur tout type d'extensions de fichiers par l'assignation des index permettant leur identification et la dissimulation du secret ;

– le partage de données confidentielles est basé sur l'indirection mise en oeuvre à l'aide des environnements de stockage multiclouds

Objectifs de la thèse

Dans cette thèse, nous nous intéressons au problème de détection des messages secrets dissimulés dans les supports de couverture à l'aide de la stéganographie. Notre objectif est de proposer des solutions de transmission et de partage d'informations confidentielles robuste à la détection et à l'extraction du secret. De plus, mettre en œuvre un moyen capable de masquer l'existence du canal secret entre les parties communicantes. Nous considérons que les supports de couverture échangés durant la communication ne subissent aucune modification, garantissant ainsi une sécurité supérieure [10]. De ce fait, le fichier porte l'information sans être altéré et la seule façon d'y accéder est de posséder la clé. Dans ce travail, chaque support de couverture utilisé constitue un pointeur vers le secret.

Le but sera donc d'adopter un mécanisme non seulement subtil mais aussi transparent à tout attaquant. L'intégration du secret vise à exploiter les services populaires disponibles sur internet en maintenant leur principal objectif sans déclencher de suspicion auprès des utilisateurs. Il s'agit notamment du service de stockage des fichiers dans un environnement multi-cloud et de l'ouverture des pages web. Ce mécanisme crée une rupture avec les supports habituels (texte, image, son, vidéo, etc.) masquant ainsi l'existence d'un canal secret entre les parties communicantes.

Contributions

Dans le but de résoudre le problème de recherche défini dans cette thèse, puis de proposer des solutions efficaces pour palier aux limites des méthodes existantes, nous proposons de nouvelles méthodes pour la dissimulation du secret en considérant non seulement l'intégrité du support de couverture mais aussi les services de données en ligne. Ces considérations mettent en lumière l'intérêt en cours sur le camouflage des données sans modification du support de couverture. L'objectif principal est de définir une nouvelle approche pour faire correspondre les blocs du secret avec des fichiers de données sans compromettre le secret en cas de d'altération du fichier pendant le transfert. Ce qui est différent de la récente approche de la littérature basée sur le coverless qui s'appuie sur les fonctions de hachage pour faire correspondre le secret au contenu du fichier, ne permettant pas de garantir l'extraction de la totalité du secret en cas d'altération du support. Ces méthodes sont principalement limitées par le fait que malgré la transmission des données sans modification du support, l'analysent de la robustesse lors de la transmission du secret montre des taux d'erreurs non nuls en cas d'altération du

support lors de l'extraction du secret. Dans le but d'assurer une transmission du secret et son extraction dans son entièreté, nous allons considérer que le secret n'est plus lié au contenu du fichier mais plutôt à une indexation des fichiers stockés dans des répertoires. Ainsi, les trois contributions majeures sont les suivantes :

1. **L'indexation des supports de couverture** : Nous proposons deux méthodes de dissimulation basées sur l'indexation des supports. La première méthode utilise les supports de fichiers multimédia. A chaque support sera associé un index ou un numéro en fonction d'un ordre de classement des fichiers à l'intérieur des répertoire. Cet ordre peut être par exemple : le classement des fichiers par nom, par type ou par date de création. Cela va permettre de faire correspondre chaque bloc du secret au moment de la dissimulation à un index de fichier et non plus à son contenu. Tandis que la seconde méthode utilise comme support les URL. Une liste de n URL d'un site web est sélectionnée pour cacher le secret. Ainsi, nous faisons correspondre chaque bloc du secret à une séquence d'URL en utilisant les permutations d'URL. Cela est possible grâce à l'utilisation de deux fonctions : Rank qui détermine la permutation de n URL connaissant son numéro (secret) et Unrank pour effectuer l'opération inverse. En somme, l'indexation de fichiers permet de garantir que le secret à transférer n'a pas de corrélation en terme de contenu avec les fichiers utilisés comme support lors de la dissimulation.

2. **Extensible à tout type de format de fichiers** : Avec l'indexation, un format de fichier précis n'est pas obligatoire. Car, les méthodes existantes fondent leur dissimulation soit sur un seul type de format de fichiers par exemple les images. Aini, la nature du format du fichier importe peu, puisqu'il sera possible de combiner à la fois les images, les documents textes, videos, audio ou même les URL en leur associant un numéro d'ordre. Cela crée une rupture avec l'existant en terme de format de fichier et contribue à rendre les méthodes de dissimulation imperceptible et qui n'attire pas l'attention par rapport aux schémas classiques.

3. **L'indirection entre l'émetteur et le destinataire** : Le secret une fois dissimulé dans le support n'est pas directement transféré au destinataire. Un intermédiaire est placé entre l'émetteur et le destinataire pour couper tout lien direct entre les deux et ainsi effacer toute trace de communication. Car s'il est établi qu'il y a communication, les conséquences possibles sont la destruction des supports ou l'interruption de la communication par quelque moyen que ce soit. La première méthode proposée utilise comme intermédiaire le stockage du cloud à travers le partage de fichiers entre les communicants. L'émetteur stocke des fichiers et les partage au destinataire. Ce dernier peut y avoir accès en utilisant le même compte (si l'émetteur à partagé ses accès avec lui) ou à travers son compte personnel. Tandis que la seconde méthode utilise un serveur web à travers l'ouverture des pages web chez l'émetteur pour transférer le secret et l'extraction du secret au niveau du serveur web.

Plan de la thèse

La suite de cette thèse est organisée en quatre chapitres. Nous débuterons le chapitre 1 par la présentation des concepts de base en stéganographie et la classification des techniques de dissimulations. Le chapitre 2 illustrera ensuite l'état de l'art sur la détection du secret. Puis suivra le chapitre 3, dévolu à la proposition d'un nouveau schéma de dissimulation dans l'espace de stockage multi-cloud. Le chapitre 4 propose une seconde solution basée sur les séquences de requêtes HTTP. Enfin, nous terminerons par une conclusion générale.

Chapitre 1

Stéganographie : l'art de la dissimulation du secret

1.1 Introduction

Ce chapitre est dédié à la présentation des concepts en stéganographie. Il revisite le principe et les critères de performances des schémas stéganographiques. Puis présente la classification des différentes techniques de dissimulation des informations confidentielles et leurs applications dans la vie de tous les jours. Le chapitre s'achève sur la description des attaques connues en stéganalyse permettant de rechercher l'existence des messages secrets.

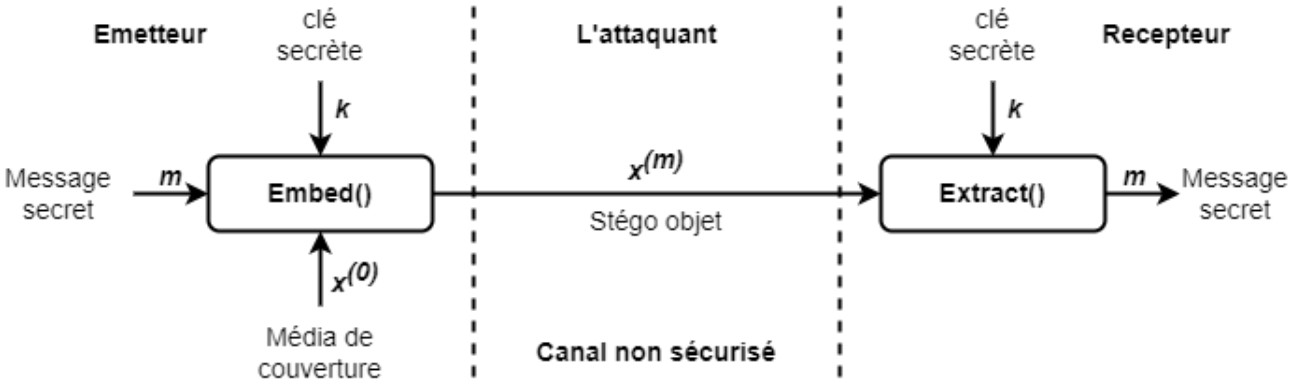
1.2 Modèle de communication secrète

1.2.1 Problème du prisonnier et du canal secret

En 1984, Gustavus Simmons présente l'intérêt de la stéganographie à travers le problème des prisonniers [11], où les complices d'un crime, Alice et Bod ont été arrêtés et emprisonnés dans des cellules séparées. Ainsi, leur seul moyen de communication après leur incarcération sera par le biais d'échanges de messages transmis par le gardien Oscar. Cependant, comme le gardien a toutes les raisons de soupçonner que les détenues voudraient coordonner un plan d'évasion, il n'admettra les échanges que si les informations contenues dans ces messages lui sont totalement ouvertes et vraisemblablement inoffensives. En revanche, Alice et Bod doivent être ingénieux pour établir un canal secret et le rendre indétectable. Par conséquent, les détenus doivent utiliser un canal invisible au gardien. Par ailleurs, si le gardien détecte une communication secrète entre les deux, non seulement leur communication sera interrompue mais leur plan d'évasion échouera aussi.

La stéganographie [5] est donc l'art et la science consistant à cacher des informations confidentielles de sorte que leur présence ne puisse être détectée. Son objectif est de mettre en place un canal secret pour les communications secrètes afin que personne, à l'exception du destinataire, ne soit au courant de l'existence du message secret. Les techniques classiques de stéganographie dissimulent les informations à l'intérieur des fichiers numériques tels que les fichiers textes, images, audio et vidéos [12, 13, 14, 15].

La Figure 1.1 présente le modèle de communication à travers un canal secret[16]. Ce canal diffuse les informations secrètes de sorte que tout adversaire ou attaquant ne puisse observer ce qui se passe. Nous illustrons sur cette figure comment les échanges exploitent le canal de communication non sécurisé, sur lequel est superposé le contenu légitime et les données secrètes à l'aide du stégo objet.



$$\text{Stego system} = (\chi^*, M, K, Embed, Extract),$$
$$x^{(0)} \in \chi^*,\ m \in M,\ k \in K$$
$$Embed(x^{(0)}, m, k) = x^{(m)}$$
$$Extract(x^{(m)}, k) = m$$

FIGURE 1.1 – Modèle de communication secrète

Comme illustré à la Figure 1.1, ce modèle de communication possède cinq paramètres :

- **L'ensemble des supports de couverture** $X^{(*)}$: il s'agit de sélectionner un ou plusieurs supports dans lesquels sera incorporer le secret ;
- **L'ensemble des messages** M : il s'agit des messages secrets à transférer au destinataire ;
- **L'ensemble des clés** K : il s'agit des clés partagés entre l'émetteur et le destinataire pour garder le secret confidentiel. En l'absence de cette clé, il sera impossible à un attaquant d'extraire le secret ;
- **La fonction Embed** : il s'agit de la méthode de dissimulation employée. Elle prend en entrée le message secret, le support couverture et la clé secrète et produit en sortit un support modifié qui contient le secret ;

– **La fonction Extract** : il s'agit de la méthode d'extraction du message secret connaissant la clé secrète et le support modifié envoyé par l'émetteur.

1.2.2 Stéganographie et processus de dissimulation

Le processus général de steganographie [17] est visible dans la Figure 1.2. Pour réaliser des communications non suspectes, les éléments suivants sont utilisés :

– **L'objet de couverture** : il s'agit du support ou média utilisé pour embarquer les informations secrètes de manière à ce qu'il soit difficile de détecter leur présence.

– **Le message secret** : il s'agit des informations confidentielles à dissimuler dans l'objet de couverture.

– **La clé secrète** : il s'agit d'une donnée connue uniquement des deux parties en communication nécessaire à la dissimulation et à l'extraction du message secret.

– **L'algorithme d'insertion** : qui altère l'objet de couverture pour insérer le message secret en utilisant la clé secrète.

– **Le stégo objet** : il s'agit de l'objet de couverture obtenu après l'insertion du message secret. Il est encore appelé stéganogramme.

– **L'algorithme d'extraction** : qui consiste à dissocier le message secret du stégo objet en appliquant la clé secrète.

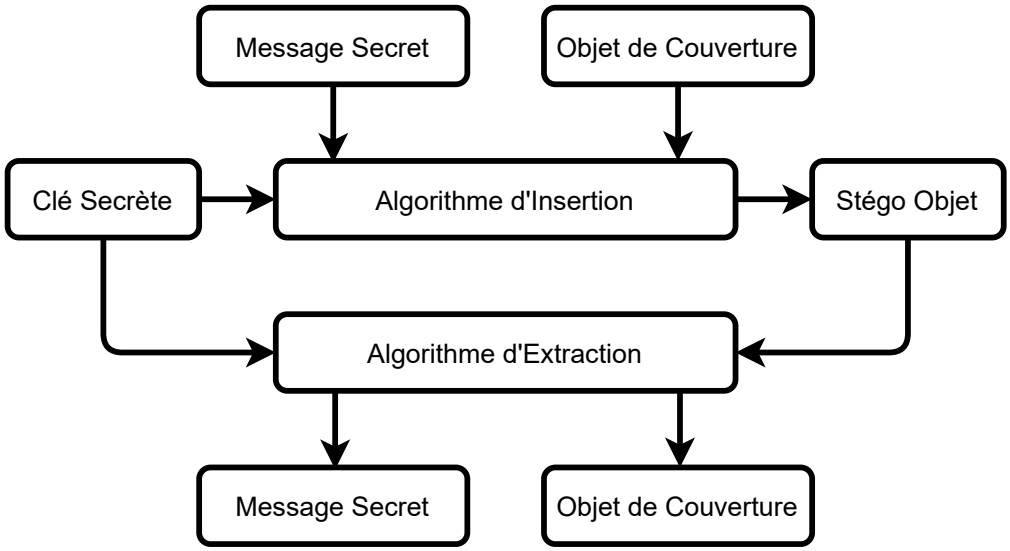


FIGURE 1.2 – Processus de dissimulation.

L'un des aspects les plus importants pour chaque méthode stéganographique est le choix du support approprié pour embarquer les données secrètes. Le support le plus favorable doit avoir les deux caractéristiques suivantes :

- **Le support doit être populaire** : autrement dit, l'utilisation d'un tel support ne doit pas être considérée comme une anomalie en soi. Plus ces supports sont présents et utilisés dans les réseaux mieux c'est, car ils masquent l'existence de la communication cachée.

- **La modification du support doit être illisible** : autrement dit, l'intégration du secret ne doit pas être visible par une tierce personne n'ayant pas connaissance de la procédure stéganographique en cours.

1.2.3 Métriques d'évaluation des performances d'une technique stéganographique

Une méthode stéganographique doit répondre à quatre principales caractéristiques [18] :

- **L'imperceptibilité** : qui fait référence à la qualité du stégo objet après l'incorporation du secret. L'objet de couverture et le stégo objet doivent être indistinguable visuellement. C'est la caractéristique clé de toute technique stéganographique. Une caractéristique importante permettant d'évaluer la performance des techniques de dissimulation appliquées aux images s'appelle le PSNR (Peak Signal to Noise Ratio). Elle calcule la distorsion dans le stégo média en Décibel (dB). Une meilleure qualité du stégo média peut être obtenue lorsque le PSNR est élevé. A l'inverse, lorsque cette valeur est base, cela oriente la détection du message secret. Elle peut être mesurée à l'aide de l'équation suivante :

$$PSNR = 10 \times log_{10} \frac{d^2}{EQM} \tag{1.1}$$

  Où d est la valeur maximum possible pour un pixel, dans le cas standard d'une image codée sur 8-bits, $d = 255$. Et EQM, l'Erreur Quadratique Moyenne . Elle est définie pour 2 images I_o et I_s de taille $m \times n$ par la formule suivante :

$$EQM = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I_o(i,j) - I_s(i,j))^2 \tag{1.2}$$

  Le PSNR est utile pour mesurer la proximité de l'image stéganographiée par rapport à l'image originale après insertion du message.

- **La Capacité** : ou Charge Utile (CU) fait référence à la quantité maximale de données cachées dans un objet de couverture. Elle s'exprime généralement en bits. Une technique efficace doit posséder une capacité d'embarquement élevée. Cependant, cela peut entraîner parfois une faible qualité du stégo objet, ce qui conduit

finalement à une distorsion visuelle. La capacité d'embarquement est le rapport de taille du secret et de la taille du support en bits. La formule est la suivante :

$$C = \frac{|Secret|}{|support|} \times 100 \tag{1.3}$$

– **La Robustesse** : fait référence à la résistance à la destruction ou à la modification des données secrètes durant la transmission. Le besoin d'évaluation de la robustesse né des évènements inattendu durant la transmission tels que le bruit, la mise à l'échelle, la compression, etc. Le taux d'erreur sur les bits : ou Bit Error Rate (BER), est utilisé pour déterminer si le message secret a été corrompu ou non lors de la transmission. Le BER est calculé à l'aide de l'équation suivante :

$$BER = \frac{\sum_{r=1}^{Z} xor(a_r, b_r)}{Z} \times 100 \tag{1.4}$$

Où a_r est le bit de message caché avant la transmission, b_r est le bit de message extrait après la transmission et Z est la longueur du message transmis en bits.

– **La Sécurité** : C'est une exigence fondamentale lors de l'élaboration d'une technique. Elle fait référence à l'Habileté de Résistance aux Attaques (HRA). Autrement dit, le stego objet doit être résistant aux techniques de stéganalyse. La stéganalyse revient en pratique à vérifier la statistique du support intercepté pour déterminer si elle est ou non altérer par un algorithme particulier Puis d'extraire le secret. De manière formelle, pour un support donnée $x = \{x_1, ..., x_n\}$, le problème de détection du message secret peut être représenté comme un test à deux hypothèses :

$$\begin{cases} H_0 & x \sim P_c \text{ le support x ne contient pas de message caché (cover) ,} \\ H_1 & x \sim P_c \text{ le support x contient un message caché (Stego) .} \end{cases}$$

En définitive, toute technique stéganographique doit posséder simultanément une capacité élevée, une meilleure imperceptibilité et une sécurité élevée. Toutefois, ces paramètres ont un impact négatif sur la performance des uns et des autres. La figure 1.3 montre les relations de conflits entre ces trois métriques. C'est pourquoi un compromis doit être pris en compte lors de la phase de l'élaboration de la technique stéganographique.

Il existe d'autres caractéristiques qui ne sont pas nécessairement exigeantes. Néanmoins, elles sont utilisées pour certaines applications spécifiques ou parfois pour améliorer la communication ou la sécurité du stégo objet. Il s'agit de :

– **La réversibilité** : qui fait référence à la récupération exacte du support de couverture et du secret après l'extraction du côté du destinataire. Ceci est principalement utilisé dans les applications où la recherche sans perte de données est primordiale, comme dans les applications médicales et militaires [19].

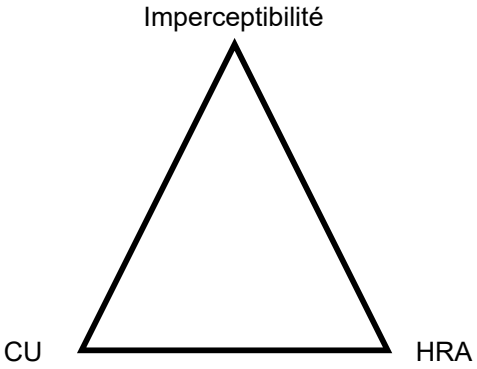


FIGURE 1.3 – Relation conflictuelle entre les trois principales métriques.

– **Le chiffrement** : utilisé pour fournir une sécurité supplémentaire au stégo objet, qui est généralement effectué en chiffrant le message secret à l'aide des algorithmes tels que : Advanced Encryption Standard (AES) , Rivest Shamir Adleman (RSA) , etc. Ainsi, le chiffrement des données secrètes est effectué avant la dissimulation [3].

– **La complexité de calcul** : qui évalue le temps de calcul et l'espace mémoire occupé dans les les techniques stéganographiques utilisées [20].

1.3 Classification des méthodes de stéganographie

1.3.1 Stéganographie ancienne

1.3.1.1 Apparition de la stéganographie

Tout au long de l'histoire, une multitude de méthodes et de variations ont été utilisées pour cacher des informations. Des écrits à travers l'histoire ont été enregistrés, relatant les histoires de Hérodote [21, 22]. Dans la Grèce antique, le texte était écrit sur des tablettes en bois. Ainsi, dans une des histoires, Dematerus voulait informer Sparte que Xerxès avait l'intention d'envahir la Grèce. Pour éviter la capture, il a écrit un message sur la tablette en bois et a ensuite recouvert la tablette avec de la cire. La tablette semblait vierge et inutilisée. La Figure 1.4 illustre les tablettes de cire.

Une autre méthode ingénieuse consistait à raser la tête d'un messager et tatouer un message ou une image sur la tête du messager. Après avoir laissé pousser ses cheveux, le message ne serait pas détecté jusqu'à ce que la tête soit à nouveau rasée.

Ces exemples illustrent bien les deux principales méthodes de stéganographie utilisées au cours des siècles, où l'existence d'un message est physiquement cachée sur le crâne d'un esclave, ou alors l'information est dissimulée sur un support qui transmet déjà l'information, comme les tablettes de cire.

FIGURE 1.4 – Illustration des tablettes de cire.

1.3.1.2 Ecriture dissimulée

Une autre forme courante d'écriture invisible consiste à utiliser des encres [23]. C'est le plus connu des procédés de stéganographie, dans lequel au milieu des textes écrits à l'aide d'encre, un message est écrit à l'aide de jus de citron, de lait ou de certains produits chimiques. Ces éléments sont invisibles à l'œil, mais pourtant une simple flamme ou un bain dans un réactif chimique révèle le message. La Figure 1.5 illustre un exemple réalisé à l'aide de lait.

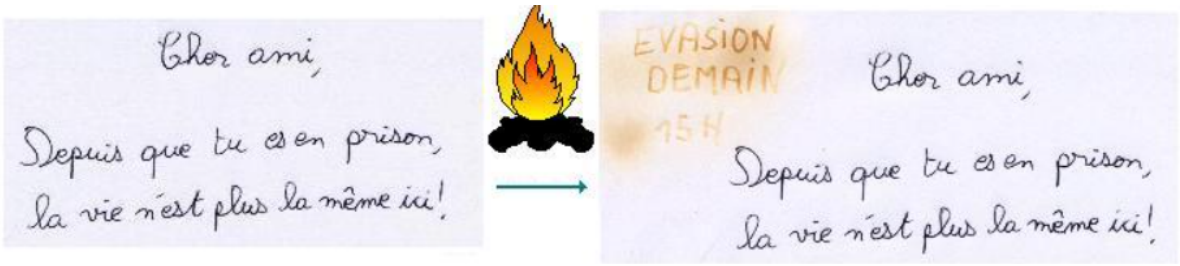


FIGURE 1.5 – Écriture dissimulée à l'aide d'encre.

Une autre méthode employée durant la seconde guerre mondiale par un espion Allemand, consistait à camoufler le secret dans un message à consonance innocente [21]. Le message envoyé par cet espion était le suivant :

Apparently neutral's protest is thoroughly discounted and ignored. Ismam hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

Cela semble tout à fait anodin. Maintenant, en prenant la deuxième lettre de chaque mot, nous obtenons : **"Pershing sails from NY June 1"** (le Pershing part de New-York le 1er juin).

Des méthodes plus sophistiquées furent utilisées par la suite, à l'instar de microfilms cachés sous des timbres postes ou sur des documents de magazine [21]. Les microfilms sont de petites photographies de la taille d'un caractère, mais qui peuvent contenir

l'équivalent d'une page de livre. En grossissant le microfilm à l'aide d'un microscope, la reproduction photographique d'une longue lettre tapée à la machine est révélée.

1.3.2 Méthodes actuelles

Avec l'avènement de l'informatique et le développement des échanges électroniques, les possibilités de cacher un message se sont multipliées. La stéganographie peut être utilisée dans différents domaines en fonction du type de support pris en compte. Ces supports peuvent être du texte, des disques et périphériques de stockage, les protocoles et trafic réseau, par voie logicielle ou par agencement des circuits, ou tout autre code ou transmission représenté numériquement. Ceux-ci fournissent d'excellents transporteurs pour les informations à cacher et de nombreuses techniques différentes ont été introduites. La classification des tendances actuelles [24] de la stéganographie est illustrée à la Figure 1.6.

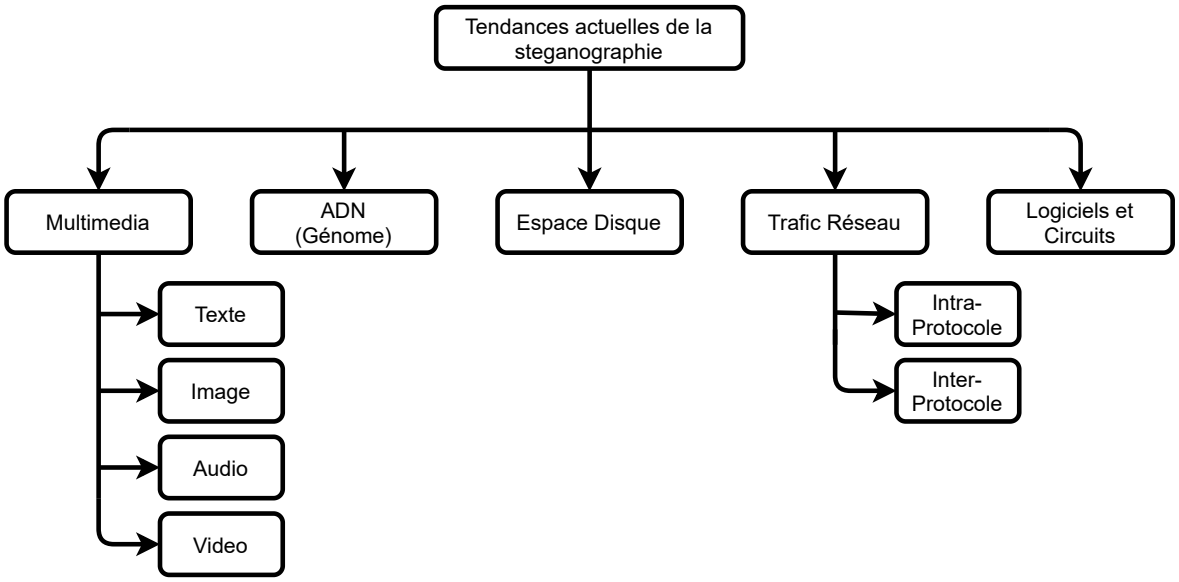


FIGURE 1.6 – Classification des tendances actuelles de la stéganographie

1.3.2.1 Stéganographie basée sur les fichiers multimédia

La stéganographie des médias numériques remonte aux années 1970 [25], lorsque les chercheurs se sont concentrés sur le développement de méthodes pour intégrer secrètement une signature dans une image numérique. De nombreuses méthodes différentes ont été proposées, entre autres, les modifications de Bits les Moins Significatifs (BMS) ou le codage par blocs de texture. Les techniques introduites étaient destinées aux images avec à la fois une compression avec perte, par exemple JPEG et une compression sans perte, par exemple BMP .

La variété d'algorithmes pour l'incorporation du secret dans les images numériques peut être regroupée selon le type d'altérations induites. Les modifications apportées sont

au niveau du bit, influençant ainsi les caractéristiques du domaine spatial de l'image ou affectant les caractéristiques du domaine fréquentiel. Alternativement, des subtilités de fichiers spécifiques peuvent être exploitées. De plus, un mélange de toutes ces techniques est possible. Les algorithmes utilisés peuvent impliquer des techniques telles que la Transformation en Cosinus Discret (TCD), la Transformation en Ondelettes Discrète (TOD) et la transformé de Fourrier, ce qui peut entraîner des altérations de la luminosité ou d'autres propriétés mesurables d'une image.

La stéganographie par l'image numérique est la branche la plus importante de la stéganographie, principalement orientée vers le fait de tromper le système visuel humain en lui faisant croire que la perception de l'image n'a pas été manipulée d'aucune façon. La même règle s'applique à tout le domaine de la stéganographie des médias numériques, dont la fonction principale est de faire croire à l'observateur que la "contrefaçon" fabriquée est en effet authentique. L'aspect de communication de l'algorithme stéganographique est secondaire par rapport au processus d'incorporation des données secrètes.

Le système auditif humain est tout aussi sujet aux illusions que la perception visuelle. La recherche s'est concentrée sur les fichiers audio tels que les fichiers MPEG. Les techniques développées comprenaient entre autres le masquage de fréquence, le masquage d'écho, le codage de phase et l'étalement du spectre. Il est également devenu évident que le codage de correction d'erreurs et un bon support supplémentaire pour la stéganographie audio, toute donnée redondante peut être utilisée pour transmettre le stéganogramme au prix d'une perte de robustesse face aux erreurs aléatoires. Cette idée a été utilisée plus tard dans la steganoraphie basée sur les protocoles réseaux.

Ensuite, les stéganographes ont pris les fichiers vidéo comme support cible. La plupart de méthodes proposées étaient des adaptations des algorithmes proposées pour les fichiers audio et images. Les solutions spécifiques à la vidéo impliquent l'utilisation de l'espace colorimétrique vidéo à une image comme support stéganographique ou des vecteurs de mouvements des images. Actuellement, les méthodes existantes utilisent les propriétés intrinsèques de la transmission vidéo, telles que le codage de mouvement.

Parallèlement à l'image numérique et à la stéganographie audio, l'information cachée dans le texte a été proposée. Les méthodes disponibles exploitent divers aspect de l'écrit. Le premier ensemble de techniques modifie l'espacement des mots. Une technique qui a été utilisé par Margaret Tchatcher pour suivre les fuites de documents confidentiels dévoilés dans la presse. Des méthodes stéganographiques plus avancées utilisent une structure syntaxique et sémantique du texte comme support. Les méthodes introduites comprenaient le déplacement des signes de ponctuation, l'ordre des mots ou des modifications du choix des synonymes, auxquels on pouvait attribuer une certaine signification.

1.3.2.2 Stéganographie basée sur l'ADN

L'ADN (Acide Désoxyribonucléique) stéganographie également connu sous le nom de génome stéganographique, permet de combiner la technologie ADN avec le cryp-

tage des données. Cette technique découverte en 1999, dissimule un message au sein d'un grand nombre d'objets similaires [26]. Il réduit un message à la taille d'un micro-point photographique. La première étape de cette technique consiste à cacher le message dans l'ADN. Puis la seconde étape consiste à cacher l'existence de l'échantillon d'ADN contenant le message en le réduisant à un petit point et en le mettant dans une lettre inoffensive. Ensuite cette dernière est expédiée au destinataire. Ce dernier utilise des produits biochimiques standards pour détecter et lire le message secret codé dans l'ADN.

Les explorations poussées sur l'usage de l'ADN en stéganographie révèlent qu'il peut être utilisé pour marquer secrètement des données ou des objets de valeur. De plus, des études ont montrés que l'ADN est un support attrayant pour le stockage de données en raison de très grandes quantités de données pouvant être stockées dans des volumes compacts. Il dépasse largement les capacités de stockage des supports électroniques, magnétiques et optiques. De ce fait, des études ont montré qu'un gramme d'ADN contient environ 10^{21} souches d'ADN, soit environ 10^{8} Téra octets d'informations. Par conséquent, quelques grammes d'ADN ont le potentiel de sauvegarder toutes les données stockées dans le monde.

1.3.2.3 Stéganographie basée sur l'espace disque

L'invention d'un système de fichiers stéganographiques par Enderson et al. a été une révélation [27]. Il est devenu évident que les informations peuvent être intégrées stéganographiquement, même dans des environnements informatiques isolés. Le principe fondamental de la préparation du stéganogramme est similaire aux encres invisibles. Lorsque la manière de rechercher le secret est connue, il est possible de révéler les fichiers cryptés et cachés à partir d'un disque. Le mécanisme repose sur le fait que les données chiffrées ressemblent à des bits aléatoires naturellement présents sur le disque. Ainsi, seule la possibilité d'extraire les vecteurs marquants les limites du fichier permet de localiser les fichiers cachés.

Ce type de stéganographie permet aux données de se cacher dans l'espace libre du disque dur. Parfois, les données secrètes sont marquées dans les canaux logiques par l'usage de champs inutilisés et redondants, ce qui n'est pas évident pour les cyber-experts et examinateurs médico-légaux . Par exemple, cela est possible grâce au Master Boot Record (MBR) des disques non bootables et l'espace disque libre provoqué par le désalignement du fichier du disque dur.

1.3.2.4 Stéganographie basée sur le trafic réseau

Actuellement, dans le monde du digital, il est facilement inimaginable que le support dans lequel sont intégrés des données secrètes, ne soient pas nécessairement une image numérique ou un fichier audio. Par ailleurs, il peut s'agir de tout autre type de fichier ou unité organisationnelle de données tel qu'un paquet ou une trame, obtenu naturellement dans les communications réseaux.

Par conséquent, à côté des types de stéganographie numérique mentionnés plus haut, actuellement un intérêt accru s'opère sur la stéganographie réseau, qui est la plus jeune branche de dissimulation d'informations. C'est un domaine en développement rapide qui propose de multiples méthodes de dissimulation d'informations exploitable dans divers types de réseaux. L'exploitation des protocoles sur les mêmes couches du modèle de référence Open System Interconnexion (OSI), est l'essence même de la stéganographie réseau. Cette famille de méthodes peut utiliser simultanément un ou plusieurs protocoles ou les relations entre eux, en s'appuyant sur les modifications de leurs propriétés intrinsèques pour incorporer les données secrètes.

La montée en puissance de la dissimulation d'informations sur le réseau a ainsi été soutenue par deux facteurs, la rendant supérieure par exemple à la stéganographie des fichiers multimédia. Tout d'abord, l'espace pour les données à transférer n'est pas limité car de nouvelles données peuvent être transmisses à la demande alors que la capacité d'un objet multimédia est limitée. Ensuite, incorporer des messages secrets au niveau réseau permet des fuites d'informations, même très lentes, pendant de longues périodes. Ces transmissions sont plus difficiles à analyser par des experts en analyse du crime numérique, car le trafic habituel est capturé et considéré comme normal. En conséquence, l'utilisation de telles données cachées éphémères rend les méthodes stéganographiques plus difficiles à détecter et à éliminer dans les communications réseaux.

Après avoir choisi un protocole ou un nombre de protocoles comme support pour les données secrètes, le choix est fait sur comment l'incorporation doit s'effectuer. La première possibilité consiste à injecter les informations secrètes dans l'Unité de Données de Protocole (PDU) . Cela peut être fait en modifiant les champs spécifiques du protocole ou en insérant le secret dans le champ de données appelé payload. De plus, il est possible de modifier le temps d'arrivée des PDU ou les opérations propres au protocole. Ces changements peuvent avoir un impact par exemple sur l'ordre des PDU, leur perte ou leur retard. Bien entendu, des méthodes hybrides qui utilisent à la fois la synchronisation des PDU et leur modification sont possibles.

Le prédécesseur des méthodes actuelles de stéganographie réseau était l'utilisation de différents champs de la pile de protocole TCP/IP comme support de données cachées. La majorité de ces méthodes se sont concentrées sur l'incorporation de messages dans les champs de protocoles inutilisés ou réservés pour transmettre les informations secrètes. Cependant, de nos jours, des méthodes plus sophistiquées et avancées sont développées, qui sont orientées vers des environnements ou des services spécifiques. Les solutions récentes exploitent :

– **Les services internet populaires** : par exemple skype [28], bitTorrent [29].

– **Les nouveaux protocoles réseaux** : tels que SCTP (Stream Control Transmission Protocol) [30].

– **Les nouvelles technologies de mise en réseau** : telles que les réseaux sans fils, par exemple les réseaux de 4^e génération (LTE) [31].

1.3.2.5 Stéganographie basée sur le logiciel et les circuits

Les données peuvent également être masquées en fonction de la disposition physique d'un support. L'arrangement en lui-même peut être une signature intégrée qui est unique au constructeur. Un exemple de ce type est la disposition distribuée du code dans un programme ou la disposition des circuits électroniques sur une carte. Ce type de marquage peut être utilisé pour identifier de manière unique l'origine d'une fabrication et ne peut être enlevé sans modification significative de l'empreinte [32].

Des techniques d'empreintes digitales ont été introduites pour appliquer des marques codées cryptographiquement aux conceptions numériques FPGA (Field Programmable Gate Array), afin de prendre en charge non seulement l'identification de l'origine de la carte mais aussi le client officiel. Cette approche est capable d'encoder de longs messages et d'être sécurisée contre des collusions malveillantes.

1.4 Stéganographie et Applications

1.4.1 Applications légales de la stéganographie

La stéganogrphie a son application dans la plupart des domaines courants tels que l'armée, la médecine, en entreprise et dans le multimédia où un canal secret peut être nécessaire pour des fins de sécurité. Les principaux domaines sont les suivants :

– **Le domaine militaire** : dans le domaine militaire et de la défense, le transfert sécurisé de message est l'objectif principal lors d'une communication distante. La stéganographie a été largement utilisée à cette fin depuis longtemps par des militaires [33]. La communication autorisée dans ce domaine est la base de leur système de communication. De plus, plusieurs couches supplémentaires sont également fournies en utilisant différents schémas de crypatge avant d'intégrer les données secrètes.

– **Le domaine du renseignement** : la communication dans les agences de renseignement est en général couvert et a été largement utilisée pour le camouflage [34]. La National Security Agencies (NSA) par exemple, utilise la stéganographie pour transférer des messages confidentiels à l'extérieur de l'agence, et vice versa.

– **Le domaine médical** : la stéganographie médicale a été utilisée en médecine en employant des images médicales utilisées pour le diagnostic des maladies [35]. Les informations privées et cruciales diagnostiquées auprès du patient sont cachées à l'intérieur de ces images. En outre, les séquences d'ADN permettent de cacher des

informations secrètes [36]. Ce procédé permet de sécuriser les données du patient dans le but d'éviter toute fuite d'informations privées de la part d'une personne non autorisée.

- **Le domaine du multimédia** : dans les applications multimédia, la stéganographie est fréquemment utilisée pour le marquage des informations de copyright [4]. Ceci est fait pour vérifier et protéger les médias à l'instar des images, des films, les copies illégales, des infractions, etc.

- **En entreprise** : Les entreprises et les industries font face à la fuite de données, considérée comme menaçante et dangereuse pour leur fonctionnement au quotidien. La communication sécurisée utilisant la stéganographie est toujours appréciée pour le maintien de la confidentialité et de l'authenticité des échanges [37].

1.4.2 Cybercriminalité et stéganographie

En raison du progrès de la technologie, les techniques et outils stéganographiques attirent une immense attention de la part des cybercriminels, des groupes antisociaux et des terroristes, car ils fournissent un moyen facile de communiquer par dissimulation qui est bien souvent difficile à remarquer. Les données cachées dans les fichiers multimédia représentent une nouvelle opportunité pour des activités criminelles traditionnelles. Il exite de nombreux exemples d'utilisation de la stéganographie par des criminels et des terroristes pour la communication. Quelques exemples concrets qui attirent l'attention des médias publics sont :

- **L'attaque du 11 septembre (2001)** : plus particulièrement, après l'attaque du 11 septembre, l'utilisation de la stéganographie par les terroristes ont été mis en évidence. Des responsables américains et de nombreux articles ont affirmé qu'Al-Qaïda a utilisé la stéganographie pour planifier cette attaque.

- **Shadowz Brotherhood (2002)** : C'est une opération menée pour attraper des criminelles pratiquant la pornographie juvénile. Ces derniers utilisaient la stéganographie pour masquer les communications avec les adolescents [38] [39].

- **Juan Carlos, trafiquant de drogue colombien (2008)** : Il fut arrêté au Brésil, puis après enquête, il a été révélé plusieurs preuves d'utilisation de la stéganographie dans l'ordinateur du suspect où se trouvaient des messages vocaux et textuels cachés dans les images d'un personnage de dessin animé [40].

- **Attaques de touristes (2004)** : de nombreux articles affirmaient que les attaques contre les touristes dans les hôtels de Mumbai ont également été planifiées et réalisés en utilisant la stéganographie pour cacher des informations dans les mails et envoyés depuis le serveur Yahoo [41].

- **L'affaire d'espionnage russe (2010)** : Le FBI (Federal Bureau of Investigation)

a affirmé que dix espions russes aux Etats-Unies utilisaient la stéganographie et l'exfiltration des données secrètes étaient effectuées des États-Unis à Moscou [42].

– **Arrestation d'un membre d'Al-Quaïda (2012)** : Un membre de ce groupe a été arrêté à Berlin avec une puce composée de fichiers vidéo dans lesquels la stéganographie a été utilisée pour cacher 141 fichiers contenant les détails des futures opérations [43].

1.5 Stéganalyse

1.5.1 Description

Le but de la stéganographie est d'éviter les soupçons quant à la transmission d'un message caché. Si le soupçon est avéré, alors cet objectif se trouve être un échec. La stéganalyse est l'art de découvrir et de rendre inutiles de tels messages secrets [32]. Deux aspects de la stéganalyse impliquent la détection et la distorsion des messages embarqués. La détection nécessite que l'analyste observe diverses relations entre les combinaisons d'objets de couverture, du message, du stégo-média et d'outils de stéganographie. Les attaques de distorsion nécessitent que l'analyste manipule le stego-média pour rendre les informations intégrées inutiles ou les supprimer complètement. Essentiellement, les activités d'observation et de manipulation décrivent deux classifications d'attaques : les attaques passives et les attaques actives.

La dissimulation d'informations dans les médias numériques nécessite des modifications des propriétés des médias, ce qui peut introduire une certaine forme de dégradation ou des caractéristiques inhabituelles. La dégradation peut parfois devenir perceptible [44]. Ces caractéristiques peuvent agir comme des signatures diffusant l'existence du message embarqué et des outils de stéganographie utilisés, allant ainsi à l'encontre de la finalité de la stéganographie qui consiste à masquer l'existence d'un message. Les attaques passives de la stéganalyse impliquent la détection de ces caractéristiques et signatures.

Manipuler les médias numériques dans le but de désactiver ou de supprimer les messages intégrés est une tâche plus simple que de détecter les messages. Toute image peut être manipulée dans le but de détruire certaines informations cachées, plutôt que de rechercher l'existence du message intégré. La détection de l'existence d'un message caché permettra de gagner du temps dans l'activité de désactivation ou de suppression des messages en guidant l'analyste à traiter uniquement les médias contenant des informations cachées. Le but ici n'est pas de préconiser la suppression ou la désactivation des informations confidentielles valides des stégo-média, mais de souligner les vulnérabilités de ces méthodes car elles ne sont pas aussi robustes qu'on le prétend.

1.5.2 Attaques courantes de stéganalyse

La détection des signatures stéganographiques est basée sur les combinaisons des outils suivants : le support de couverture, le stégo-media, le message embarqué et l'algorithme de stéganographie connus par l'analyste. Les attaques associées sont de type stégo uniquement, média de couverture connu, message connu, stégo choisi, message choisi et stégo connu. Chaque attaque et sa description est résumée ainsi [32] :

1) Attaque par Stégo-média uniquement : Seul l'objet stégo est disponible pour l'analyse.

2) Attaque par média de couverture connue : L'objet de couverture "original" et l'objet stego sont tous deux disponibles.

3) Attaque par message connu : À un moment donné, l'attaquant peut connaître le message caché. Analyser le stégo-objet pour les modèles qui correspondent au message caché peut être bénéfique pour de futures attaques contre ce système. Même avec le message, cela peut être très difficile et peut même être considéré comme équivalent à l'attaque par stégo-média uniquement.

4) Attaque par stégo-média choisi : L'outil de stéganographie (algorithme) et le stégo-objet sont connus.

5) Attaque par message choisi : Le stéganalyste génère un stego-objet à partir d'un outil ou d'un algorithme de stéganographie et d'un message choisi. Le but dans cette attaque est de déterminer les modèles correspondants dans le stego-objet qui peuvent indiquer l'utilisation d'outils ou d'algorithmes de stéganographie spécifiques.

6) Attaque par stégo-média connu : L'algorithme de stéganographie (outil) est connu et les objets originaux et stégo sont disponibles.

1.6 Conclusion

Dans ce chapitre, nous avons abordé les concepts de base nécessaire à la maîtrise du domaine. Il s'agit du modèle général des communications secrètes et des métriques d'évaluation des performances de ces techniques. De plus, la classification des techniques de stéganographie a fait l'objet de toute une section, présentant les techniques d'antan et récentes basées sur du texte, des disques et périphériques de stockage, les protocoles et trafic réseau, par voie logicielle ou par agencement des circuits, ou tout autre code ou transmission représenté numériquement. Ceux-ci fournissent d'excellents transporteurs pour les informations à cacher. Enfin, les domaines d'application de la stéganograhie ainsi que les types d'attaques ont été présentés. Le chapitre suivant sera dédié à l'étude critique des méthodes de dissimulation du secret.

Chapitre 2

État de l'art du problème de détection du secret dans les médias de couverture

2.1 Introduction

Dans ce chapitre, nous effectuons une étude critique des méthodes de dissimulation dans le but de ressortir leur limite. Nous commençons par décrire les techniques de dissimulation basées sur la modification du support. Ensuite nous présentons les méthodes de stéganalyse permettant de détecter la présence du secret dans les supports modifiés. Enfin, nous étudions la nouvelle philosophie de dissimulation par sélection du support appelé coverless.

2.2 Stéganographie classique

Deux philosophies peuvent être utilisées au choix lors de la conception d'un schéma en stéganographie, comme illustré à la figure 2.1. La toute première philosophie consiste à modifier le support pour intégrer le secret. On distingue dans ce cas deux approches principales : les approches classiques et distribuées. Puis le deuxième courant de stéganographie qui ne modifie pas le support, pour éviter la détection en sélectionnant le support qui contient déjà le secret. Enfin la stéganalyse qui vise à rechercher les empreintes dans les supports afin de détecter la présence du secret. Notons que la stéganalyse influence grandement la conception de tout schéma stéganographique, car l'attaquant contrôle les communications à la recherche de tout message suspect. Le but est donc de réaliser des échanges indétectables en cas d'usage des méthodes courantes de stéganalyse.

Le principe des schémas classiques en stéganographie consiste à dissimuler un secret dans un unique média de couverture. Nous présenterons dans les sous sections suivantes, les techniques de dissimulation appliquées aux médias de couverture de type texte et image.

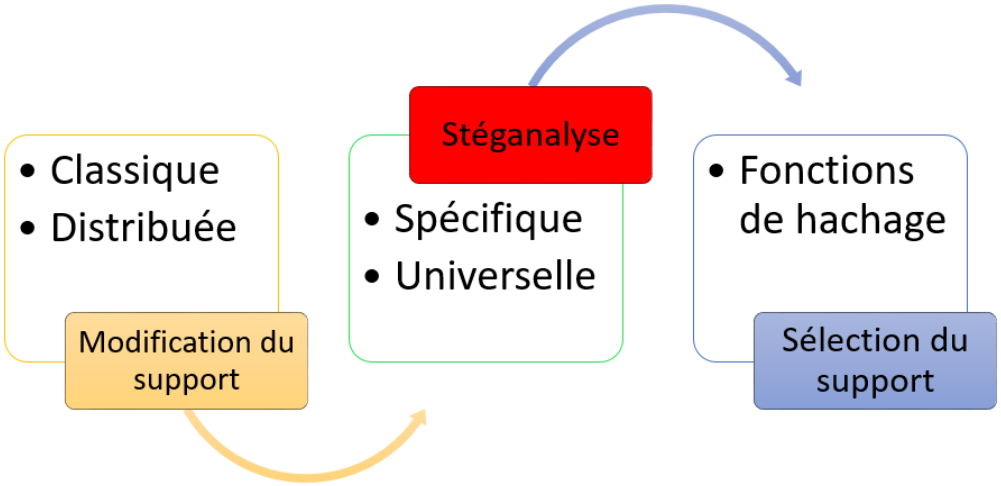


FIGURE 2.1 – Philosophies de conception d'un schéma stéganographique

2.2.1 Schémas de dissimilation dans les fichiers textes

2.2.1.1 Stéganographie des fichiers PDF basée sur le théorème des restes chinois

Ekodeck et Ndoundam [12] ont proposé une technique de communication sécrète appliquée aux fichiers PDF, considérée comme format d'échanges fixe pour représenter des documents d'une manière indépendante de l'application logicielle, du matériel ou du système d'exploitation utilisé. Ces fichiers sont très utilisés de nos jours et ce facteur rend possible leur utilisation comme média de couverture pour cacher les données confidentielles. Ce travail est basé sur celui de Lee et Tsai [45], dans lequel les données secrètes sont cachées entre les mots ou entre les caractères dans un fichier PDF, en utilisant le code ASCII de l'espace insécable nommé $A0$. Lee et Tsai ont découvert dans leur étude que le caractère insécable $A0$ est un caractère qui devient invisible lorsqu'il est inséré dans les chaines de caractères du texte. Ce caractère est pratiquement invisible dans le rendu visuel de tous les lecteurs de fichiers PDF. Cette observation est donc exploitée pour cacher des données.

Ce schéma exploite la structure lacunaire de l'équation neuronale récurrente, où il y a plus de 0 que de 1 dans une séquence $\{x^{\alpha_i}(n) : n \in \mathbb{N}\}$, générée par une équation neuronale récurrente, décrivant un cycle de longueur $3m - \alpha_i$ de la forme :

$$\underbrace{000\ldots0}_{2\alpha_i}\underbrace{100\ldots0}_{3m-\alpha_i}\underbrace{100\ldots0}_{3m-\alpha_i}\underbrace{100\ldots0}_{3m-\alpha_i}\ldots\underbrace{100\ldots0}_{3m-\alpha_i}\ldots$$

Ce schéma génère ainsi, les positions où les $A0$ seront insérés à partir d'un message secret donné et produit un fichier PDF au visuel pratiquement innocent. De plus le théorème des restes chinois employé ici permet de restreindre la taille des valeurs et ajoute aussi l'aléatoire. Dans un premier temps, le document PDF choisit comme média de couverture est parcouru du début à la fin dans le but de retirer tous les $A0$ présents

à l'intérieur. Puis l'émetteur et le récepteur se partagent une clé $m \in \mathbb{N}^*$. L'algorithme 2.1 présente l'insertion du secret dans un document PDF.

Algorithme 2.1: Algorithme d'insertion du secret dans le fichier PDF

Entrées : s : le secret en binaire ;
m : la clé secrète ;
f : le fichier PDF de couverture ;
Sorties : f' : le stégo fichier PDF ;

début
  Déterminer les nombres premiers $p_1, p_2, ..., p_t$, où t est le nombre de nombres premiers compris entre $2m$ et $3m$;
  Déterminer la taille des blocs : $k = \lfloor log_2(\prod_{i=1}^{t} p_i) \rfloor$;
  Le secret s est subdivisé en n blocks de taille k, stockés dans une matrice sp ;
  Chaque ligne i de sp est converti en une valeur décimale $dec[i]$;
  Pour chaque valeur $dec[i]$, calculer les restes $r[1,i], r[2,i], ..., r[t,i]$;
  Pour chaque reste $r[j,i]$, calculer les positions $pos[1,1], ..., pos[t,n]$;
  Trier le vecteur Pos dans l'ordre croissant.
  pour *chaque valeur de* $pos[i]$, $\quad 1 \leq i \leq (n \times t) - 1$ **faire**
    insérer un $A0$ à la position $pos[i]$ du fichier f ;
  fin
  Insérer deux $A0$ à la position $pos[n \times t]$ du même fichier pour marquer la fin du processus.
fin

Pour récupérer le secret qui a été inséré, il faut tout d'abord déterminer les positions où ont été insérés les $A0$ dans le fichier. Puis les restes qui ont générés ces positions doivent être déterminés. Et enfin nous obtenons le secret s en utilisant la clé sécrète m. Cela est décrit dans l'algorithme 2.2.

2.2.1.2 Stéganographie des fichiers compressés basée sur les codes LZW

Présentation

LZW est une technique de compression sans perte qui effectue un taux de compression élevé lorsque dans le fichier source, des motifs se répètent plusieurs fois. La méthode proposée par Zhi-Hui Wang et al. incorpore de manière réversible le message secret dans les codes de compression LZW en modifiant la valeur de ces codes en fonction du secret que l'on veut dissimuler [46]. La méthode exploite la relation entre les codes de compression et la taille du dictionnaire en garantissant que le receveur pourra retrouver les données sources après extraction des données secrètes sans aucune perte de données.

Algorithme 2.2: Algorithme d'extraction du secret dans le fichier PDF

Entrées : f' : le stégo fichier PDF ;
m : la clé secrète ;
Sorties : f' : le stégo fichier PDF ;

début
- Déterminer les nombres premiers $p_1, p_2, ..., p_t$ compris entre $2m$ et $3m$;
- Déterminer la taille des blocs k telle que : $k = \lfloor log_2(\prod_{i=1}^{t} p_i) \rfloor$;
- Calculer la taille des blocs contenu dans le fichier f telle que : $h = t \times p_t$;
- Récupérer les positions où le caractère A0 a été inséré ;
- Calculer les restes à partir de la table *pos* ;
- Déterminer chaque valeur en décimale des blocs dec[i] du secret s ;
- Transformer chaque valeur $dec[i]$ dans une séquence binaire $sp[i, j], \quad 1 \leq i \leq k : dec[i] = \underbrace{sp[i, 1]sp[i, 2]...sp[i, k]}_{kbits}$;
- Fusionner toutes les chaines de bits en une seule chaine pour obtenir le secret s : $s[(i-1)k + j] = sp[i, j], \quad 1 \leq i \leq n, \quad 1 \leq j \leq k$;

fin

Dans le fonctionnement normal de l'algorithme LZW, un dictionnaire est initialisé avec les 256 caractères du code ASCII : de 0 à 255. Les codes de sortie commencent avec une taille minimale de 9 bits. De manière générale, les codes sont représentés sur k bits lorsque la valeur des codes sont strictement inférieur à $n = 2^k - 1$. Lorsque le premier entier plus grand ou égal à $2^k - 1$ est rencontré, la séquence 1...1 (le bit 1, k fois) est produite et l'on continue le codage des entiers sur $k + 1$ bits. Ainsi, l'encodeur lit le fichier source séquentiellement et si la séquence est dans le dictionnaire alors produire en sortie le code correspondant au symbole précédent de la séquence. Et dans le cas contraire, la séquence est placée dans le dictionnaire à la prochaine positionnée libre. L'émetteur n'a pas besoin d'envoyer le dictionnaire ainsi construit au destinataire car lors de la phase de décodage, le même dictionnaire est construit de manière dynamique. Par conséquent, le décodeur retrouve le fichier source en convertissant les codes produits lors de l'encodage en leur symbole respectif sur la base du dictionnaire obtenu. Le tableau 2.1 présente un exemple de compression et de décompression avec du texte suivant : sddsddssddsdsddsddsd. Après la compression le fichier compressé contient les codes suivants : "115 100 100 256 258 259 256 261 257 256".

Algorithme d'insertion du secret

Cet algorithme modifie la valeur du code de compression de sortie à l'exception des 256 index du dictionnaire initial. Chaque code permet de cacher un bit secret. Avant qu'un mot ne soit inséré dans le dictionnaire, l'algorithme modifie la valeur du code LZW en fonction du bit secret. Si le bit secret est 0, la sortie est le code LZW non modifié. Lorsque le bit secret est 1, le code de sortie est la somme du code lzw et de la taille du dictionnaire. L'algorithme 2.3 illustre les étapes de dissimulation du secret

TABLE 2.1 – Phases de compression et décompression de LZW.

Compression			Décompression		
Input	Output	New item	Input	Output	New Item
sd	115	256=sd	115	s	
d	100	257=dd	100	d	256=sd
s	100	258=ds	100	d	257=dd
dd	256	259=sdd	256	sd	258=ds
ss	258	260=dss	258	ds	259=sdd
dds	259	261=sdds	259	sdd	260=dss
ds	256	262=sds	256	sd	261=sdds
ddsd	261	263=sddsd	261	sdds	262=sds
ds	257	264=dds	257	dd	263=sddsd
d	256		256	sd	264=dds

dans les codes lzw.

Algorithme 2.3: Algorithme d'insertion du secret dans les codes LZW

Entrées : Le fichier source et le message secret ;
Sorties : Les codes LZW ;

début
  Lire le premier caractère c_0 du fichier source. ;
  Initialiser $s = c_0$;
  si *s existe dans le dictionnaire* **alors**
    Affecter le symbole précédent dans s_p : $S_p = s$;
    Lire le prochain caractère c du fichier source ;
    $s = s||c$, où || désigne l'opération de concaténation ;
  sinon
    Lire le prochain bit secret b ;
    Lire le code C du mot S_p dans le dictionnaire ;
    si *b=1* **alors**
      $C = C + size$, avec size la taille courante du dictionnaire ;
    fin
    Produire C en sortie et Ajouter s au dictionnaire;
    Initialiser $s = c_s$, avec c_s le dernier caractère de s;
  fin
  Continuer à l'étape 4 pour parcourir tous les caractères du fichier source;
fin

Algorithme d'extraction du secret

Dans cette seconde phase, lorsque le code du fichier compressé est plus grand que la taille du dictionnaire, alors le bit secret extrait vaut 1 et dans le cas contraire, le bit secret vaut 0. De plus lorsque le bit secret vaut 1, le code LZW original est obtenu en effectuant la différence entre C' et size. Lorsque le bit secret est 0, le code LZW original est C'. L'algorithme 2.4 présente l'extraction du secret dans les codes LZW.

Algorithme 2.4: Algorithme d'extraction du secret dans les codes LZW

Entrées : Le fichier compressé contenant les codes LZW ;
Sorties : le fichier source et le message secret ;

1 **début**
2 Lire le premier code LZW C_0;
3 **si** $C_0 > Size$, *avec size la taille courante du dictionnaire* **alors**
4 Co = Co – size ;
5 Extraire le bit secret 1;
6 **sinon**
7 Extraire le bit secret 0;
8 **fin**
9 Produire en sortie le symbole s du code C_0 contenu dans le dictionnaire;
10 Lire le premier caractère C_s de s;
11 Affecter a O l'ancien code : $O = C_0$;
12 lire le prochain code LZW noté C;
13 **si** $C > size + 1$ **alors**
14 Calculer $C = C - size - 1$;
15 Extraire le bit secret 1 ;
16 **sinon**
17 Extraire le bit secret 0;
18 **fin**
19 **si** *C n'est pas dans le dictionnaire* **alors**
20 Déterminer $s = S_O||C_s$, avec S_O le symbole du code O dans le dictionnaire;
21 **sinon**
22 Déterminer $S = S_O$, S_O le symbole du code O dans le dictionnaire;
23 **fin**
24 Produire en sortie le mot S ;
25 $S_{new} = S_O||C_s$, avec S_{new} un nouveau symbole, S_O le symbole du code O dans le dictionnaire et C_s le premier caractère de s;
26 Ajouter S_{new} au dictionnaire ;
27 Effectuer l'opération $O = C$ puis continuer à l'étape 12.
28 **fin**

2.2.2 Schémas de dissimulation dans les images

Une image dans un ordinateur est un tableau de nombres qui représente l'intensité de la lumière à des points variés appelés pixels. Ces pixels constituent les données d'une image visuelle. Les pixels sont stockés sur 24 bits ou 8 bits. Un pixel de 24 bits peut avoir 16 777 216 (2^{24} combinaisons de couleurs possibles) et une telle image non compressée peut être très large en taille. Toute variation de couleurs d'un pixel est dérivé de trois couleurs de base : rouge, vert et bleu. Typiquement, dans un pixel de 24 bits d'une image, chaque couleur de base est codée sur un octet. Chaque octet représente l'intensité de couleur dans une plage de 0 à 255. Et la couleur du pixel est codée sur 6 nombres hexadécimaux. Différentes techniques sont mises en œuvre pour dissimuler des bits secrets dans les pixels que constitue l'image.

2.2.2.1 Stéganographie par substitution LSB

La technique LSBS (Least Significant bit Substitution) est la plus ancienne et la plus simple des techniques de stéganographie appliquée aux images [47]. Elle consiste à remplacer les bits les moins significatifs présents des pixels de l'image originale par des bits secrets. La Figure 2.2 illustre la technique LSB qui cache un bit par octet du pixel. Cette technique est capable de dégrader la qualité visuelle de l'image.

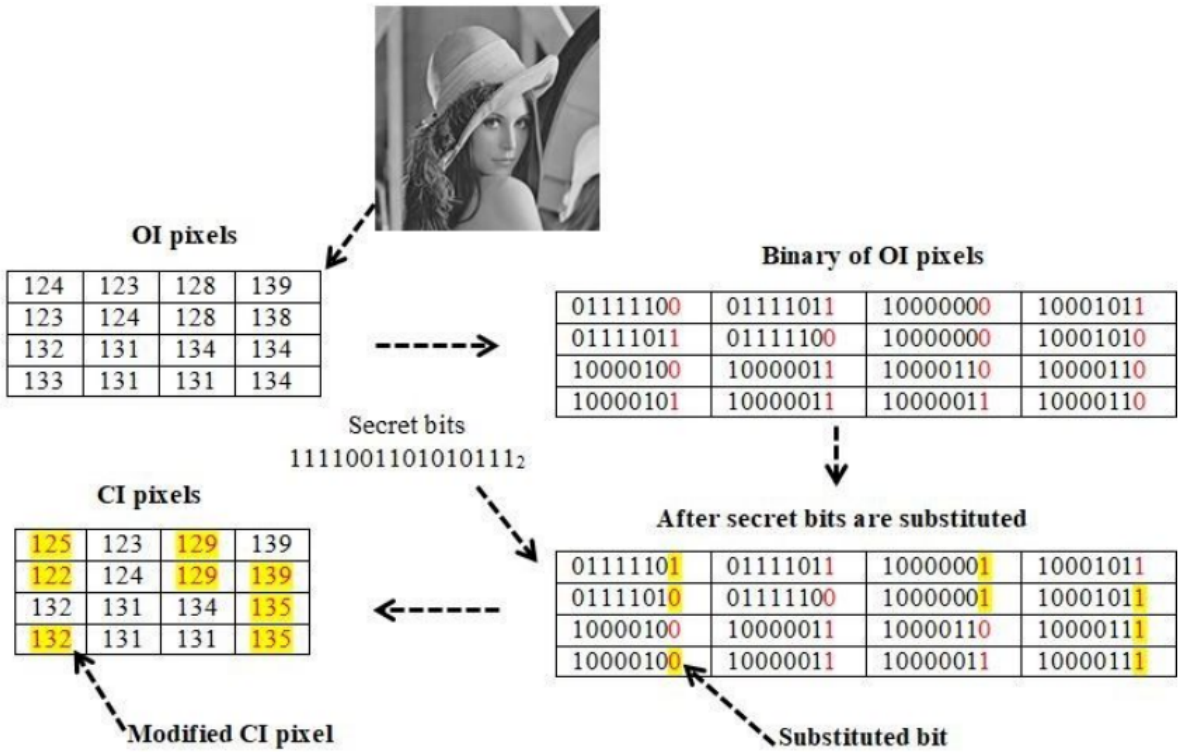


FIGURE 2.2 – Illustration de la technique de substitution LSB à un bit

2.2.2.2 Stéganographie par variation des index et substitution LSB

Wu et Hwang [48] proposent une technique améliorée de substitution LSB qui produit une meilleure qualité visuelle de la stégo image. Pendant la phase d'insertion du secret, trois pixels consécutifs sont considérés à partir de l'image de couverture et trois bits secrets sont insérés dans ces trois pixels. Considérons o_1, o_2 et o_3 les trois pixels consécutifs et b_1, b_2 et b_3 les trois bits secrets. o_1, o_2 et o_3 se représentent en binaire de la manière suivante :

$$o_{bin1} = o_1^{LSB8} o_1^{LSB7} o_1^{LSB6} o_1^{LSB5} o_1^{LSB4} o_1^{LSB3} o_1^{LSB2} o_1^{LSB1}$$

$$o_{bin2} = o_2^{LSB8} o_2^{LSB7} o_2^{LSB6} o_2^{LSB5} o_2^{LSB4} o_2^{LSB3} o_2^{LSB2} o_2^{LSB1}$$

$$o_{bin3} = o_3^{LSB8} o_3^{LSB7} o_3^{LSB6} o_3^{LSB5} o_3^{LSB4} o_3^{LSB3} o_3^{LSB2} o_3^{LSB1}$$

Puis en utilisant l'opération XOR les valeurs de A, B et C sont calculées comme suit :

$$A = o_1^{LSB1} \oplus o_1^{LSB2} \oplus o_2^{LSB1} \tag{2.1}$$

$$B = o_2^{LSB1} \oplus o_2^{LSB2} \oplus o_3^{LSB1} \tag{2.2}$$

$$A = o_3^{LSB1} \oplus o_3^{LSB2} \oplus o_1^{LSB1} \tag{2.3}$$

Ainsi, les pixels sont ajustés après comparaison des valeurs de A, B et C avec celles de b_1, b_2 et b_3. Du côté du receveur, les pixels de la stégo image sont notés c'_1, c'_2 et c'_3. L'extraction des bits secrets b_1, b_2 et b_3 s'obtiennent à partir des équations (2.1), (2.2) et (2.3) appliqués sur les pixels de la stégo image. Une illustration de la technique de Wu et Hwang [48] est présentée à la Figure 2.3.

2.3 Stéganographie distribuée

La stéganographie distribuée [49, 50] est une amélioration de la stéganographie classique qui vise à fragmenter un secret puis à le cacher dans plusieurs médias secrets, ce qui rend plus difficile la détection de l'intégralité du message secret. Ceci est appliqué lorsqu'il y a plusieurs expéditeurs indépendants et un seul destinataire. Ainsi le récepteur acquiert l'union d'entrées distinctes.

2.3.1 Partage du secret et la stéganographie

Le fait de protéger une donnée secrète à travers plusieurs personnes est nommé partage de secret (secret sharing). Les schémas de partage de secret nécessitent trois phases

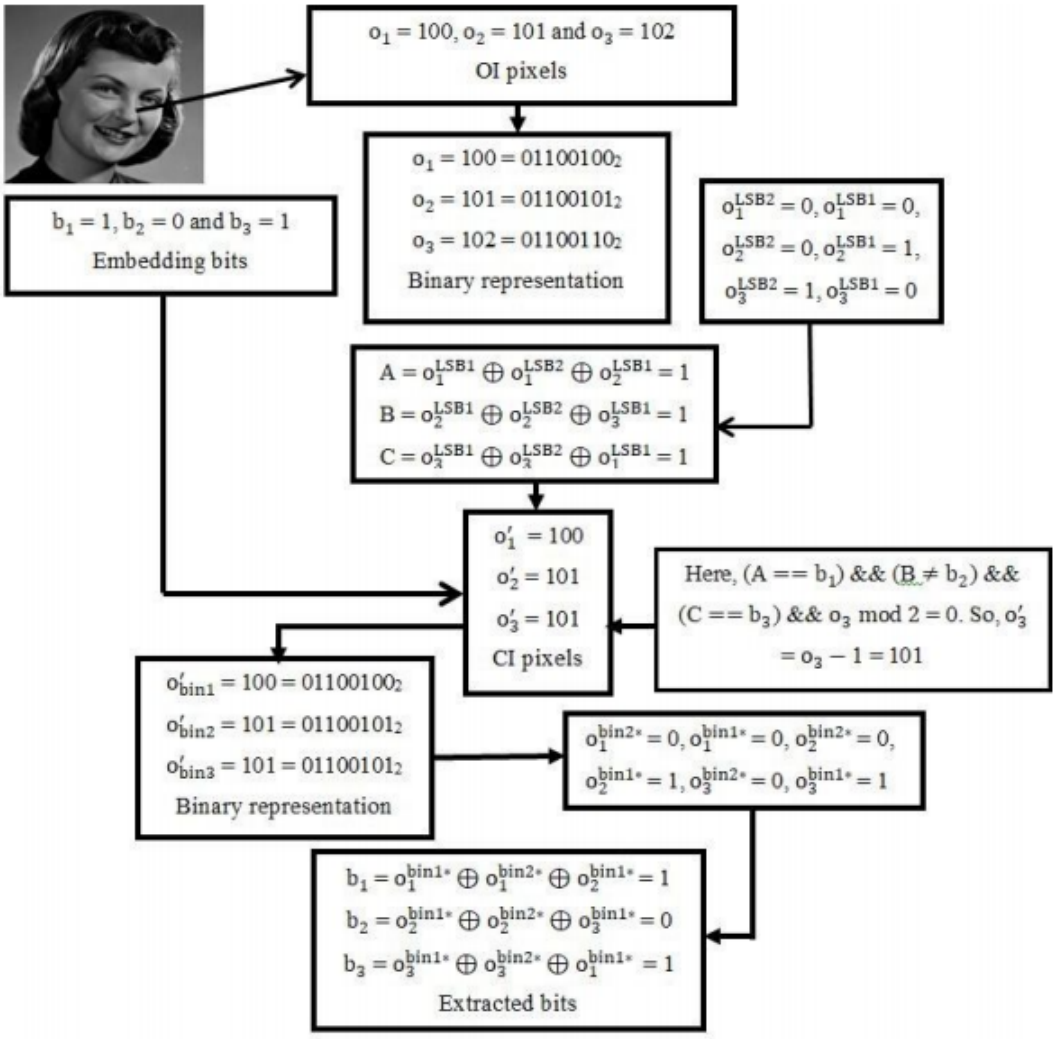


FIGURE 2.3 – Illustration de la technique de substitution LSB de Wu et Hwang

principales : la génération de la clé cible, la distribution des clés de partage aux participants et la reconstruction du secret. La clé cible peut ainsi être obtenue à partir d'un ensemble de n parts déterminé automatiquement par le système. Ce dernier distribue une clé de partage à chaque participant via un canal privé. Enfin, le système combine les différentes clés de partage pour accéder au système à partir de la clé cible obtenue. Le partage de secrets est appliqué dans des domaines critiques nécessitant un accès contrôlé par plusieurs utilisateurs, tels que le lancement de fusées, l'ouverture de coffres-forts bancaires, l'exactitude de la preuve des systèmes de vote électronique [51].

Des améliorations du partage de secret ont été proposées pour permettre l'accès à un nombre restreint de participants, appelé (k, n) secret sharing. La valeur k est connue sous le nom de seuil de partage secret et doit être inférieure ou égale à n. Comme contrainte, la reconstruction de la clé cible nécessite au moins k participants. Une des approches qui s'inspire de ce principe est le partage de secret basé sur le comptage [51], consistant

à générer les clés de partage en remplaçant à différentes positions de la clé secrète un ou deux bits zéro par le bit 1. Ainsi, la reconstruction de la clé cible est obtenue en additionnant bit à bit les clés de partage de même position, de sorte que le bit résultat est égal à 1 si la somme est supérieure ou égale à la valeur du seuil k et 0 sinon. La méthode de partage de secret basé sur le comptage est formulée suivant l'algorithme 2.5.

Algorithme 2.5: Algorithme de partage du secret

Entrées : k : le seuil ; n : le nombre de participant maximum ; TK : la clé cible à partager ;
Sorties : $share_1, share_2, ..., share_n$: les parties secrètes à partager.

1 **début**
2 Considérer les zéros présents dans la clé cible TK ;
3 Chaque zéro dans TK est changé en différente position pour générer les parties secrètes;
4 Sélectionner k parties secrètes ;
5 Considérer les k parties secrètes en parallèle;
6 Compter les bits à 1 de la position x sur les parties secrètes m_x;
7 **si** $m_x > k - 1$ **alors**
8 représenter ce bit à la position x par 1 ;
9 **sinon**
10 dans le cas contraire, représenter par un bit 0;
11 **fin**
12 Comparer les bits obtenus avec la clé cible TK : ;
13 **si** $TK[x] = m_x$ **alors**
14 les parties secrètes sont correctes, puis aller à l'étape 22 ;
15 **sinon**
16 changer une partie secrète dans l'ensemble des k parties, puis aller à l'étape 4;
17 **fin**
18 **si** *toutes les parties secrètes ont été sélectionnées* **alors**
19 Reporter : impossible de construire un partage de secret pour cette clé cible TK avec la valeur de k ;
20 **fin**
21 **fin**

A titre d'exemple, la génération des clés de partage pour une clé cible $TK = 10010010$ est présenté dans le Tableau 2.2.

En prenant $n = 8$ et $k = 4$ comme valeur seuil, les parties secrètes 1,2,3 et 4 sont prises en compte pour reconstituer le clé cible $TK = (10010010)_2 = (92)_{16}$. Cela est illustré dans le Tableau 2.3.

Cette technique nécessite moins de calculs comme point fort, cependant elle génère un

TABLE 2.2 – Génération des clés de partage par substitution des bits à zéro de la clé cible.

TK :	1 0 0 1 0 0 1 0	»	92 (Hexadecimal)
Share 1 :	1 $\underline{1}$ 0 1 0 0 1 0	»	D2
Share 2 :	1 0 $\underline{1}$ 1 0 0 1 0	»	B2
Share 3 :	1 0 0 1 $\underline{1}$ 0 1 0	»	9A
Share 4 :	1 0 0 1 0 $\underline{1}$ 1 0	»	96
Share 5 :	1 0 0 1 0 0 1 $\underline{1}$	»	93
Share 6 :	1 $\underline{1}$ $\underline{1}$ 1 0 0 1 0	»	F2
Share 7 :	1 $\underline{1}$ 0 1 $\underline{1}$ 0 1 0	»	DA
Share 8 :	1 $\underline{1}$ 0 1 0 $\underline{1}$ 1 0	»	D6
Share 9 :	1 $\underline{1}$ 0 1 0 0 1 $\underline{1}$	»	D3
Share 10 :	1 0 $\underline{1}$ 1 $\underline{1}$ 0 1 0	»	BA
Share 11 :	1 0 $\underline{1}$ 1 0 $\underline{1}$ 1 0	»	B6
Share 12 :	1 0 $\underline{1}$ 1 0 0 1 $\underline{1}$	»	B3
Share 13 :	1 0 0 1 $\underline{1}$ $\underline{1}$ 1 0	»	9E
Share 14 :	1 0 0 1 $\underline{1}$ 0 1 $\underline{1}$	»	9B
Share 15 :	1 0 0 1 0 $\underline{1}$ 1 $\underline{1}$	»	97

TABLE 2.3 – Reconstitution de la clé cible.

Share 1 :	1 $\underline{1}$ 0 1 0 0 1 0	»	D2
Share 2 :	1 0 $\underline{1}$ 1 0 0 1 0	»	B2
Share 3 :	1 0 0 1 $\underline{1}$ 0 1 0	»	9A
Share 4 :	1 0 0 1 0 $\underline{1}$ 1 0	»	96
Share 5 :	1 0 0 1 0 0 1 $\underline{1}$	»	93
Résultat du comptage	4 1 1 4 1 1 4 0		
Sortie	1 0 0 1 0 0 1 0	=	92

nombre réduit d'actions. Plusieurs optimisations ont été faites pour garantir la sécurité des clés partagées [52] ainsi que pour augmenter le nombre de partages générés [53, 54]. Ces schémas sont d'excellents outils utiles pour les protocoles cryptographiques. De plus, ils sont également utilisés en stéganographie pour masquer de manière distribuée les clés de partage de chaque participant. À cette fin, des méthodes sont fournies pour masquer la clé cible dans des supports secrets spécifiques tels que le texte [55, 56] ou les images [57, 58].

2.3.2 Distribution du secret dans les fichiers multimédias

Xin Liao et al. [59] ont proposé un modèle de stéganographie distribuée (voir Fig. 2.4), où n expéditeurs $p_1, p_2, ..., p_n$ souhaitent établir un canal secret avec un récepteur R. Chaque partie ne connaît que son message secret m_i alors que personne d'autre que le récepteur R ne peut récupérer la combinaison de ces messages secrets en la recevant via un canal public.

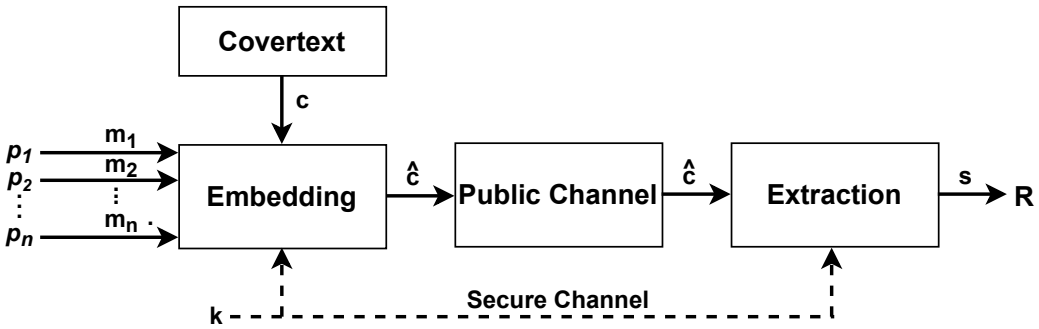


FIGURE 2.4 – Le modèle de la stéganographie distribuée

Avec l'avènement des technologies cloud, il est courant de nos jours que les utilisateurs cachent des informations secrètes dans une masse d'images et les stockent dans l'espace cloud. Ces images stégo sont ensuite partagées avec le destinataire pour extraire le secret. En général, ces algorithmes de dissimulation traitent le problème de la répartition du payload dans une séquence d'images pour éviter la détection. Cela explique l'émergence de stratégies d'intégration de la distribution du payload dans plusieurs images en fusionnant plusieurs fonctionnalités pour décrire la complexité de l'image [60]. D'autres stratégies récentes sont basées sur la complexité de la texture de l'image et la distribution de la distorsion comme indicateur de la capacité sécurisée de chaque image de couverture [61]. Ces stratégies sont appliquées sur des algorithmes stéganographiques à image unique et les expériences ont montré une meilleure résistance à la stéganalyse universelle par rapport aux méthodes existantes. Ces techniques sont certes meilleures que l'approche classique mais demeurent détectables [61].

William [62] illustre le processus de stéganographie distribuée en utilisant un bloc de petite taille. L'exemple considère des blocs de 4 octets sur le message secret "steganography is cool". Comme chaque caractère est représenté sur 1 octet, ainsi chaque suite de 4 caractères formeront un bloc secret. En prenant en compte les espaces, le message aura 5 blocs secrets stockés dans 5 images distinctes. La Figure 2.5 représente le découpage en blocs par image de couverture. Le challenge dans cette méthode réside dans l'extraction du secret, c'est à dire comment récupérer les blocs du message secret cachés dans les images. Ce problème implique de savoir combien de blocs au total devraient être récupérés de même que l'ordre de chaque bloc dans l'ensemble pour faciliter la reconstruction du message entier à partir des blocs obtenus. De plus, le destinataire devrait connaître comment sélectionner les stégo images contenant les blocs secrets. Chaque bloc stocké

dans une image aurait deux octets supplémentaires ajoutés à l'image. Le premier octet doit contenir les informations pour spécifier de quel bloc il s'agit (par exemple bloc 3 sur 9) et le second octet doit stocker le nombre total de blocs contenus dans le message. Puisque 1 octet peut stocker des nombres décimaux entre 0 et 255, cela nécessiterait de manière générale à cacher un message d'au plus de 256 bits. La taille du bloc serait déterminée par la taille du message d'origine divisée par 256. Les algorithmes de stéganographie basés sur les images peuvent être utilisés pour cacher les blocs de données, par exemple les algorithmes de substitution LSB [48, 63].

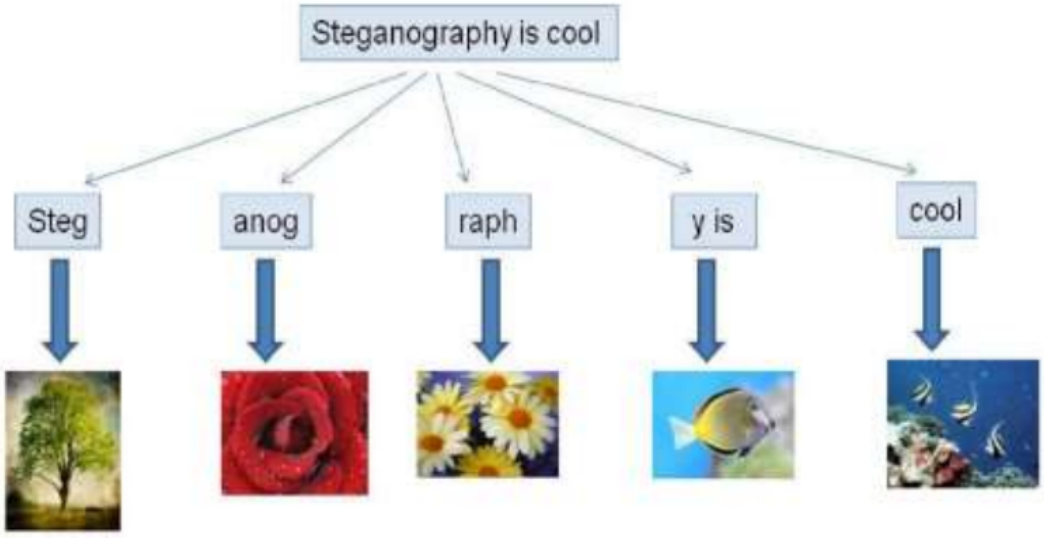


FIGURE 2.5 – Exemple de distribution du secret dans les images

D'autres techniques plus performantes ont été proposées pour non seulement augmenter la capacité d'embarquement des bits secrets mais aussi pour diminuer le niveau de dégradation de l'image. C'est le cas de la technique de distribution LSB des données dans de multiples images [64]. La méthode d'insertion est repartie entre les pixels en fonction de leur niveau d'intensité. D'autres techniques de stéganographie distribuée sur les images sont basées sur les blocs DCT (Discrete Cosine Transformation) [49] et la différence de paire de pixels [50].

Une caractéristique importante permettant d'évaluer la performance des techniques de dissimulation appliquées aux images s'appelle le PSNR (Peak Signal to Noise Ratio). Elle calcule la distorsion dans le stégo média en Décibel (dB). Une meilleure qualité du stégo média peut être obtenue lorsque le PSNR est élevé. A l'inverse, lorsque cette valeur est base, cela oriente la détection du message secret. Elle peut être mesurée à l'aide de l'équation suivante :

$$PSNR = 10 \times log_{10} \frac{d^2}{EQM} \tag{2.4}$$

Où d est la valeur maximum possible pour un pixel, dans le cas standard d'une image

codée sur 8-bits, $d = 255$. Et EQM, l'Erreur Quadratique Moyenne . Elle est définie pour 2 images I_o et I_s de taille $m \times n$ par la formule suivante :

$$EQM = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I_o(i,j) - I_s(i,j))^2 \tag{2.5}$$

Le PSNR est utile pour mesurer la proximité de l'image stéganographiée par rapport à l'image originale après insertion du message.

2.4 Stéganalyse des supports modifiés

La stéganalyse vise à rechercher la présence des messages secrets dissimulés dans les supports. Le stéganaliste contrôle le canal de transmission et analyse toutes les communications à la recherche des supports suspects. A terme, le but est de pouvoir détecter ou non, la présence des messages cachés. Nous présenterons les méthodes de détection des messages cachés à l'intérieur des supports de type texte et image.

2.4.1 Détection des messages cachés dans les fichiers textes

La stéganalyse du texte est le processus d'identification de la présence des messages secrets à l'intérieur d'un fichier texte. Et si possible, extraire et récupérer le message secret. En pratique, la stéganalyse du texte est une tâche compliquée, en raison de la grande variété de caractéristiques de texte numérique, d'approches d'intégration et généralement la faible distorsion du fichier de couverture après l'intégration du secret. Dans certains cas, la stéganalyse des fichiers textes est possible en raison du fait de l'incorporation des données qui modifient les statistiques du fichier de couverture. En d'autres termes l'existence des symboles d'intégration rend toujours le fichier de couverture différent du stégo fichier, bien que cela soit très souvent imperceptible à l'œil humain. Les attaques possibles peuvent être classifiées en trois types : les attaques visuelles, structurelles et statistiques.

Les attaques visuelles ou manipulations par les utilisateurs se réfèrent au facteur humain, généralement un lecteur qui peut percevoir visuellement les modifications contenues dans le stégo fichier. Ces modifications consistent en des changements syntaxiques, sémantiques, lexicales, etc. Supposons un attaquant qui a accès au stégo fichier et il suspecte qu'il existe des modifications non conventionnelles à l'intérieur du stégo fichier. Ainsi, il peut intentionnellement supprimer, insérer ou réordonner les mots ou les caractères [65, 66]. En pratique, ces manipulations détruisent le message secret.

En ce qui concerne les attaques structurelles, les modifications s'effectuent au niveau de la structure du stégo fichier. Dans certains cas, l'attaquant est capable de changer le formatage (couleur, police, etc.) ou même l'encodage des caractères (ASCII, Unicode, UTF-8, etc.) du stégo fichier [12, 45, 67]. Ce qui entraîne généralement la destruction du message secret.

Pour ce qui est des attaques statistiques, le but est de rechercher les possibilités de trouver un message secret en considérant le nombre de mots, d'espaces ou de caractères spécifiques dans le stégo fichier. Ce type d'attaque exploite la connaissance de la technique stéganographique utilisée pour décoder le message secret [68, 69].

2.4.2 Détection des messages cachés dans les images

La sécurité intégrée est d'une importance capitale lors de la transmission des données. Par conséquent, pour toute technique stéganographique, le succès réside avant tout dans la capacité à résister aux attaques. Lorsque des méthodes de stéganographie sont proposées et utilisées au quotidien pour des communications secrètes, des adversaires cherchent des empreintes dans les supports. Le stéganaliste contrôle le canal de transmission. Il analyse toutes les communications à la recherche des supports suspects. Le système steganographique est considéré cassé si la méthode de stéganalyse peut décider qu'il s'agit d'un support modifié. Les méthodes de détection sur les images sont classées en deux catégories : les méthodes de détection spécifiques et les universelles.

2.4.2.1 Méthodes de détection spécifiques

Les méthodes spécifiques utilisent un algorithme particulier pour la détection. La précision de détection est plus élevée. Les tests statistiques [47] sont généralement utilisés dans les méthodes spécifiques. Les tests les plus courant sont :

L'analyse PDH (Pixel Difference Histogram)
- consiste à analyser la différence entre l'image originale et l'image modifiée à l'aide d'un histogramme
- construire les paires de valeurs de pixels : $(p_i, p_{i+1}), p_i \in [0, 255]$
- faire la différence de chaque pair de pixel : $d_j = p_i - p_{i+1}, d_i \in [-255, 255]$
- calculer la fréquence de chaque d_j.
- tracer un graphique avec la fréquence de d_j sur l'axe des ordonnées et les valeurs d_j distinctes sur l'axe des abscisses
- si les courbes se chevauchent alors il n'y a aucune preuve de données cachées.
- Cependant, une courbe en zigzag avec une séparation claire avec la courbe de l'image originale expose la technique.

L'analyse du plan des bits :
- l'analyse capture la présence d'informations intégrées à partir des LSB du pixels
- un plan de bits est un ensemble de bits correspondant à une position de bit donnée dans chacun des pixels représentant l'image
- construire un plan de bit pour chaque position de pixel : $p_i = O_1, ..., O_i, ..., O_8$, avec $1 \leq i \leq 8$
- La caractéristique structurelle des pixels peut être modifiée en cas de changement de l'un des plans de bits
- l'analyse des plans de bits individuels des deux images conduit à exposer la présence d'informations secrètes.

2.4.2.2 Méthodes de détection universelles

Ces méthodes détectent indépendamment de l'algorithme de dissimulation utilisé. Se généralise aux algorithmes de dissimulation nouveaux ou inconnus. Les méthodes universelles sont classées en deux sous catégories :

La stéganalyse aveugle [70]
- Entrée : les supports originaux pour l'entrainement et l'extraction des caractéristiques
- Aucune connaissance des algorithmes de dissimulation utilisé dans les supports
- Sortie : détecte les supports originaux et modifiés.

La stéganalyse semi-aveugle [71] :
- Entrée : les supports originaux et modifiés pour l'entrainement et l'extraction des caractéristiques
- Les supports modifiés contiennent des messages secrets obtenus à partir des algorithmes de dissimulation
- Aucune connaissance des algorithmes de dissimulation utilisé dans les supports
- Sortie : détecte les supports originaux et modifiés.

Les méthodes de détection marchent en général lorsque la comparaison des supports originaux et supports stéganographiés sont différents. Les récentes méthodes considèrent que ces deux supports sont identiques. Car le secret est déja intégré dans le support et un mapping permet d'effectuer la correspondance entre le secret et les blocs du support. Ainsi, ces méthodes sont indétectables, puisqu'aucune trace ne peut révéler la présence du secret après analyse.

Il existe différents outils disponibles pour la stéganalyse des fichiers. Ces outils sont utilisés pour détecter la présence des données secrètes dans ces fichiers. La plupart de ces outils sont open source et librement téléchargeable. Le Tableau 2.4 présente en détail les outils de stéganalyse appliqués aux images.

2.5 Stéganographie par sélection du support ou coverless

Dans ce thèse, nous nous intéressons aux différentes techniques stéganographiques qui dissimulent le secret sans modifier le fichier de couverture. Ce travail s'inscrit dans ce courant méthodologique appelé coverless. Dans ces techniques, les fichiers avant dissimulation et après dissimulation du secret sont identiques. Les méthodes existantes considèrent que le secret à communiquer est déjà dans le support, ainsi ce dernier n'a plus à être altéré. L'approche utilisée dans ces techniques consiste à utiliser les fonctions de hachage pour faire la correspondance entre le secret et le contenu du fichier, ce qui rend ces méthodes vulnérables aux attaques par altération du contenu du support. Par

TABLE 2.4 – Outils de détection utilisés en stéganalyse.

Outils	Source
VSL	http ://vsl.sourceforge.net
XstegSecret	http ://stegsecret.sourceforge.net
Stegdetect	https ://www.paladion.net/blogs/steganalysis
StegSpy V2.1	https ://www.blackhat.com/
StegExpose Image	https ://www.darknet.org.uk/
Steganabara	https ://www.openhub.net/p/steganabara
StegCracker	https ://github.com/Paradoxis/StegCracker
Stegextract	https ://github.com/evyatarmeged/stegextract
Stegbreak	https ://linux.die.net/man/1/stegbreak
2Mosaic	https ://www.petitcolas.net/watermarking/2mosaic/
Phototile	https ://phototileapp.com
StirMark Benchmark	https ://www.petitcolas.net/watermarking/stirmark
Stego Watch	https ://www.wetstonetech.com/products/#stegohunt
Stego Sute	https ://www.wetstonetech.com/products/#stegohunt
Hex Editor Neo	https ://www.hhdsoftware.com/free-hex-editor
Win Hex	https ://www.x-ways.net/winhex
Zsteg	https ://github.com/zed-0xff/zsteg

contre, ces méthodes sont invulnérables aux approches de stéganalyse et par conséquent indétectables par analyse du contenu des médias de couverture échangés. Les sections suivantes présenterons ces méthodes et leurs limites.

2.5.1 Méthode de Zhou et al.

2.5.1.1 Principe

Les auteurs proposent un schéma de stéganographie basé sur les images sans modification du support pour dissimuler les données secrètes [6]. Cela est illustré à la Figure 2.6. Une série d'images contenant déja le secret est choisie dans une base de données préalablement construite. Et ces images sont considérées comme des images stéganographiques. Dans un premier temps, les images sont collectées pour construire la base de données d'images. Ensuite ces images sont indexées en fonction de leur séquence de hachage générée par un algorithme robuste de hachage. Par la suite, les données secrètes sont transformées en chaines de bits puis divisées en un nombre précis de segments. Pour implémenter la dissimulation, les images dont les séquences de hach sont identiques aux segments du secret sont choisies dans la base de données comme stégo images. La méthode est résistante aux outils de stéganalyse.

2.5.1.2 Calcul de la séquence de hachage des images

La détermination du haché d'une image est présentée à la Figure 2.7. Les étapes sont les suivantes :

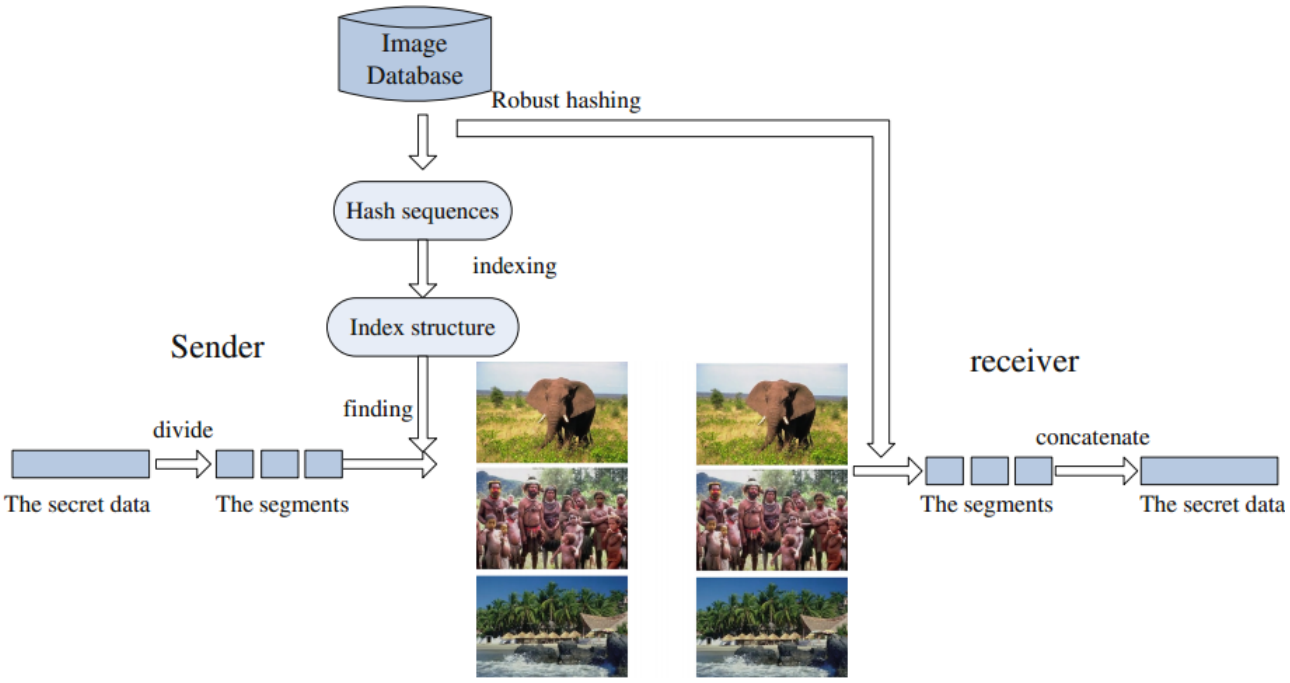


FIGURE 2.6 – Vue d'ensemble de la communication sécrète

1. L'image est transformée en niveau de gris et divisé en blocs de 3×3 ;
2. L'intensité moyenne de chaque bloc de pixels est calculée ;
3. Les valeurs de l'intensité des pixels sont concaténées dans un ordre en zig-zag pour former un vecteur et chaque valeur d'intensité est comparée à la valeur adjacente pour générer la séquence du haché.

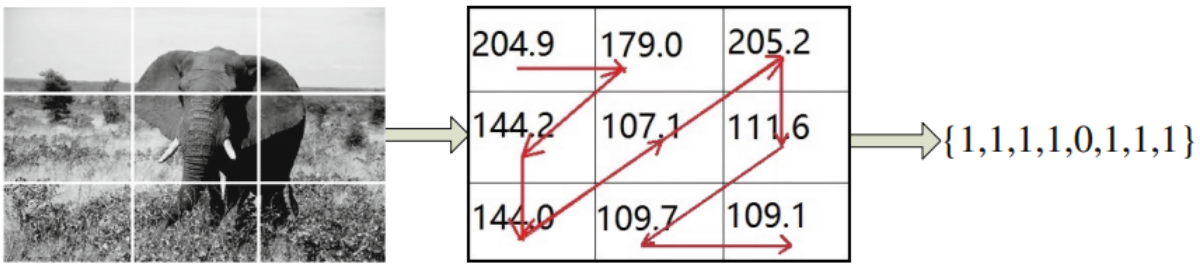


FIGURE 2.7 – Calcul de la séquence de hachage des images

Une séquence de hachage d'une image $h = h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8$ est déterminée en comparant les pixels I_i et I_{i+1} en utilisant la formule suivante :

$$h = \begin{cases} h_i = 1 & \text{si } I_i \geq I_{i+1}, \\ h_i = 0 & \text{sinon, avec } 1 \leq i \leq 8. \end{cases}$$

2.5.1.3 Construction d'une structure d'index inversée pour l'indexage d'images

Chaque image de la base de données est indexée avec son haché correspondant. Une table de recherche T est construite contenant tous les hachés sur 8 bits comme entrée de la table. Chaque entrée pointe vers une liste stockant des images dont le haché est identique. Cette table est illustrée dans la Figure 2.8.

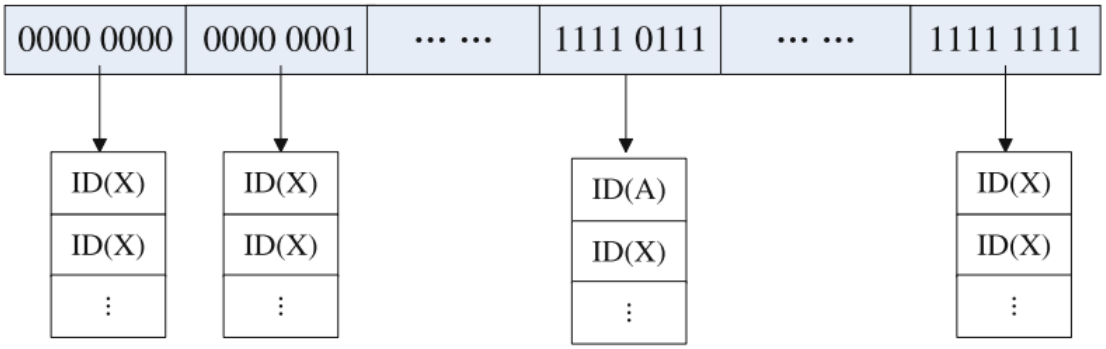


FIGURE 2.8 – Table de correspondance

2.5.1.4 Rechercher les images appropriées dans la structure d'index

le secret est décomposé en segments de 8 bits.Une recherche est effectuée pour trouver les images ayant pour haché, la valeur des segments du secret. Puis une image est choisie aléatoirement dans chaque cas. Ces images sont considérées comme des images stégo.

2.5.1.5 Transfert des images et extraction du secret

Les images sont transmises au destinataire une par une. Le récepteur les reçoit dans l'ordre. Cela permet la reconstruction dans l'ordre. Le récepteur calcule le haché de chaque image et les concatène dans l'ordre de réception des images.

2.5.1.6 Capacité d'embarquement

Une base de données 2^8 séquences de hachés est construite pour réaliser le mapping avec chaque segment du secret. Chaque segment de 8 bits correspond à un fichier transmis vers le destinataire. Ainsi, la capacité est de 8 bits par image transmise.

2.5.2 Méthode de Zhen et al.

2.5.2.1 Principe

Les auteurs proposent la conception du haché des images en utilisant les informations d'orientation des points de caractéristiques SIFT [7]. Cela est illustré à la Figure 2.9. Une base de données d'images locale est créée avec les valeurs du haché correspondant à ces images. Le message secret est divisé en segments de même taille que les séquences de hachés. Une série d'images est choisie dans la base de données d'images en matchant

les segments de secret avec les séquences de hachés. Finalement, les images sont envoyés au destinataire comme image stégo. Le destinataire extrait le secret en utilisant la méthode de calcul du haché. Une structure d'arbre quaternaire d'index inversée est conçue. Comparée aux méthodes de stéganographie classique, cette méthode ne modifie pas le contenu de l'image et résiste aux outils de stéganalyse.

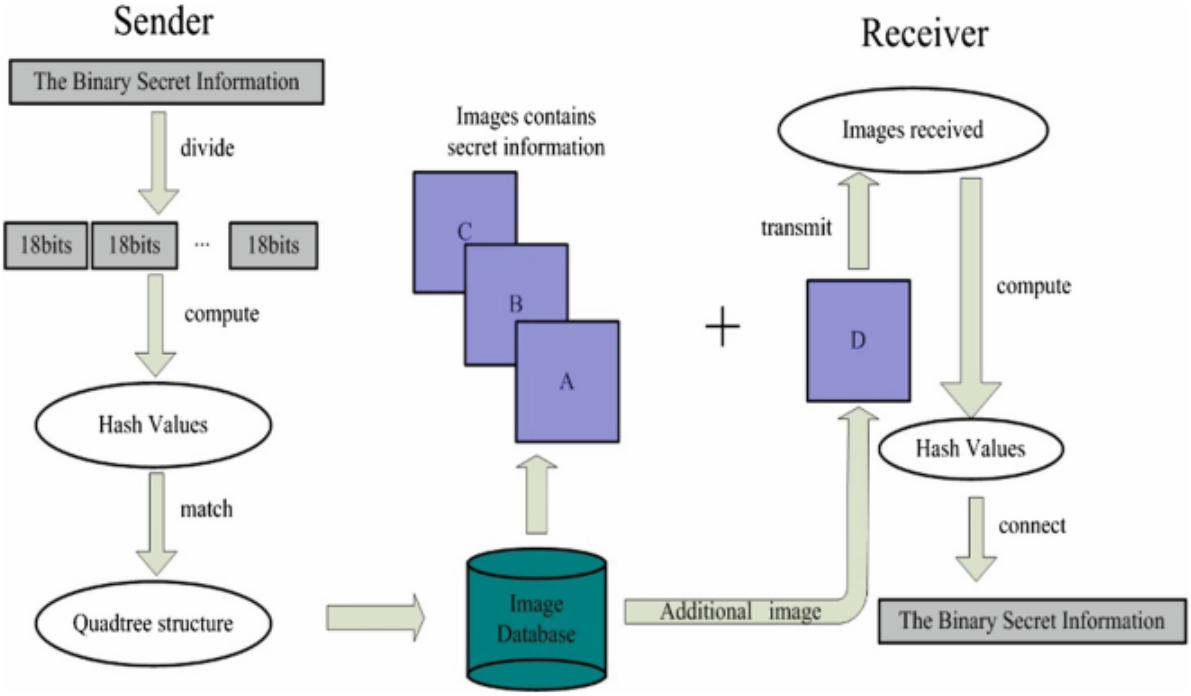


FIGURE 2.9 – Vue d'ensemble de la méthode de Zhen et al.

2.5.2.2 Méthode de calcul du haché de l'image

Le calcul du haché est présenté à la Figure 2.10. Considérer les images de taille 512×512, puis diviser chaque image en bloc de taille 3×3. Utiliser la stabilité SIFT pour calculer le haché des images. La direction du gradient des points stables sont accumulés pour former un histogramme. Obtenir la valeur max pour l'histogramme de chaque bloc. Si la valeur max est comprise entre 0 et 90 degré alors, la valeur du haché est 00 pour ce bloc d'images. Si elle est comprise entre 90 et 180 degré, la valeur est 01, ainsi de suite.

2.5.2.3 Structure d'arbre quaternaire de recherche

Le secret est divisé en blocs de 18 bits pour être associés aux images. Pour accélérer le processus de recherche d'index, un arbre quaternaire est utilisé. La hauteur de cet arbre est de 10 et chaque nœud de cet arbre possède quatre fils dont les valeurs sont $00, 01, 10, 11$ correspondant aux quatre différentes valeurs de haché d'un bloc d'images. Il existe une liste sur chaque noeud de type feuille pour stocker les informations sur les

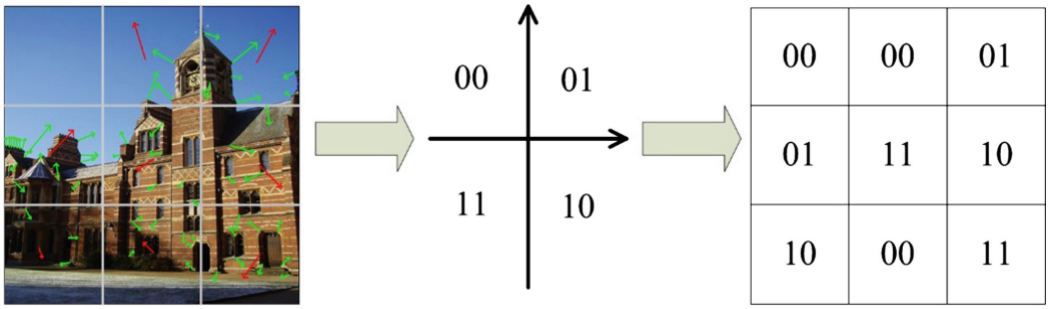


FIGURE 2.10 – Calcul de la séquence de hachage des images

images de même que la séquence de haché dans un certains ordre des blocs de l'image. Chaque feuille peut contenir plusieurs images. Cet arbre est illustrée dans la Figure 2.11.

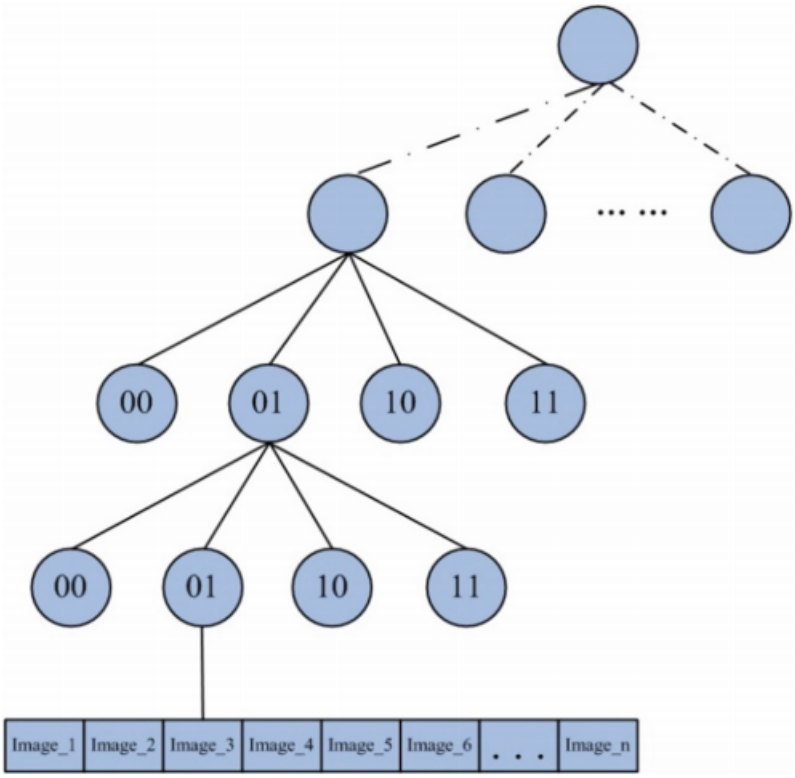


FIGURE 2.11 – Arbre quaternaire de recherche

2.5.2.4 Processus de dissimulation

Cela se fait en suivant les étapes suivantes :

1. Découper l'image en n segments de 18 bits
2. Utiliser la structure en arbre quaternaire pour déterminer la feuille de l'arbre correspondant au segment du secret. Chaque image sur la feuille peut être considérée comme image stégo.

3. Une image est sélectionnée de manière aléatoire dans une base de données d'images pour stocker l'ordre des blocs des images stégo et la taille l du secret. La dissimulation se fait à l'aide de la technique LSB. Les n images stégo et une image additionnelle sont envoyée vers le destinataire.

2.5.2.5 Processus d'extraction

Le récepteur reçoit séquentiellement toutes les images et extrait dans l'image additionnelle l'ordre des blocs des n images et la longueur du secret. Puis il calcul la valeur du haché de chaque image en appliquant l'ordre des blocs. Puis les valeurs de hachés sont connectées pour obtenir le message secret.

2.5.2.6 Capacité d'embarquement

Dans cette méthode, chaque image embarque 18 bits du secret. Soit le double de la capacité de la méthode de Zhou et al. [6].

2.5.3 Méthode de Wu et al.

2.5.3.1 Principe

Wu et al présentent un nouvelle méthode de stéganographie basée sur les images qui n'effectue aucune modification lors de la transmission de l'image secrète[8]. Cela est illustré à la Figure 2.12. L'approche consiste à utiliser un ensemble dupliqué d'images partiels comme image stégo, extrait d'une base de données d'images normales. Après avoir divisé chaque image de la base de données en un nombre de morceaux. Puis en indexant ces images basées sur les caractéristiques extraites de ces morceaux, il est question de chercher les copies partiels de l'image secrète dans la base de données pour obtenir les images stégo. Ainsi, chaque image partielle partage un ou plusieurs morceaux similaires avec le message secret. Du côté du récepteur, en utilisant les morceaux d'images stégo, l'approche est capable de retrouver approximativement les images secrètes.

2.5.3.2 Construction de l'index des fichiers images de la base de données

Pour faciliter la recherche des copies partiels dans l'approche stéganographique, les caractéristiques locales des morceaux d'images sont extraites et indexées selon leurs caractéristiques. Les trois étapes importantes sont détaillées ci-dessous.

Diviser l'image en plusieurs morceaux d'images : Chaque image de définition $I_w \times I_h$ est découpé en P_B morceaux, $P_B = \frac{I_w}{P_w} \times \frac{I_h}{P_h}$. Avec $P_w \times P_h$, la définition des morceaux de l'image. Le processus est illustré à la Figure 2.13.

Extraction de caractéristiques pour chaque morceau d'image : En trouvant les images partiellement dupliquées qui partagent une région assez similaire avec l'image secrète, il est possible de retrouver approximativement l'image secrète à partir des copies partielles et l'image extraite peut être assez similaire visuellement à l'image secrète originale. Cela est illustré à la Figure 2.14. Par conséquent, la

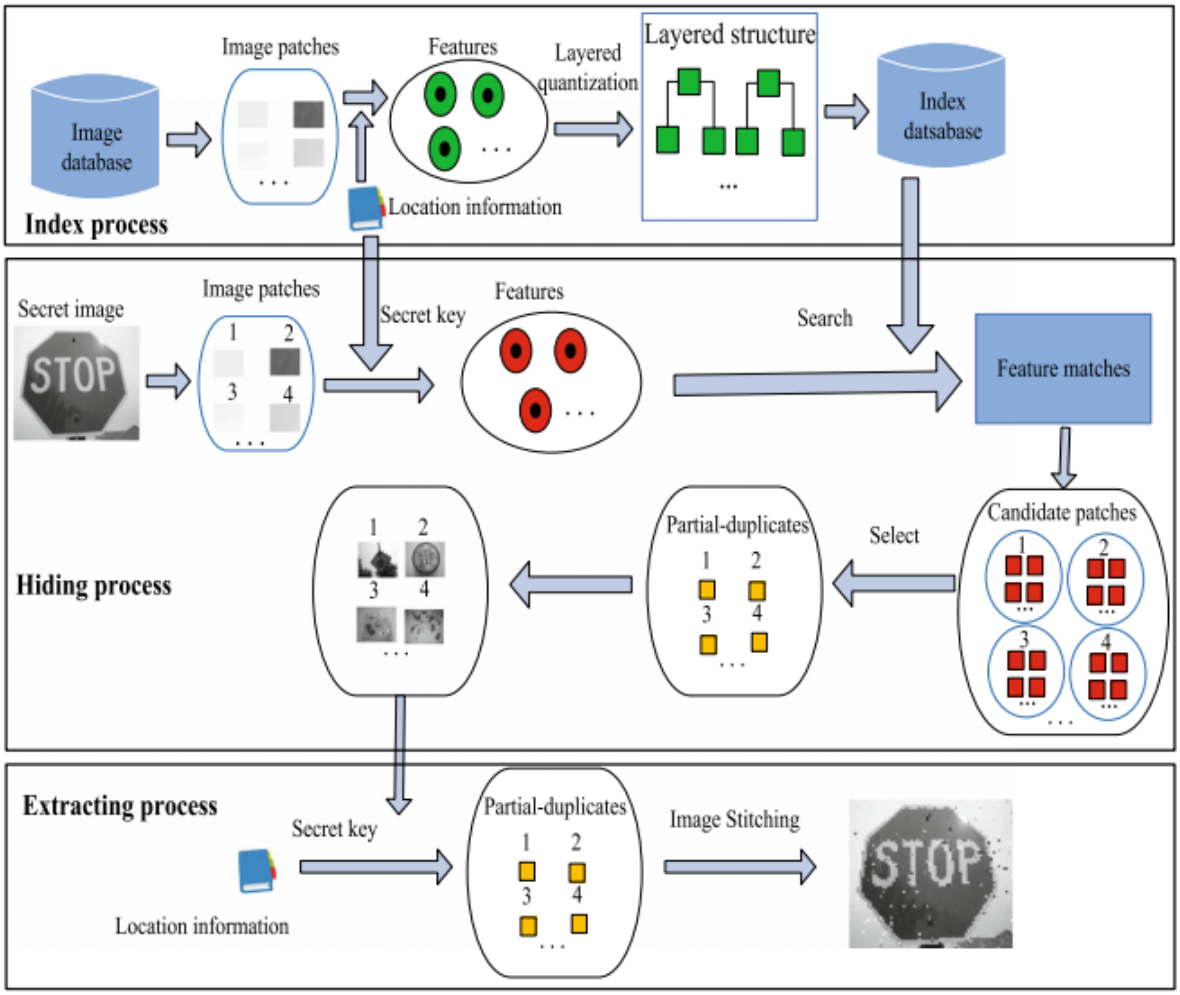


FIGURE 2.12 – Vue d'ensemble de la méthode de Whu et al.

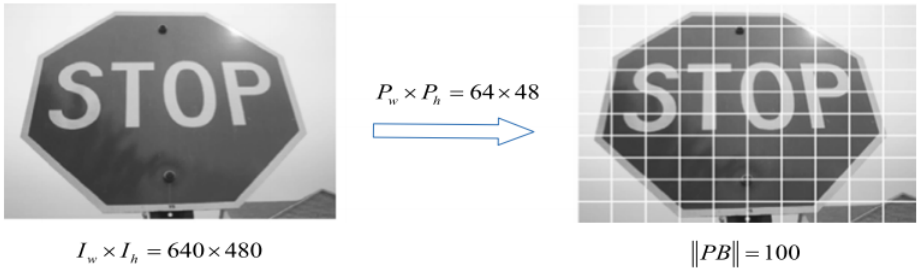


FIGURE 2.13 – Mécanisme de division de l'image

région de l'image doit être suffisamment décrite en capturant les caractéristiques de l'image. Pour cela, les caractéristiques de l'image sont extraites à partir d'un l'histogramme en niveau de gris des images.

Calcul du haché : Dans un premier temps, chaque morceau d'image est découpé en $m \times n$ régions. Dans un second temps, on évalue l'intensité de pixel minimum de chaque région, soit $m \times n$ valeurs. Puis l'étape suivante est de concaténer les

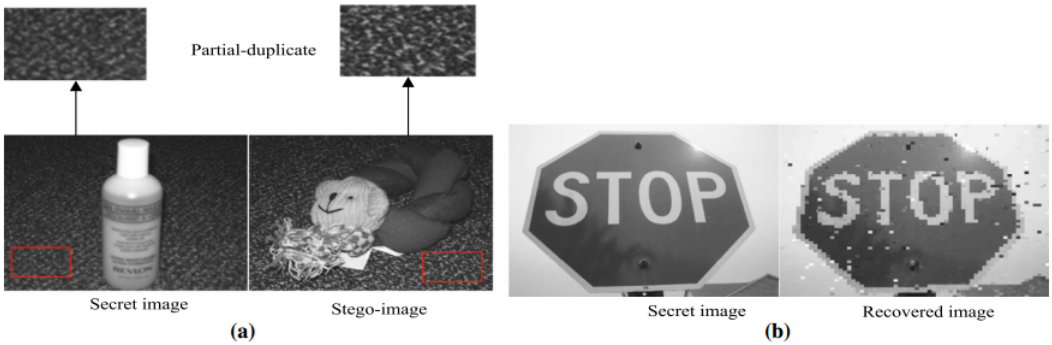


FIGURE 2.14 – Extraction des caractéristiques des images

valeurs en zig-zag pour obtenir un vecteur noté $V_1, V_2, ..., V_{m*n}$. Chaque valeur V_i est comparée à la valeur adjacente V_{i+1} pour générer une sequence de hachage notée : $h_1, h_2, ..., h_{m*n-1}$. Le processus de calcul du haché est donné à la Figure 2.15. Le haché obtenu est considéré comme l'index du morceau d'images à faire correspondre avec les parties de l'image secrète.

$$h = \begin{cases} h_i = 1 & \text{si } I_i \geq I_{i+1}, \\ h_i = 0 & \text{sinon, avec } 1 \leq i \leq m * n - 1. \end{cases}$$

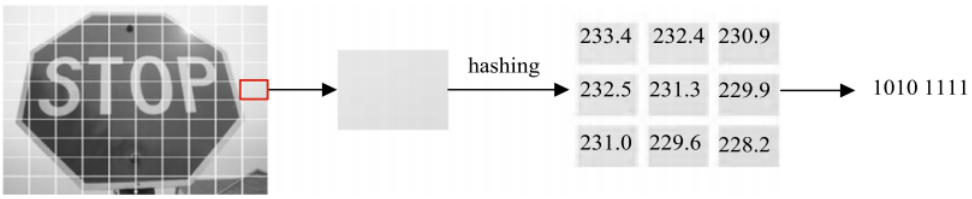


FIGURE 2.15 – Calcul du haché pour des régions de taille 3×3

2.5.3.3 Dissimulation de l'image secrète

Le processus de dissimulation de l'image secrète par la recherche d'images dupliquées partielles est divisée en quatre étapes :

1. Pour une image secrète donnée devant être masquée, similaire au processus d'image de la base de données, elle est divisée en un ensemble de morceaux (voir Fig 2.13). Puis la séquence de haché de chaque morceau est évaluée.
2. Définir une clé secrète, qui est partagée entre l'expéditeur et le destinataire au préalable. La clé secrète contient un ensemble de valeurs d'étiquette pour indiquer quelle partie d'une image partiellement dupliquée est utilisée pour cacher des informations secrètes.
3. Nous obtenons de nombreuses copies partielles de la base de données d'images à l'aide du haché des images. Dans l'approche, pour chaque valeur d'index, nous

pouvons obtenir un ensemble de copies partielles qui sont partiellement similaires au morceau d'image secrète. Nous pouvons sélectionner un parmi ces candidats.

4. Nous devons envoyer une image complète au récepteur, plutôt qu'un morceau d'image. Par conséquent, les copies partielles doivent être remplacées par leur image d'origine.

2.5.3.4 Extraction de l'image secrète

L'extraction d'image secrète pour le côté récepteur implique obtenir les informations de localisation et extraire l'image secrète dissimulée dans les stégo-images. Les étapes sont les suivantes :

1. Étant donné une clé secrète détenue du côté récepteur, les mêmes informations de localisation des morceaux d'images peuvent être obtenues.
2. Le côté récepteur obtient chaque copie partielle à partir de l'image originale à l'aide des informations de localisation.
3. Une zone vide dont la taille est la même que celle de l'image secrète est créée.
4. Enfin, le côté récepteur place ces copies dans la zone vide une par une pour générer une image, qui est similaire à l'image secrète.

2.5.3.5 Capacité d'embarquement

Dans notre approche, une image en niveaux de gris est utilisée comme image secrète, et la capacité est définie comme le nombre de bits cachés dans une stégo-image. Par conséquent, nous supposons qu'une taille d'image secrète est $I_w \times I_h$, et l'image secrète peut être masquée avec succès en utilisant m stégo-images. La capacité C est calculée en utilisant la formule suivante :

$$C = \frac{I_w \times I_h \times 8}{m}$$

L'image secrète est divisée en morceaux de taille $P_w \times P_h$ et que chaque morceau est dissimulée dans une image originale contenant une copie partielle du secret. Ainsi, en considérant des morceaux de taille 8×6 dans les expérimentations, la capacité des bits secrets par image stégo est de 384 bits.

2.5.4 Méthode d'Abdulsattar

2.5.4.1 Principe

Abdulsattar présente une méthode de stéganographie basée sur la non modification du support en utilisant une seule image pour transférer le secret en utilisant la décomposition Eigen [9]. Cela est illustré à la Figure 2.16. L'approche proposée effectue un mapping entre le code de hachage des blocs de l'image et les caractères du message secret. Le code de hachage est calculé en decomposant les blocs en 9 sous blocs et en comparant les valeurs eigen max sur la base de 4 arrangements. Pour accélérer le processus de

dissimulation, une table de recherche est construite pour stocker les codes de hachage pré-calculés associés aux positions des blocs. L'approche a 3 principaux paramètres : les différents blocs, l'arrangement des sous blocs et la taille des blocs. Les arrangements des sous blocs ont un léger impact sur le nombre de code de hachage unique et la résistance contre les attaques sur les images. Plus la taille des blocs augmente, plus la capacité diminue et la résilience contre les attaques sur les images est meilleur.

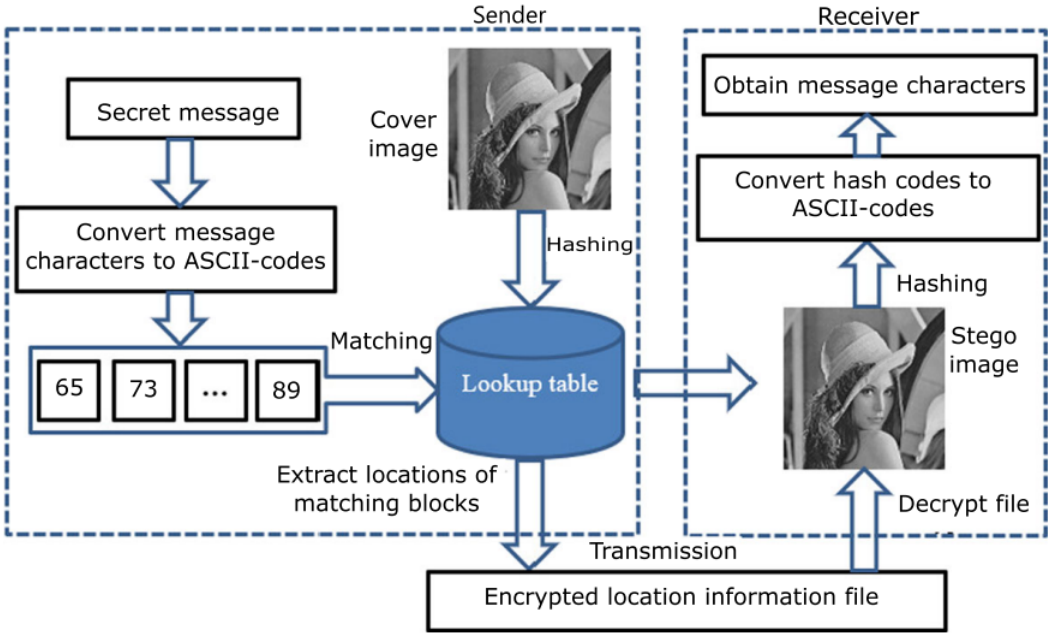


FIGURE 2.16 – Vue d'ensemble de la méthode d'Abdulsattar

2.5.4.2 Calcul du code de hachage

Une image de taille $M \times N$ est découpée en S_i blocs de taille $L \times L$. Puis chaque S_i bloc est ensuite décomposé en 9 sous blocs ss_j de taille $H \times H$ telle que : $H \times H = \frac{L}{3} \times \frac{L}{3}$. Cette opération est visible à travers la Figure 2.17.

Suivant le choix des quatre arrangements (voir Fig. 2.18), la valeur Eigen la plus élevée est comparée aux valeurs adjacentes pour déterminer la séquence du haché sur 8 bits.

2.5.4.3 Mise en place d'une table de correspondance

Plusieurs codes de hachage sur 8 bits peuvent être générés à partir de l'image. Le nombre de ces codes est identique au nombre de blocs d'images. Ces codes de hachage sont convertis en leurs codes ASCII à utiliser dans le processus de dissimulation. Au cours du processus d'intégration des informations, chaque caractère dans le message secret est converti en sa valeur de code ASCII et peut ensuite être mappé sur un ou plusieurs blocs d'image, qui ont le même code de hachage que le caractère du message. Pour accélérer la recherche d'un bloc d'image particulier correspondant à chaque caractère du

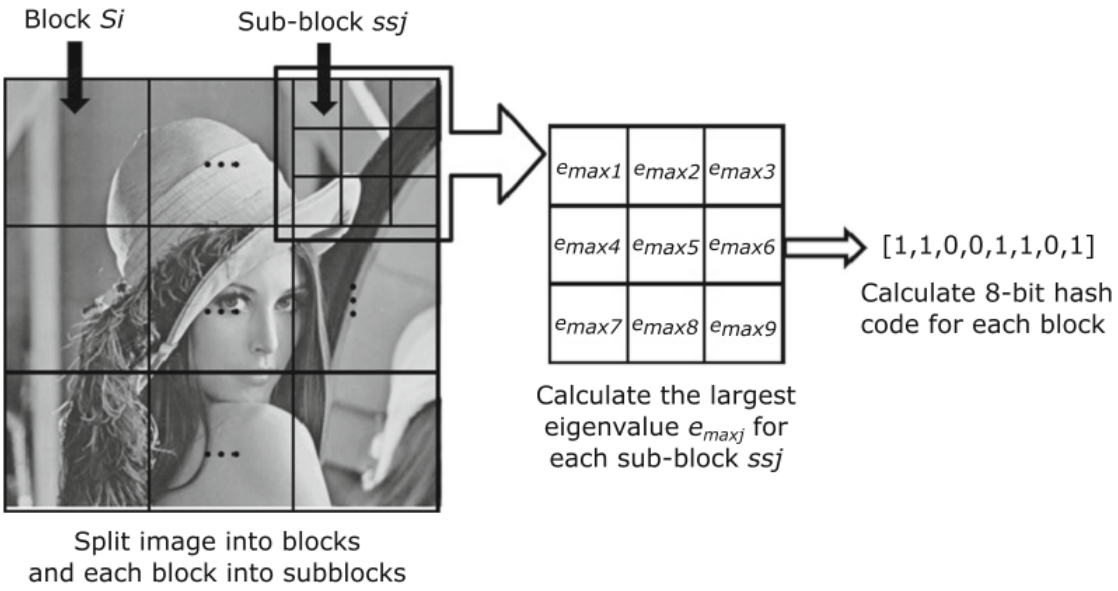


FIGURE 2.17 – Calcul du code de hachage

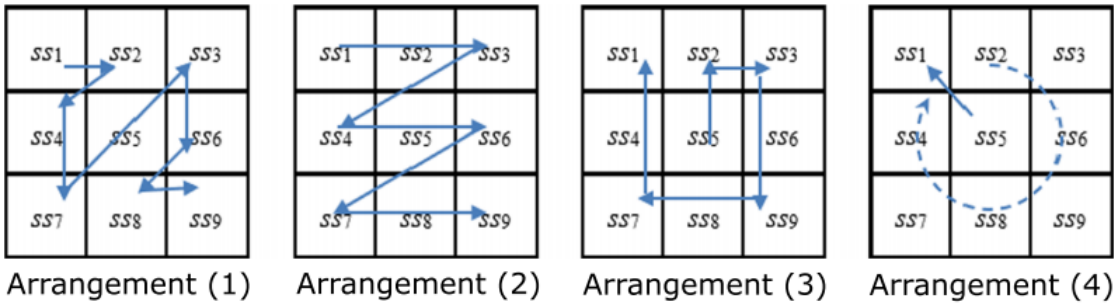


FIGURE 2.18 – Quatre arrangements pour générer le code de hachage

message, une table de correspondance est établit pour l'image de couverture. La table de recherche est utilisée pour stocker l'emplacement de chaque bloc d'image et son code de hachage correspondant après l'avoir converti en son code ASCII.

L'emplacement du bloc dans l'image est déterminé à partir de la coordonnée du premier (coin gauche supérieur) pixel dans le bloc. Il existe également un champ supplémentaire, qui sera utilisé comme un masque pour indiquer si le bloc courant a été utilisé ou non pour incorporer des données. La Figure 2.17 montre la disposition de la table de recherche.

2.5.4.4 Intégration d'informations

L'algorithme d'intégration d'informations est mis en œuvre du côté de l'expéditeur pour cacher les secret dans une image de couverture appropriée, qui sera envoyée au récepteur sous forme de stégo-image. Pour améliorer la sécurité de l'approche, l'expéditeur doit sélectionner une nouvelle image de couverture pour chaque nouveau message

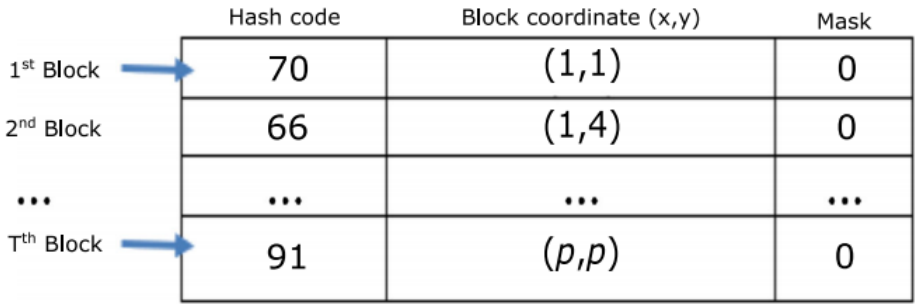


FIGURE 2.19 – Calcul du code de hachage

secret. La procédure d'intégration comprend les étapes suivantes :

1. L'image de couverture est redimensionnée à une taille prédéterminée ($M \times N$)
2. L'expéditeur calcule les codes de hachage à partir des blocs d'images et établit une table de recherche pour stocker les codes de hachage calculés (en valeur ASCII) avec les positions des blocs correspondant pour accélérer le processus de dissimulation.
3. Pendant le processus d'intégration, chaque caractère du message secret est d'abord converti au code ASCII correspondant, puis mis en correspondance avec le(s) bloc(s) d'image approprié(s), qui a le même code de hachage dans la table de recherche et les informations d'emplacement du bloc correspondant sont placées dans un fichier. Le champ de masque du bloc correspondant est défini sur un dans la table de recherche. Si le caractère du message correspond à plusieurs blocs, l'approche sélectionne le bloc d'image qui n'était pas utilisé auparavant en utilisant le champ de masque pour améliorer la sécurité de l'incorporation. Ainsi, si le même caractère apparaît plusieurs fois dans le message, alors il peut être mis en correspondance avec un bloc d'image différent à chaque fois.
4. Lorsque tous les caractères du message correspondent à leurs blocs d'image, le fichier d'informations de localisation est crypté à l'aide de l'algorithme de cryptage à clé publique Paillier. Le fichier crypté représente une clé secrète partagée entre les deux parties communicantes (expéditeur et destinataire) et l'image de couverture est envoyée au destinataire sous la forme d'une stégo-image.

2.5.4.5 Extraction d'informations

Pour récupérer les caractères de message cachés de la stégo-image, le récepteur effectue la procédure d'extraction d'informations suivante :

1. La stégo-image est d'abord redimensionnée à la taille prédéterminée ($M \times N$).
2. Le fichier d'informations de localisation est déchiffré pour obtenir les positions des blocs de l'image cible dans la stégo-image.
3. Le récepteur divise chaque bloc cible en 9 sous-blocs pour calculer les codes de hachage.

4. La plus grande valeur eigen de chaque sous-bloc est calculée et le code de hachage du bloc est obtenu en utilisant soit l'un des 4 arrangements choisit (voir Fig. 2.18) .
5. Le code de hachage binaire du bloc est converti en son code ASCII correspondant pour obtenir le caractère du message secret. Ensuite, les étapes 3 à 5 sont répétées pour obtenir le caractères restants.

2.5.4.6 Capacité d'embarquement

L'approche proposée peut stocker 8 bits d'information dans chaque bloc d'image. Par conséquent, le nombre de blocs d'images peut déterminer la capacité de dissimulation de l'approche proposée. Les meilleurs résultats sont obtenus avec des images en niveaux de gris de taille $M \times N = 512 \times 512$ et la taille de bloc $L \times L = 18 \times 18$. Soit une capacité de 6472 bits par image . La capacité de dissimulation de l'approche est calculée en utilisant la formule suivante :

$$C = 8 \times \frac{M}{L} \times \frac{N}{L}$$

2.5.5 Comparaison des différentes approches et limites

Les comparaisons seront menées sur la base des trois propriétés suivantes : la capacité, la robustesse et la sécurité des techniques stéganographiques.

2.5.5.1 Capacité

Les approches de stéganographie basées sur des supports non modifiés ont en général une capacité de bits secrets assez réduite comparée aux approches classiques qui modifient le support. C'est l'un des inconvénients majeurs. Les auteurs proposent au fil du temps des méthodes dont les capacités sont sans cesse croissantes. Les capacités d'embarquement des bits secrets des différentes méthodes sont illustrées dans le Tableau 2.5.

TABLE 2.5 – Capacité des différentes méthodes.

Méthodes	**Capacité (Bits secrets/Image)**	**# Images secret=1024 Bits**
Zhou et al.	8	128
Zhen et al.	18	56
Wu et al.	384	3
Abdulsattar et al.	6472	1

Pour chaque méthode, nous évaluons également le nombre de d'images stégo à envoyer au destinataire en fonction de la capacité des bits secrets insérés par image. En somme, nous avons les limites suivantes :

– La capacité de dissimulation du secret est basse pour la majorité des méthodes

- Le nombre de fichiers à transférer au destinataire est élevé sauf pour la méthode de Abdulsattar et al. qui envoit uniquement un seul fichier quelque soit la taille du secret à dissimuler
- La sélection des fichiers nécessite une grande base de données compatible

2.5.5.2 Robustesse

Au cours d'un processus de transmission, les images peuvent subir plusieurs attaques de traitement d'image qui pourraient endommager une partie de leur contenu. La robustesse est un paramètre essentiel pour juger de la performance des techniques de dissimulation d'informations. Il mesure à quel point le récepteur peut récupérer le message secret. Les attaques suivantes sont considérées dans les expérimentations :

- Remise à l'échelle (**Rescaling**) : permet de modifier la résolution de l'image, mais pas de manière significative. Ce paramètre est pris en compte à une valeur de 25 %.

- Changement de luminance (**luminance change**) et amélioration du contraste(**contrast enhancement**) : un changement d'éclairage entraînera une constante à ajouter à chaque pixel de l'image, et une amélioration du contraste provoquera la multiplication de chaque valeur de pixel par une constante. Pour le changement de luminance la valeur 10 est ajoutée à chaque intensité des pixels de l'image. Tandis que l'amélioration du contraste se fait en multipliant l'intensité des pixels de l'image par un facteur de 1,2.

- Ajout de bruit (**Noise adding**), compression JPEG (**JPEG compression**) : ces attaques peuvent modifier considérablement la corrélation d'intensité entre les pixels individuels en ajoutant du bruit. La compression JPEG et le bruit Guassien s'effectuent avec les paramètres par défaut.

Pour tester la robustesse des différentes méthodes aux attaques sur les images les unités de mesures suivantes peuvent être prise en compte :

- **Le taux de succès d'extraction des données secrètes** : ou Success rate of extraction (SRE)

$$S = 1 - \frac{\sum_{i=1}^{n} d(M_i, M_i')}{n \times l} \times 100 \tag{2.6}$$

  Où $d(x, y)$ désigne la distance de Hamming entre les vecteurs x et y, M_i et M_i' indiquent respectivement les données embarquées et les informations extraites de l'image i. l représente la capacité d'embarquement de chaque image et n taille de l'échantillon ou nombre d'images pris en compte dans l'expérimentation.

- **L'indice SSIM** : ou (Structure Similarity) permet d'évaluer la qualité visuelle de l'image secrète récupérée lors de l'extraction du secret. Ainsi, une bonne méthode de mesure de la qualité de l'image est nécessaire pour mesurer avec précision la similarité entre l'image secrète et l'image récupérée. L'indice SSIM se situe dans

la plage [-1,1], et quand il vaut 1, les deux images sont identiques.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.7}$$

Avec μ_x et μ_y, dénotent l'intensité moyenne de l'image originale et stégo,σ_x^2 et σ_y^2 la variance de l'image originale et stégo. σ_{xy} représente la covariance. C_1 et C_2 se réfèrent à des constantes.

– **Le taux de variation de l'indice SSIM** : ou SSIM index Change Rate (SCR) permet de vérifier la robustesse des différents moyens d'attaques d'images. Nous supposons que l'index SSIM entre l'image secrète et l'image de récupération avant d'attaquer l'image de récupération est S_{before}, et l'index SSIM entre l'image secrète et l'image de récupération après que l'image de récupération soit attaquée est S_{after}. Par conséquent, le SCR peut être calculé en utilisant la formule suivante :

$$SCR = \frac{S_{before} - S_{after}}{1} \times 100 \tag{2.8}$$

– **Le taux d'erreur sur les bits** : ou Bit Error Rate (BER), est utilisé pour déterminer si le message secret a été corrompu ou non lors de la transmission. Le BER est calculé à l'aide de l'équation suivante :

$$BER = \frac{\sum_{r=1}^{Z} xor(a_r, b_r)}{Z} \times 100 \tag{2.9}$$

Où a_r est le bit de message caché avant les attaques, b_r est le bit de message extrait après les attaques et Z est la longueur du message secret en bits.

Le Tableau 2.6 présente la taux d'erreur d'extraction des bits secrets (BER) des différentes méthodes présentées. A travers ce tableau nous observons que les différentes méthodes sont vulnérables aux attaques de traitement d'images. Les méthodes de Zhou et al et de Wu et al évaluent la robustesse sur la base de toutes les cinq opérations de traitement d'images. Tandis que Le méthode de Zhen et al évalue juste La compression JPEG et l'ajout du bruit gaussien. Après avoir intégrer le secret dans les images, chaque opération de traitement d'image est appliquée sur ces images pour pertuber la distribution des données binaires. Le but est de simuler une transmission du secret à travers un support au destinataire avec intégration du bruit, empêchant ainsi, une réception exacte du support.

Au regard des taux d'erreur d'extraction, le message secret dans la majorité des cas ne peut être extrait à 100%. De plus le pourcentage d'erreur est bien plus élevé dans le cas de l'ajout de bruit gaussien, atteignant un taux d'erreur de 60%. En somme, ces méthodes sont vulnérables aux attaques par distorsion (altération) du support.

2.5.5.3 Sécurité

Le Tableau 2.7 présente l'étude comparative des différentes méthodes en fonction de :

TABLE 2.6 – Taux d'erreur d'extraction des bits secrets en cas d'attaques

	Rescaling	Luminance change	Contrast enhancement	JPEG compression	Guassian noise adding
Zhou et al.[6]	0%	0%	0%	3%	2%
Zhen et al.[7]	-	-	-	5%	4%
Wu et al.[8]	1%	2%	1%	1%	60%
Abdulsattar[9]	-	-	-	-	26%

– **Nombre de fichiers stégo** : pour un secret de 1024 bits, nous déterminons le nombre de fichiers nécessaires à transférer au destinataire pour la transmission du secret. La méthode de Zhou et al. necessite l'envoi de 128 fichiers à raison de 8 bits par fichier, ce qui est vraiment élevé.

– **Type de communication** : qui peut être soit directe si les fichiers sont directement envoyés au destinataire ou indirecte s'il y a un intermédiaire entre l'émetteur et le destinataire. Vu que les différentes techniques envoient plusieurs fichiers au destinataire, cela peut attirer l'attention, surtout si la communication est directe. On Peut donc suspecter et analyser tout ce qui est transmis à la recherche d'empreintes stéganographiques.

– **Type de support** : qui détermine les formats de fichiers utilisés pour incorporer le secret. Ici, indépendamment de la méthode, le format de fichier est identique, c'est-à-dire les images. Ce qui peut attirer l'attention.

– **Corrélation entre le support et le secret** : qui détermine si la correspondance entre le secret et du support dépend du contenu du secret. En effet, toutes ces méthodes envoient les données confidentielles en tenant compte du contenu des supports. Il existe donc une corrélation forte entre le secret et les supports de couverture dans toutes ces méthodes. Ce qui explique leur vulnérabilité aux attaques d'altération du contenu des supports.

Après évaluation, il en ressort que :

– Il y a aucune précision sur le mécanisme de transfert du grand volume de fichiers entre les parties en communication ;

– La communication est directe entre l'émetteur et le destinataire et peut attirer l'attention ;

– Le mapping entre le secret et le fichier de couverture utilise le même support de type image ;

– La sélection du support dépend du secret à transférer.

TABLE 2.7 – Evaluation de la sécurité des différentes méthodes

	Nombre de fichiers stégo	**Type de Communication**	**Type de support**	**Support et secret**
Zhou et al.	128	Directe	Identique	Corrélation
Zhen et al.	56	Directe	Identique	Corrélation
Wu et al.	3	Directe	Identique	Corrélation
Abdulsattar et al.	1	Directe	Identique	Corrélation

2.6 Conclusion

Dans ce chapitre, nous avons présenté les deux grandes philosophies de conception d'un schéma en stéganographie et l'analyse des médias à l'aide des méthodes de détection par stéganalyse. Ainsi, nous avons présenté les schémas stéganographiques classiques et distribués qui modifient les médias de couvertures. Les schémas classiques ont la particularité de cacher un secret dans un seul type de média de couverture. Tandis que les schémas distribués dissimulent les données secrètes dans une multitude de média de couverture en fractionnant le secret en parties distinctes. Cette seconde approche a plus de considération car non seulement elle augmente la capacité d'embarquement mais aussi rend difficile la détection. Différents média de couverture ont été pris en compte dans la présentation des techniques de dissimulation. Ensuite des méthodes de détection du secret ont été présentées, ressortant ainsi les limites des techniques de dissimulation par modification du supports. Ces techniques de dissimulation ont un point commun, à savoir la présence des traces qui peuvent conduire à des analyses et pouvant révéler l'existence du canal secret établi par les personnes en communication. Pour terminer, nous avons présenté la stéganographie basée sur la sélection du support de couverture qui résiste aux méthodes de détection du secret. Toutefois, ces méthodes sont limitées par la capacité d'embarquement, la construction d'une base de données dépendant du secret, le nombre élevé de fichiers stégo à transférer au destinataire et le type de support manipulé. De plus, elles sont vulnérables à la distorsion ou opération d'altération du support. Dans la suite de ce travail, nous proposons une solution aux problèmes de construction de la base de données, d'altération du support et du type de support utilisé par une méthode de dissimulation distribuée dans l'environnement de stockage multicloud.

Chapitre 3

Stéganographie distribuée basée sur l'environnement de stockage multi-cloud

3.1 Introduction

La stéganographie classique ou traditionnelle vise à cacher un secret dans des médias de couverture tels que texte, image, audio, vidéo ou même dans les protocoles réseaux. Des recherches récentes ont amélioré cette approche appelée stéganographie distribuée en fragmentant le message secret et en incorporant chaque partie secrète dans un média de couverture distinct. L'intérêt majeur de cette approche est de rendre la détection du message secret extrêmement difficile. Cependant, ces modifications de fichiers laissent des empreintes qui peuvent révéler un canal secret à un attaquant. Notre contribution est un nouveau modèle de communication sécurisé en stéganographie transparent à tout attaquant et résistant à la détection et à l'extraction de secret. Deux propriétés contribuent à atteindre ces objectifs : les fichiers ne subissent aucune modification tandis que la diffusion du secret dans l'environnement de stockage multi-cloud permet de masquer l'existence du canal secret entre les parties communicantes. Les informations sont généralement cachées à l'intérieur du support de couverture. Dans ce travail, les médias secrets sont un pointeur vers l'information. Le fichier porte donc les informations sans être modifié et le seul moyen d'y accéder est d'avoir la clé.

3.2 Modèle du canal secret

3.2.1 Présentation

Le canal secret proposé est transparent aux communications secrètes entre les deux parties. La solution utilise l'espace de stockage cloud pour stocker les fichiers. Les fichiers téléchargés ne subissent aucune altération, l'information associée à chacun d'eux est leur ordre de classement dans une liste de fichiers. Ainsi, la sélection du fichier et le téléchargement vers le cloud dépendent du secret à partager avec le destinataire.

L'idée originale de la stéganographie a été proposée par Simmons [11]. L'idée de base

était de cacher des données secrètes à l'intérieur du média de couverture pour passer inaperçues. Les travaux connexes [12, 46, 47, 48] ont basés la conception de leurs schémas stéganographiques sur cette idée.

Dans ce travail, la contribution est la proposition d'un schéma stéganographique où les fichiers utilisés comme support véhiculent l'information sans être modifiés. Le média de couverture est un pointeur vers des données secrètes. Ces données secrètes se trouvent dans la clé. Cette clé est constituée des ensembles suivants : la liste des clouds et identifiants de connexion, les listes de fichiers et la base utilisée. L'expéditeur et le destinataire s'échangent cette clé avant d'initier leur communication secrète. L'échange de clés peut se faire lors d'une réunion physique ou à l'aide de communications cryptées. A titre d'exemple concret, un agent secret C est employé par son pays d'origine A. Lors d'une réunion physique dans l'agence gouvernementale A du pays, les responsables de l'agence remettent à l'agent secret une mémoire amovible qui contient la clé. Le pays A envoie l'agent secret dans le pays B. Lors de sa mission d'espionnage dans le pays B, l'agent secret envoie des informations confidentielles du pays B au pays A.

3.2.1.1 Vue d'ensemble du processus de dissimulation

Le modèle de communication proposé dans la Figure 3.1, montre l'expéditeur et le destinataire partageant des listes de documents identiques. L'expéditeur transcode le secret dans une base spécifique et les regroupe en k groupes de n valeurs, avec n le nombre de clouds utilisés. Pour chaque valeur d'un groupe donné (couleur vert, rouge ou jaune), les fichiers à cette valeur d'index sont envoyés vers le cloud correspondant : $c_0, c_1, ...cn - 1$. Le processus est répété pour chaque groupe ou séquence de n valeurs du secret. A chaque bloc est attribuée une nouvelle liste.

3.2.1.2 Vue d'ensemble du processus d'extraction du secret

A la Figure 3.2, le récepteur ayant le même accès à chaque cloud, les parcourt pour récupérer les fichiers sauvegardés. Par la suite, il reconstitue le secret à partir des positions des fichiers trouvées dans les listes. L'originalité de ce schéma réside dans l'incapacité de l'attaquant à établir un lien de communication entre l'émetteur et le récepteur.

3.2.1.3 De la praticabilité de la méthode proposée

La question que l'on se pose c'est de savoir comment l'utilisateur ayant accès à la plateforme de stéganographie peut effectuer les opérations sans percevoir la complexité des opérations à faire en arrière plan. Nous décrivons cela dans la figure 3.3, qui illustre l'architecture de la solution proposée. On y retrouve deux acteurs : l'émetteur et le récepteur ainsi que deux modules à savoir la dissimulation et l'extraction du secret. La réalisation de ces deux modules est conditionnée par l'exécution de deux opérations principales :

FIGURE 3.1 – Vue d'ensemble du processus de dissimulation du secret.

Création des dossiers et fichiers nécessaire à la dissimulation : Un ensemble de dossiers est créé contenant chacun des fichiers de formats divers et variés (texte, image, son et vidéos). En fonction du secret à dissimuler et de la valeur de la base, ces fichiers seront sélectionnés pour les envoyés vers les environnements de stockage du cloud.

Partage confidentiel de la liste de dossiers et de fichiers : Cette liste doit être identique pour le codage et le décodage du message secret. Raison pour laquelle elle doit être partagée entre l'émetteur et le destinataire. Le partage peut se faire par exemple lors d'une rencontre physique des deux parties communicantes avec la liste stockée dans une mémoire de masse.

Pour effectuer la dissimulation, l'émetteur doit entrer le message secret qu'il veut transmettre au destinataire. Ensuite entrer la clé qui est partagée entre l'émetteur et le destinataire constituée de la base B et des différents clouds intervenants. Puis les fonctions suivantes sont exécutées dans l'ordre pour dissimuler le secret :

1. **Conversion du texte en binaire** : cette a pour but à convertir le message secret en une séquence binaire. Chaque caractère est converti en code ASCII sur 8 bits. Puis les blocs de 8 bits sont concaténés pour obtenir la séquence binaire.

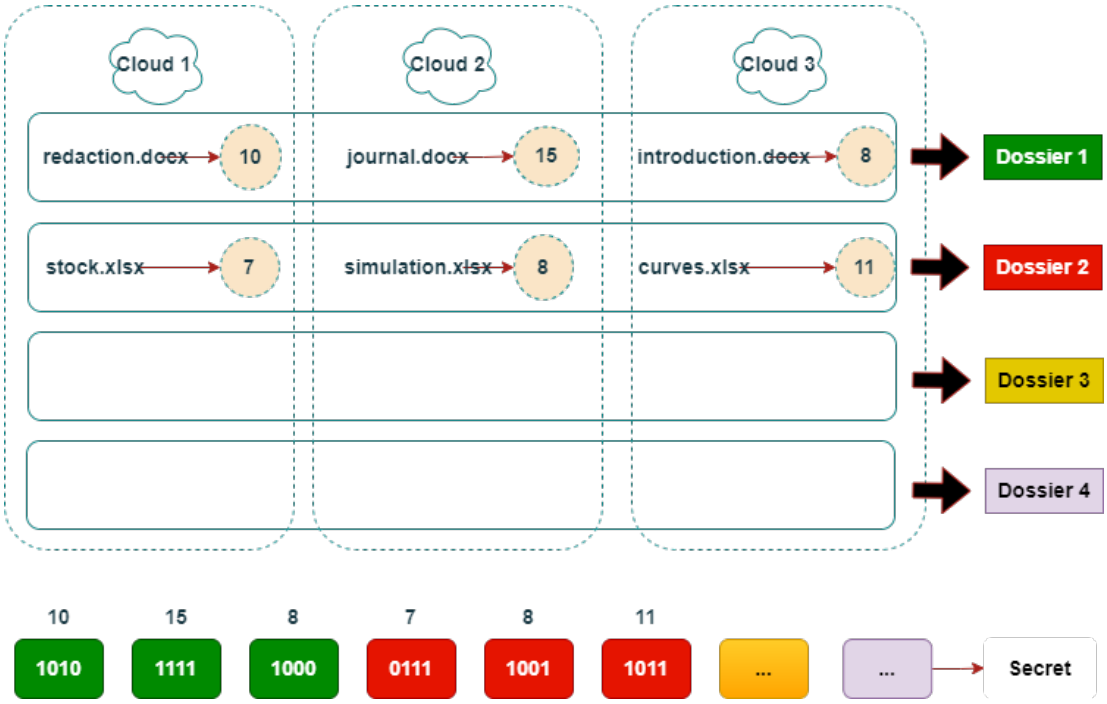


FIGURE 3.2 – Vue d'ensemble du processus de récupération du secret.

2. **Conversion de la séquence Binaire en base B** : cette fonction converti la séquence binaire en une suite de nombre compris entre 0 et B-1. Nous déterminons premièrement le nombre bits nécessaire pour représenter un nombre en base B. Cela s'obtient calculant : $k = log_2(B)$, sachant que B est une valeur choisie sous forme d'une puissance de 2. Ensuite décomposer la séquence binaire en blocs de k bits et enfin chaque bloc est converti en decimal pour avoir la représentation en base B. Chaque valeur en base B sera considéré par la suite comme un index.
3. **Mapping entre les index en base B et les fichiers** : Cela nécessite une liste de dossiers contenant des fichiers. La création de cette base de données est créée au préalable de sorte qu'a l'intérieur de chaque dossier les fichiers sont rangés par nom indexé de 0 à t, avec $t \geq B$. Ainsi, chaque nombre z_i représenté en base B ($0 \leq z_i \leq B-1$) permet de sélectionner un fichier qui a cet index dans un dossier précis. Notons que pour une séquence de n clouds, un ensemble de n fichiers est sélectionné dans un répertoire distinct. Puis la séquence suivante, un autre répertoire est choisi. Cette opération sélectionne un ensemble de fichiers qui porte le message secret.
4. **Upload des fichiers vers les comptes clouds** : la dernière opération de la dissimulation consiste à transférer les fichiers sélectionnés dans les espaces de stockage des clouds.

Pour effectuer la récupération du secret, le récepteur doit entrer la clé qui est partagée entre l'émetteur et le destinataire constituée de la base B et des différents clouds intervenants. Puis les fonctions suivantes sont exécutées dans l'ordre pour récupérer le secret :

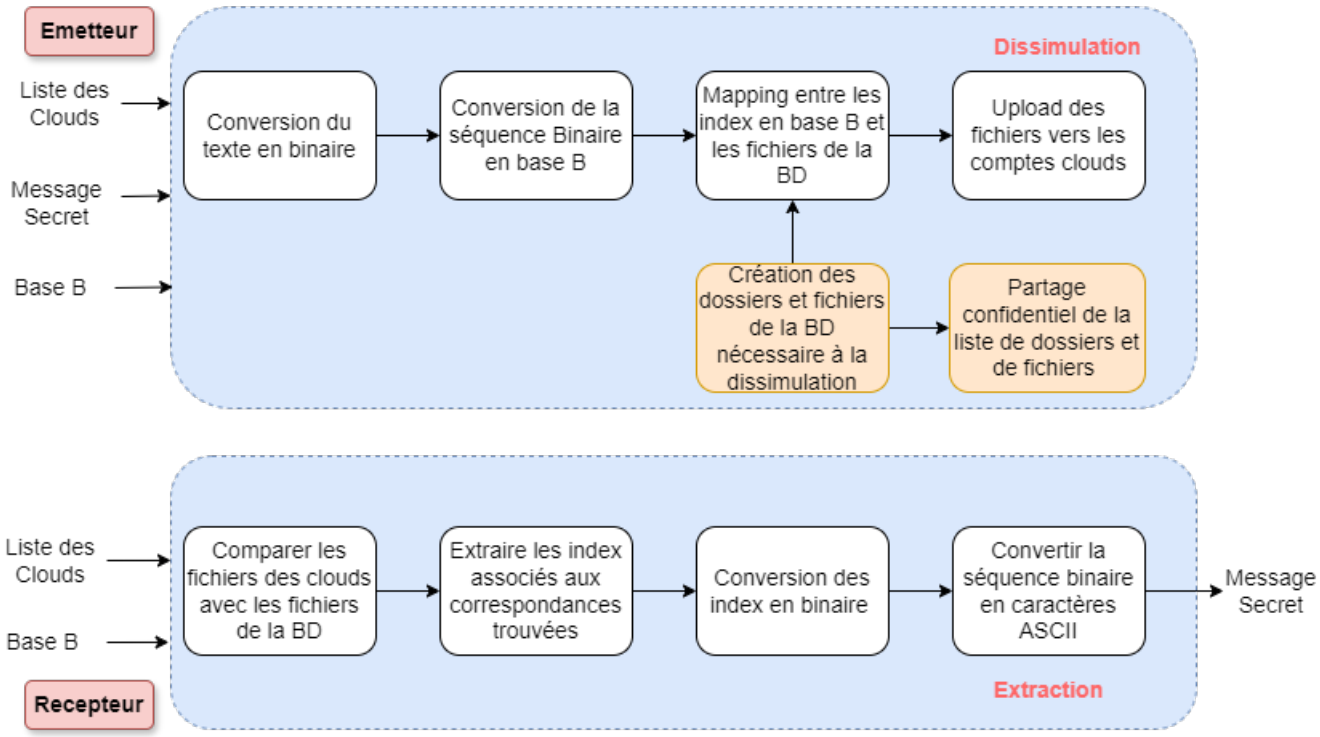


FIGURE 3.3 – Architecture de la méthode proposée

1. **Comparer les fichiers des clouds avec les fichiers de la BD** : le destinataire ayant la liste des clouds, va se connecter et comparer chaque fichier trouvé aux fichiers contenus dans les dossiers de la base de données. La comparaison se fait par nom. En cas de correspondance les fichiers de la base de données sont sélectionnées.
2. **Extraire les index associés aux correspondances trouvées** : Pour chaque fichier sélectionné, déterminer son index ou ordre de classement dans le répertoire. Ces index sont réorganisés d'une part en fonction des dossiers ou ils proviennent et d'autre part en à partir du cloud ou le fichier à été déposé.
3. **Conversion des index en binaire** : cette opération convertit Chaque index en binaire sur $k = log_2(B)$ bits. Nous obtenons à la fin une séquence binaire de la concaténation des différents blocs de k bits.
4. **Convertir la séquence binaire en caractères ASCII** : la séquence binaire est convertie en bloc de 8 bits puis chaque bloc est remplacé par un caractère ASCII. Cela permet d'obtenir en définitive le message secret initial.

3.2.2 Méthodologie

Nous apportons des réponses aux questions de recherche formulées à l'introduction générale, pour décrire comment notre solution améliore les insuffisances des méthodes existantes.

– Pour pallier à la vulnérabilité des attaques par distorsion, nous construisons une base de données utilisant des pointeurs ou un référencement local sur les fichiers. Cela permet l'assignation des index aux fichiers pour faciliter leur identification et la dissimulation du secret. Ainsi, toute modification du contenu d'un fichier n'empêche pas la reconstruction du secret dans sa totalité, car les index ou fragment du secret sont directement associés aux noms des fichiers et non au contenu.

– L'indépendance du support est mise en œuvre par l'utilisation de fichiers ayant tout type d'extensions de fichiers. En effet, l'idée est de pouvoir faire correspondre chaque partie du secret avec un fichier quelconque, indépendamment de son format ;

– Pour transférer un grand volume d'informations sans attirer l'attention, nous utilisons l'indirection mise en oeuvre à l'aide des environnements de stockage multiclouds. L'idée est de pouvoir transférer les fichiers en exploitant le stockage du cloud, jouant un rôle d'intermédiaire entre l'émetteur et le destinataire à l'aide du partage de fichiers.

3.2.3 Objet de couverture

Il s'agit de fichiers de n'importe quel type d'extension, sélectionnés pour être déposés dans l'environnement de stockage des clouds. Les fournisseurs de stockage cloud utilisés sont nommés ainsi : $c_0, c_1, ..., c_{n-1}$, $n \geq 2$.

3.2.4 Messages secrets

L'étape préliminaire nécessite que le secret soit chiffré en utilisant la clé sécrète partagée entre l'émetteur et le destinataire à l'aide du système AES pour empêcher l'extraction du secret [72].

3.2.5 Clé stéganographique

Cinq éléments sont partagés entre l'expéditeur et le destinataire :

– L'ordre des clouds : $c_0, c_1, ..., c_{n-1}$;
– Les comptes d'authentification (nom d'utilisateur et mot de passe) pour l'accès au cloud nommés comme : $w_0, w_1, ..., w_{n-1}$;
– Un ensemble de listes disjointes $L^{(0)}, L^{(1)}, ..., L^{(k-1)}$, où chaque liste i contient exactement B fichiers : $L_0^{(i)}, L_1^{(i)}, ..., L_{B-1}^{(i)}$, $i = 0, 1, ..., k-1$. Ces fichiers peuvent prendre n'importe quel type de format tel que texte, image, audio, vidéo, application, archive, ...
– La base B telle que : $\mid L^{(0)} \mid = \mid L^{(1)} \mid = ... = \mid L^{(k-1)} \mid = B$, avec $|.|$ le cardinal de l'ensemble.
– La clé secrète du chiffrement AES.

3.2.6 Notations et hypothèses

Ces notations sont utiles pour l'algorithme d'insertion et d'extraction du secret :

- s : le secret d'entrée formaté en base 2 ou 10 ;
- B : la base utilisée telle que : $B \geq 2$;
- $(z_{q-1}...z_1z_0)_B$: est la représentation secrète de s dans la base B ;
- $V[i]$: est le $i^{ème}$ bloc du secret ;
- $Mat[i, j]$: est la valeur à la position j du bloc numéro i ;
- n : est le nombre de clouds manipulés ;
- $c_0, c_1, ..., c_{n-1}$: les n clouds utilisés $c_0, c_1, ..., c_{n-1}$:
- k : est le numéro de bloc secret ;
- $L^{(i)}$: est la $i^{ème}$ liste des fichiers. Chaque bloc secret utilise une liste distincte i, $0 \leq i \leq k-1$;
- $L_j^{(i)}$: est le $j^{ème}$ fichier dans la liste numéro i, $0 \leq i \leq k-1$ et $0 \leq j \leq B-1$;
- w_i : Le compte d'authentification (nom d'utilisateur et mot de passe) pour l'accès au cloud numéro i ;
- c_i : est l'espace de stockage du cloud numéro i.

Voici les hypothèses du schéma proposé :

- Les listes : $L^{(0)}$, $L^{(1)}$, ..., $L^{(k-1)}$ sont disjointes :
  $\forall i_1, i_2$ désignant les listes, $0 \leq i_1, i_2 \leq n-1$,
  $\forall j_1, j_2$ désignant les fichiers, $0 \leq j_1, j_2 \leq B-1$,
  Si $i_1 \neq i_2$ alors $L_{j_1}^{(i_1)} \neq L_{j_2}^{(i_2)}$.

3.2.7 Algorithme d'insertion

Cet algorithme est utilisé pour transférer un secret au destinataire. Dans un premier temps, l'émetteur chiffre le secret puis convertit le résultat en base B. Puis la représentation secrète dans la base B est divisée en blocs de n valeurs. Ainsi, pour chaque bloc secret, ouvrir le stockage cloud avec le nom d'utilisateur et le mot de passe associé. Trouvez dans la liste le fichier ayant comme index la valeur du bloc. Puis déplacer le fichier correspondant vers le cloud. Ensuite répéter l'opération sur le prochain espace de stockage cloud. L'algorithme 3.1 présente l'insertion du secret dans l'environnement de stockage multicloud.

Une amélioration de ce schéma a été proposé par Mossebo et al. [73]. Le but est de réduire la complexité en effectuant les opérations de dissimulation directement à l'intérieur d'un seul cloud. Le seul paramètre intervenant est la base B. De même, la liste des dossiers et fichiers sont directement accessible dans le cloud.

Algorithme 3.1: Algorithme d'insertion du secret dans l'environnement multi-cloud

Entrées : C : l'ensemble des clouds ;
W : l'ensemble des comptes clouds d'authentification ;
L : l'ensemble des listes de fichiers ;
B : la base ;
s : le message secret en binaire ;
Sorties : C' : les listes de cloud après stockage des fichiers ;

début
  Convertir le secret S en base B tel que : $s = (z_{q-1}...z_1z_0)_B$, où $0 \leq z_i \leq B-1$;
  Découper s en k blocs de longueur n stockés dans la matrice Mat tels que : $Mat[i,j] = s[(i \times n) + j]$, $0 \leq i \leq k-1$ et $0 \leq j \leq n-1$;
  pour *chaque bloc de* Mat *:* $0 \leq i \leq k-1$ **faire**
    pour *chaque valeur* $Mat[i,j]$ *:* $j = 0, 1, ..., n-1$ **faire**
      $u = Mat[i,j]$;
      Trouver dans la i^e liste $L^{(i)}$ les fichiers à l'index $Mat[i,j]$;
      Sélectionner et uploader sur le cloud c_j le fichier $L_u^{(i)}$;
    fin
  fin
fin

3.2.8 Algorithme d'extraction

Cet algorithme récupère le secret laissé par l'émetteur. Le récepteur s'authentifie et parcourt chaque espace cloud pour récupérer les fichiers trouvés dans la liste partagée entre l'expéditeur et le destinataire. Puis, il détermine les index de ces fichiers et trie les index dans l'ordre de numérotation des listes d'où ils proviennent. Ensuite, l'émetteur stocke les index de chaque cloud par colonne dans une matrice. Il utilise par la suite la valeur de la base B pour récupérer le secret contenu dans la matrice. Pour terminer, le secret est déchiffré puis tous les fichiers déposés dans les espaces de stockage cloud sont supprimés. L'algorithme 3.3 présente l'extraction du secret dans l'environnement multicloud.

Algorithme 3.2: Algorithme d'extraction du secret dans les environnements clouds

Entrées : C : l'ensemble des clouds ;
W : l'ensemble des comptes cloud d'authentification ;
L : l'ensemble des listes de fichiers ;
B : la base ;
Sorties : s : le message secret ;

1 **début**
2 **pour** *chaque nuage* c_j, $j = 0, 1, ..., n-1$ *:* **faire**
3 Extraire dans le cloud c_j les fichiers contenus dans les listes $L^{(0)}, L^{(1)}, ..., L^{(k-1)}$;
4 Trouver les index associés à ces fichiers nommés : $z_0, z_1, ..., z_{k-1}$;
5 **pour** *chaque numéro de bloc* $i = 0, 1, ..., k-1$ *:* **faire**
6 $Mat[i, j] = z_i$;
7 **fin**
8 Calculer $m = \sum_{i=0}^{k-1} \sum_{j=0}^{n-1} (Mat[i, j] \times B^{(i \times n)+j})$;
9 Convertir m en binaire et retrouver le message secret s;
10 Supprimer des clouds tous les fichiers utilisés pour récupérer le secret.
11 **fin**
12 **fin**

3.2.9 Analyse de la complexité en temps

Dans cette sous-section, nous étudions l'analyse de la complexité en temps du schéma stéganographique proposé. Nous supposons que nous avons un secret s à répartir entre n espaces de stockage cloud en utilisant la valeur de base B. Nous supposons également que le secret s est caché à l'aide de k listes de fichiers, chacune contenant B fichiers. L'algorithme 3.1 d'intégration proposé convertit le secret s en base B en $O(log_B(s))$ (ligne 2). La subdivision de s en blocs (ligne 3) et le choix des fichiers (ligne 4 à 10) à stocker dans le cloud se font en $O(n * k)$. Par conséquent, la complexité en temps du schéma d'insertion est de $O(n * k)$.

Dans l'algorithme 3.3 d'extraction proposé, nous supposons que le cloud contient m fichiers. L'extraction des fichiers du cloud (ligne 3) contenus dans les listes se fait en $O(m*k*B)$. Trouver les index des fichiers (ligne 4) et le remplissage de la matrice (ligne 5 à 7) se font en $O(k)$. De plus, le secret est converti en décimal (ligne 8) en $O(n * k)$. Enfin, la conversion de la valeur décimale secrète en base 2 (ligne 9) se fait en $O(log_2(s))$ et la suppression des fichiers dans le cloud est en $O(n)$. Par conséquent, la complexité en temps du schéma d'extraction du secret est de $O(m * k * B)$.

3.2.10 Exemples

Pour décrire le schéma stéganographique proposé, des exemples numériques simples sont détaillés ci-dessous. Dans ces exemples, s=1111101101000001 est un secret de 16 bits et le nombre de clouds gérés est défini à la valeur quatre (n=4). Les fournisseurs de stockage utilisés et leurs identifiants respectifs sont : iCloud (c_0), Dropbox (c_1), OneDrive (c_2) et Google Drive (c_3). Le tableau 3.1 montre les identifiants de connexion cloud associés. Les quatre listes $L^{(0)}, L^{(1)}, L^{(2)}$ et $L^{(3)}$ utilisées pour intégrer le secret sont présentées dans le tableau 3.2. Ensuite, deux scénarios sont mis en évidence avec la base prenant ces valeurs successives : B=2, B=4, B=9 et B=17. Chaque cas présente la distribution secrète entre ces environnements de stockage cloud et explique en détail comment le secret est intégré et extrait à l'aide de listes de fichiers. Notez que les listes de fichiers, la valeur de la base, l'ensemble des clouds et leurs identifiants de connexion forment la clé, partagée entre l'expéditeur et le destinataire.

TABLE 3.1 – Ensemble des quatre environnements de stockage cloud et leur information de connexion

Code	Nom du Cloud	Login	Mot de Passe
c_0	iCloud	userlogin0@gmail.com	User-pwd0
c_1	Dropbox	userlogin1@gmail.com	User-pwd1
c_2	OneDrive	userlogin2@gmail.com	User-pwd2
c_3	Google Drive	userlogin3@gmail.com	User-pwd3

TABLE 3.2 – Quatre listes de fichiers et leur numéro d'index.

	(a) Liste $L^{(0)}$		(b) Liste $L^{(1)}$		(c) Liste $L^{(2)}$		(d) Liste $L^{(3)}$
0	thesis.docx	**0**	scheduling.xlsx	**0**	conference.pptx	**0**	dataHiding.pdf
1	article.docx	**1**	statistics.xlsx	**1**	results.pptx	**1**	cryptography.pdf
2	balanceSheet.docx	**2**	budget.xlsx	**2**	slideshow.pptx	**2**	deepLearning.pdf
3	report.docx	**3**	data.xlsx	**3**	marketing.pptx	**3**	linearAlgebra.pdf
4	meeting.docx	**4**	bill.xlsx	**4**	management.pptx	**4**	dataScience.pdf
5	opportunities.docx	**5**	evaluation.xlsx	**5**	slides.pptx	**5**	Publications.pdf
6	lesson.docx	**6**	gradebook.xlsx	**6**	animation.pptx	**6**	sourceCode.pdf
7	chapter.docx	**7**	stocks.xlsx	**7**	overview.pptx	**7**	dataAnalysis.pdf
8	introduction.docx	**8**	simulation.xlsx	**8**	speech.pptx	**8**	modelingLife.pdf
9	tutorial.docx	**9**	project.xlsx	**9**	seminar.pptx	**9**	cloudComputing.pdf
10	redaction.docx	**10**	analysis.xlsx	**10**	symposium.pptx	**10**	masterDegree.pdf
11	news.docx	**11**	curves.xlsx	**11**	resume.pptx	**11**	bachelorDegree.pdf
12	book.docx	**12**	quotation.xlsx	**12**	shopping.pptx	**12**	bithdayCertificate.pdf
13	exercise.docx	**13**	finance.xlsx	**13**	accounts.pptx	**13**	passport.pdf
14	anthem.docx	**14**	classes.xlsx	**14**	clinical.pptx	**14**	huffman.pdf
15	journal.docx	**15**	salaries.xlsx	**15**	aviation.pptx	**15**	contacts.pdf
16	editor.docx	**16**	phonebook.xlsx	**16**	audition.pptx	**16**	awards.pdf

3.2.10.1 Cas 1 : s=1111101101000001, n=4 et B=2

Les étapes suivantes permettent d'intégrer le message secret :

Etape 1 : Le secret est déjà représenté en base 2, $s = (1111101101000001)_2$;

Etape 2 : Le secret est subdivisé en groupes de 4 bits, au regard des quatre clouds disponibles. De droite à gauche cela donne 4 blocs : 0001 0100 1011 1111 ;

Etape 3 : Pour chaque bloc, chaque bit est lié à un cloud distinct dans l'ordre c_0, c_1, c_2 et c_3 :

Bloc #0				Bloc #1				Bloc #2				Bloc #3			
0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0

Étape 4 : Les bits d'un même cloud sont regroupés sur la même colonne. Par conséquent, les parties secrètes de chaque cloud sont :

c_0	c_1	c_2	c_3
1	0	0	0
0	0	1	0
1	1	0	1
1	1	1	1

Etape 5 : Les quatre listes de la Table 3.2 sont utilisées pour dissimuler les quatre blocs secrets. Cacher respectivement la 1ère, 2ème, 3ème et 4ème ligne de la matrice obtenue à l'étape 4 avec la liste $L^{(0)}, L^{(1)}, L^{(2)}$ et $L^{(3)}$. Plus précisément, chaque valeur est remplacée par le fichier ayant cet index dans la liste correspondante. Ces fichiers servent de pointeurs vers les données à garder secrètes. Les fichiers stégos à uploader dans chaque cloud sont alloués comme suit :

Liste	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	article.docx	thesis.docx	thesis.docx	thesis.docx
$L^{(1)}$	scheduling.xlsx	scheduling.xlsx	statistics.xlsx	scheduling.xlsx
$L^{(2)}$	results.pptx	results.pptx	conference.pptx	results.pptx
$L^{(3)}$	cryptography.pdf	cryptography.pdf	cryptography.pdf	cryptography.pdf

Étape 6 : La dernière étape d'intégration consiste à transférer les fichiers article.docx, schedule.xlsx, results.pptx et cryptography.pdf vers le cloud c_0 ; thèse.docx, ordonnancement.xlsx, résultats.pptx et cryptographie.pdf vers le cloud c_1 ; thèse.docx, statistics.xlsx, conference.pptx et cryptographie.pdf vers le cloud c_2 ; thesis.docx, schedule.xlsx, results.pptx et cryptography.pdf vers le cloud c_3.

Suivre les étapes suivantes pour extraire le secret :

Etape 1 : Les fichiers de chaque cloud sont comparés à ceux disponibles dans les quatre listes $L^{(0)}, L^{(1)}, L^{(2)}$ et $L^{(3)}$. Lorsque les noms sont identiques, ces fichiers sont récupérés et triés par ordre croissant de numérotation des listes. Les fichiers extraits par cloud et par liste sont les suivants :

Cloud	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
c_0	article.docx	scheduling.xlsx	results.pptx	cryptography.pdf
c_1	thesis.docx	scheduling.xlsx	results.pptx	cryptography.pdf
c_2	thesis.docx	statistics.xlsx	conference.pptx	cryptography.pdf
c_3	thesis.docx	scheduling.xlsx	results.pptx	cryptography.pdf

Etape 2 : Les fichiers de chaque liste sont alors remplacés par leur numéro. La séquence de chaque cloud obtenu est :

Cloud	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
c_0	1	0	1	1
c_1	0	0	1	1
c_2	0	1	0	1
c_3	0	0	1	1

Etape 3 : Chaque séquence binaire appartenant à un cloud est stockée en colonne à l'intérieur d'une matrice appelée Mat :

$$\mathbf{Mat} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Étape 4 : Calculez m, le secret en décimal en utilisant la valeur de la base (B = 2). Les variables i et j parcourent respectivement les lignes et les colonnes de la matrice Mat. La conversion se fait comme suit :

$$\begin{aligned} m &= \sum_{i=0}^{3}\sum_{j=0}^{3}(Mat[i,j] \times 2^{(i\times 4)+j}) \\ &= 1*2^0 + 0*2^1 + 0*2^2 + 0*2^3 + \\ &\quad 0*2^4 + 0*2^5 + 1*2^6 + 0*2^7 + \\ &\quad 1*2^8 + 1*2^9 + 0*2^{10} + 1*2^{11} + \\ &\quad 1*2^{12} + 1*2^{13} + 1*2^{14} + 1*2^{15} \\ &= 1 + 64 + 256 + 512 + 2\,048 + 4\,096 + 8\,192 + 16\,384 + 32\,768 \\ &= 64\,321 \end{aligned}$$

Etape 5 : Le secret s est obtenu en convertissant m en base 2 :
$(64321)_{10} = (1111101101000001)_2$

Étape 6 : Tous les fichiers récupérés à l'étape 1 de l'extraction sont supprimés de l'espace de stockage cloud.

3.2.10.2 Cas 2 : s=1111101101000001, n=4 et B=17

Dans ce dernier cas, le secret s et le nombre de clouds n restent inchangés. La base considérée est B = 17. Suivons ces étapes pour intégrer le secret :

Etape 1 : Le secret est converti en base 17 :
$(1111101101000001)_2 = (64321)_{10} = (D19A)_{17}$;

Étape 2 : Le secret est subdivisé en groupes de 4 valeurs, en raison des quatre clouds disponibles. Cela donne un bloc : D19A.

Etape 3 : Pour chaque bloc, chaque valeur est liée à un cloud distinct dans l'ordre c_0, c_1, c_2 et c_3 :

Bloc #0			
D	1	9	A
↓	↓	↓	↓
c_3	c_2	c_1	c_0

Etape 4 : Les valeurs d'un même cloud sont regroupées. Les lettres du bloc sont également remplacées par leur équivalent : D=13 et A=10. Par conséquent, les parties secrètes de chaque cloud sont :

c_0	c_1	c_2	c_3
10	9	1	13

Étape 5 : Comme la subdivision a donné un bloc, une seule liste est utilisée. Cachez les valeurs du vecteur obtenues à l'étape 4 avec la liste $L^{(0)}$. Les fichiers stego à uploader dans les clouds sont répartis comme suit :

List	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	redaction.docx	tutorial.docx	article.docx	exercise.docx

Etape 6 : La dernière étape d'intégration consiste à transférer les fichiers redaction.docx vers le cloud c_0 ; tutorial.docx vers le cloud c_1 ; article.docx vers le cloud c_2 et exercice.docx vers le cloud c_3.

Suivre ces étapes pour extraire le secret :

Etape 1 : Les fichiers de chaque cloud sont comparés à ceux disponibles dans la liste $L^{(0)}$. Lorsque les noms sont identiques, ces fichiers sont récupérés et triés dans l'ordre de création des listes. Les fichiers extraits par cloud et par liste sont les suivants :

Etape 2 : Les fichiers de chaque liste sont alors remplacés par leur numéro. La séquence de chaque cloud obtenu est :

Cloud	$L^{(0)}$
c_0	redaction.docx
c_1	tutorial.docx
c_2	article.docx
c_3	exercise.docx

Cloud	$L^{(0)}$
c_0	10
c_1	9
c_2	1
c_3	13

Etape 3 : La séquence est stockée en colonne dans la matrice Mat. Le vecteur résultant ressemble à ceci :

$$\mathbf{Mat} = \begin{pmatrix} 10 & 9 & 1 & 13 \end{pmatrix}$$

Étape 4 : Calculez m, le secret en décimal en utilisant la valeur de la base (B = 17). La conversion se fait comme suit :

$$\begin{aligned} m &= \sum_{i=0}^{0}\sum_{j=0}^{3}(Mat[i,j] \times 17^{(i\times 4)+j}) \\ &= 10 * 17^0 + 9 * 17^1 + 1 * 17^2 + 13 * 17^3 \\ &= 10 + 153 + 289 + 63\,869 \\ &= 64\,321 \end{aligned}$$

Etape 5 : Le secret s est obtenu en convertissant m en base 2 :
$s = (64321)_{10} = (1111101101000001)_2$

Étape 6 : Tous les fichiers récupérés à l'étape 1 de l'extraction sont supprimés de l'espace de stockage cloud.

3.3 Expérimentations

Cette section est dédiée à l'expérimentation de la méthode proposée. Le but est de prendre un secret donné et de le dissimuler dans l'espace de stockage multicloud. Puis de vérifier si les hypothèses formulées sont validées à travers l'expérimentation.

3.3.1 Rappel des hypothèses de recherche

Nous rappelons ces hypothèses :

Hypothèse 1 : L'approche de dissimulation permet de retrouver la totalité du secret en cas d'altération du support ;

Hypothèse 2 : Tout type de fichiers multimédia peut être utilisé lors la dissimulation ;

3.3.2 Paramètres de configuration

Les paramètres de configuration suivants seront pris en compte dans cette expérimentation :

- Le secret à cacher est le suivant : "*La stéganographie est un art de communication secrète dans lequel un secret donné est dissimulé à l'intérieur d'un fichier donné*". Il est composé de 128 caractères ASCII soit 1024 bits de secret, à raison de 8 bits par caractère ;

- Ce secret est convertit avec la base B égale à 2^{20} =1 048 576. Cela revient après avoir décomposé le secret en binaire à faire des blocs de 20 bits pour passer à la phase de dissimulation.

- Cinq clouds seront utilisés pour transférer les fichiers dans l'espace de stockage. Il s'agit de : *Google drive, OneDrive, Dropbox, Pcloud et kDrive* ;

- Pour la dissimulation, nous allons construire une base de données contenant des fichiers word, excel, powerpoint, textes et images. Les extensions de ces fichiers sont les suivantes : *.docx, .xlsx, .pptx, .pdf, .txt et .jpg* ;

- Les fichiers stockés à l'intérieur de chaque dossier sont classés par ordre alphabétique. Cela permet de définir le numéro ou la position qu'occupe un fichier dans chaque répertoire. Cela va servir à faire la correspondance entre le secret et le fichier à déplacer dans l'un des clouds au moment de la dissimulation ;

- La base de données de fichiers sera organisée en 11 dossiers. Cela s'obtient par le rapport du nombre de blocs du secret par le nombre de clouds intervenant lors de la dissimulation. Chaque dossier contient 1 048 576 fichiers ayant les extensions citées plus haut ;

- Les fichiers de la base de données contenus dans les dossiers ont une taille variant entre $1Ko$ et $7Mo$.

Le tableau 3.3 présente la synthèse des paramètres de configuration utilisées.

3.3.3 Résultats de l'expérimentation

Après la dissimulation, la distribution du secret est présentée à la figure 3.4. La décomposition du secret en blocs de 20 bits permet de déterminer le numéro des fichiers

TABLE 3.3 – Paramètres de configuration de l'expérimentation

Secret	La stéganographie est un art de communication secrète dans lequel un secret donné est dissimulé à l'intérieur d'un fichier donné
Taille du secret	1024 bits
Base	2^{20} =1 048 576
Taille des blocs	20 bits
# Clouds	5
Liste des clouds	Google drive, OneDrive, Dropbox, Pcloud, kDrive
# Dossiers	11
# Fichiers/Dossier	1 048 576
# Classement des fichiers	Ordre alphabétique
Extension des fichiers	.docx, .xlsx, .pptx, .pdf, .txt, .jpg
Taille des fichiers	1 ko à 7 Mo

à déplacer dans l'un des clouds. Pour ce faire chaque valeur binaire du bloc est converti en decimal. Puis un fichier est sélectionné lorsque son numéro d'ordre dans le dossier correspond à la valeur décimale du secret. Ces valeurs sont comprises entre 0 et 1 048 576. Nous remarquons que pour ce graphique(voir figure 3.4), la plus petite valeur des blocs secrets est 9 (Bloc numéro 52) tandis que la plus grande valeur est 505 655 (bloc numéro 32).

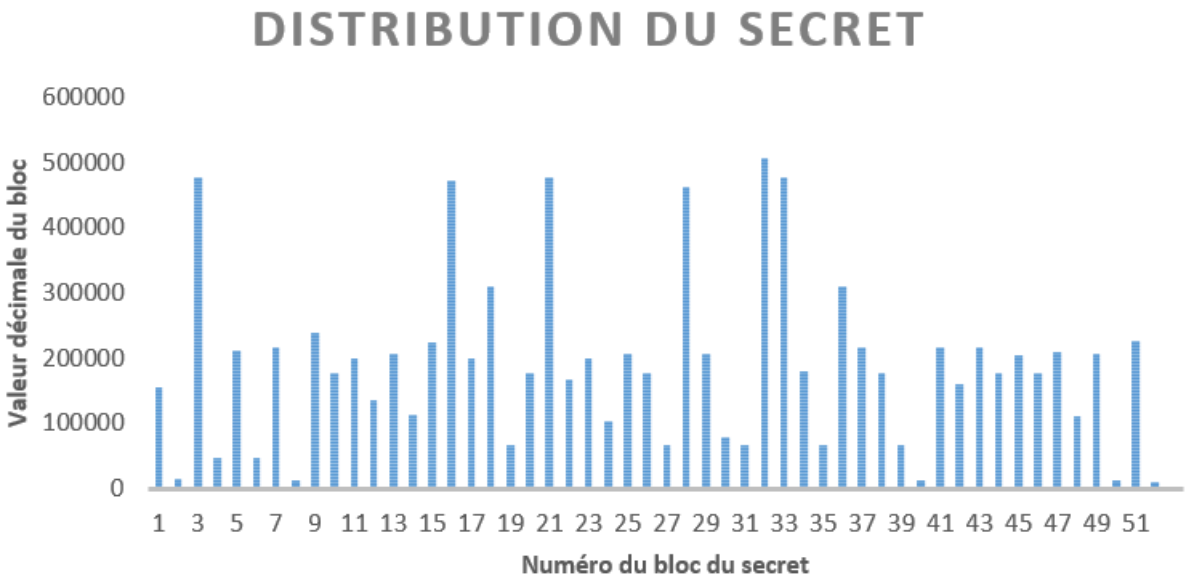


FIGURE 3.4 – Distribution du secret par bloc de 20 bits

Dans cette expérimentation, nous évaluons trois critères : Le nombre de fichiers déposés dans les environnements de stockage cloud, le nombre de bits dissimulés et l'espace disque occupé par ces fichiers. Le tableau 3.4 présente les résultats obtenus.

TABLE 3.4 – Résultats obtenus après dissimulation du secret

	Google Drive	One Drive	Dropbox	Pcloud	kDrive	**Total**
# Fichiers	11	11	10	10	10	**52**
# bits	220 bits	204 bits	200 bits	200 bits	200 bits	**1024 bits**
Espace disque	37 Ko	23 Mo	18Mo	350 Ko	180 Ko	**41.9 Mo**

La synthèse de ces résultats donnent au total :

– 52 fichiers uploadés dans le stockage des cinq clouds, soit 10 fichiers en moyenne par cloud ;

– 1024 bits du secret repartis entre les cinq clouds, soit de 204 bits en moyenne par cloud ;

– 41.9 Mo d'espace disque occupé par les différents fichiers déplacés, soit 8 Mo en moyenne par cloud.

3.3.4 Validation des hypothèses

3.3.4.1 Hypothèses 1

Le but de l'hypothèse 1 est de vérifier si la méthode proposée permet d'extraire la totalité du secret même en cas d'altération du support. Considérons les 10 premiers fichiers contenus dans le 11^e dossier illustrés dans le tableau 3.5. D'après la figure 3.4, le dernier bloc du secret a pour valeur 9, ce qui va correspondre à la sélection du fichier ayant la position 9 dans le dossier numéro 11. C'est un fichier texte intitulé book.docx contenant le chapitre 1 du livre de programmation en C. Ce fichier va être modifié en ajoutant le même contenu deux fois. Après enregistrement, le fichier aura toujours la même position, c'est à dire la position 9. Cela peut être généralisé à tous les fichiers utilisés comme base de données après modification. Ainsi, la dissimulation du même secret va permettre de sélectionner les fichiers aux mêmes positions. Comme illustré dans le tableau 3.5, le secret est lié à la position qu'occupe le fichier dans le dossier suivant un ordre définit préalablement (ordre alphabétique). Par conséquent toute attaque sur le contenu du fichier permettra toujours de retrouver le secret.

TABLE 3.5 – Extrait des 10 premiers fichiers du 11^e dossier

Secret/Position	**Nom du Fichier**
0	accounts.pptx
1	analysis.xlsx
2	animation.pptx
3	anthem.docx
4	article.docx
5	bill.xlsx
6	bithdayCertificate.pdf
7	bithdayCertificate.pdf
8	book.docx
9	budget.xlsx

3.3.4.2 Hypothèses 2

Le but de l'hypothèse 2 est de vérifier si la méthode proposée permet de manipuler tout type d'extension de fichier lors de la dissimulation. L'expérimentation s'est faite avec des fichiers d'extensions : *.docx, .xlsx, .pptx, .pdf, .txt et .jpg*. Comme illustré dans le tableau 3.5, plusieurs extensions de fichiers peuvent être utilisées pour la dissimulation. La méthode suggère d'avoir un nombre précis de fichiers par dossier. La nature ou le type de fichiers est choisi aléatoirement par l'utilisateur. La seule contrainte ici est de choisir des fichiers de taille raisonnable (quelques kilo ou méga octets) pour que leur déplacement soit rapide et pas coûteux en temps.

3.4 Evaluation

Les expérimentations ont été effectuées en utilisant l'environnement ubuntu et le langage de programmation python. Les critères d'évaluation sont les suivants :

- Nombre de fichiers à déposer dans le cloud ;
- Nombre de listes de fichiers ou taille de la base de données ;
- La capacité des bits secrets ;
- La robustesse ;
- Et la sécurité ;

3.4.1 Evaluation du nombre fichiers et de listes

Formule d'estimation du nombre de fichiers

$$m = \left\lceil \frac{|(S)_B|}{log_2(B)} \right\rceil \tag{3.1}$$

Avec m représentant le nombre de fichiers à déposer dans l'environnement multicloud après la dissimulation du secret. Cela s'obtient en faisant le rapport entre la taille du secret et le nombre de bits à dissimuler dans chaque fichier. Voir l'exemple de la sous section 3.2.10.1, avec un secret de 16 bits, $n = 4$ et $B = 2$. Cela nécessite 4 listes. Autrement dit, nous avons 16 fichiers à déplacer à raison de 4 fichiers par dossier. Puis, chacun des 4 fichiers issu du même dossier est déplacé dans un cloud donné.

Formule d'estimation du nombre de listes

$$k = \left\lceil \frac{|(S)_B|}{log_2(B^n)} \right\rceil \tag{3.2}$$

Avec :
- B : la base
- n : le nombre de clouds
- S : le secret
- k : le nombre de listes
- m : le nombre de fichiers à déposer dans les Clouds.

k représente le nombre de listes à créer autrement dit le nombre de dossiers de la base de données de fichiers utile pour la dissimulation. Cela s'obtient par le rapport entre la taille du secret et le nombre de bits dissimulé dans une séquence de n clouds. En effet, selon la méthode proposée, les fichiers à uploader proviennent du même dossier pour une séquence de n cloud. Puis un autre dossier est utilisé lors de la séquence suivante de n clouds. Voir l'exemple de la sous section 3.2.10.2, avec un secret de 16 bits, $n = 4$t $B = 17$. Cela donne 4 fichiers à déplacer vers le cloud.

La figure 3.5 présente l'évaluation du nombre de fichiers en fonction des tailles des blocs du secret. La formule qui détermine le nombre fichiers est donnée au début de la section 3.4.1. Les tailles de blocs du secret varient de 10 à 80 bits par pas de 10 bits pour des secrets de 100bits, 500 bits et 1kbit. La taille des blocs x du secret est liée à la base B choisit par la formule $B = 2^x$. Cela détermine le nombre de bits dissimulé dans chaque fichier à déplacer vers le cloud. Ainsi, pour chaque taille de secret, nous évaluons le nombre de fichiers à déplacer dans l'environnement multicloud.

La figure 3.6 présente l'évaluation du nombre de listes en fonction de la taille du secret et du nombre de clouds. La formule qui détermine le nombre de listes est donnée au début de la 3.4.1. Les tailles de blocs du secret varient de 20 à 160 bits par pas de 20 bits pour des secrets de 100bits, 500 bits et 1kbit. Le nombre de listes est déterminé pour n=5, n=15 et n=25 clouds. Ainsi, pour chaque taille de secret et nombre de clouds, nous évaluons le nombre de dossier à utiliser pour sélectionner les fichiers qui vont porter le secret.

Les figures 3.5 et 3.6 permettent d'évaluer respectivement le nombre de fichiers et le nombre de listes. Nous observons que lorsque la taille des blocs du secret augmente (la base B), le nombre de fichiers à déposer dans les clouds décroit. De même lorsque la taille des blocs du secret et le nombre de clouds augmentent, le nombre de liste diminue.

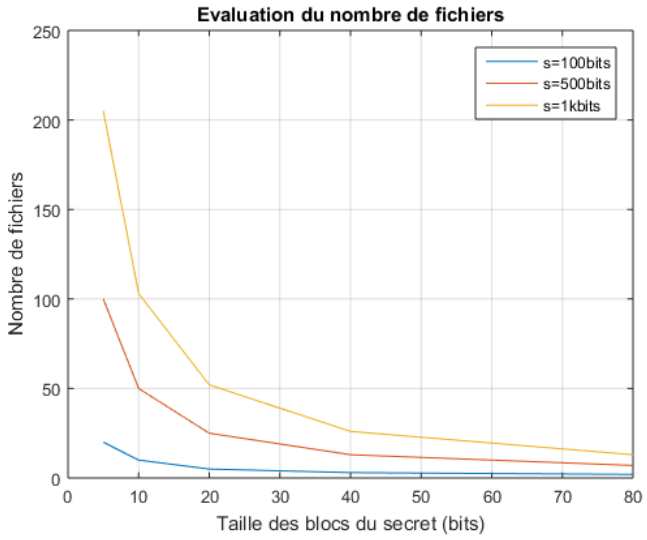


FIGURE 3.5 – Évaluation du nombre de fichiers pour différentes tailles du secret

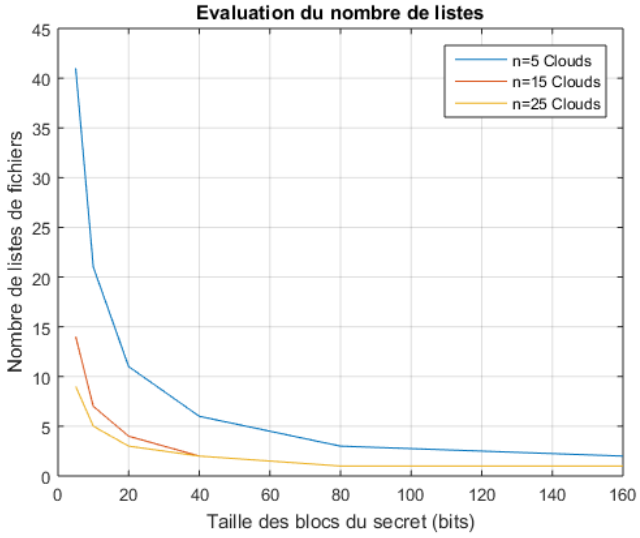


FIGURE 3.6 – Evaluation du nombre de listes en fonction de la taille du secret et du nombre de clouds

3.4.2 Evaluation de la capacité des bits secrets

L'objectif ici est de cacher des bits secrets dans un ensemble de n clouds. Chaque cloud embarque une valeur dans la base B, et cette valeur peut varier de 0 à $B-1$, donc

B possibilités. Alors pour un ensemble de n clouds, il y a B^n possibilités. Ainsi, pour un secret à k blocs, le nombre de bits cachés est : $k \cdot log_2(B^n) = k \cdot n \cdot log_2(B)$.

Dans nos expérimentations nous avons retenu comme base $B = 2^{20}$ soit une capacité de 20 bits par fichiers déposé dans le cloud. La Figure 3.7 présente la comparaison des capacités des différentes méthodes. Notre méthode de dissimulation dans l'environnement multiloud (EMC) a une capacité relativement basse comparée aux méthodes de whu et al et d'Abdulsattar et al.

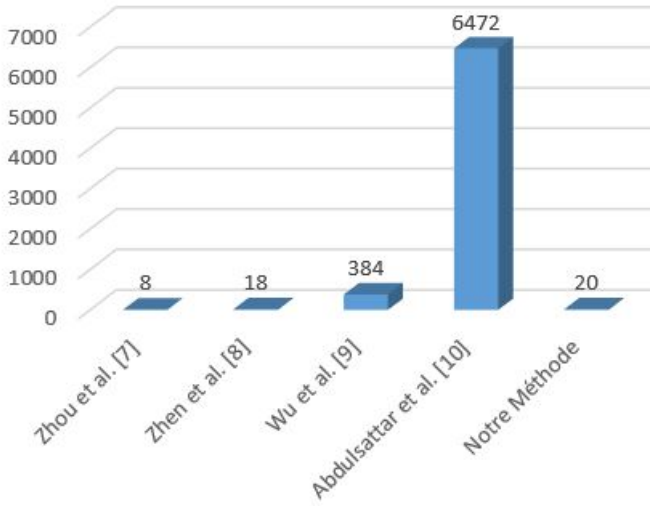


FIGURE 3.7 – Évaluation de la capacité des bits secrets

3.4.3 Evaluation de la robustesse

La méthode proposée utilise des supports dont le contenu ne dépend pas du secret. Ainsi toute distorsion du contenu n'affecte pas l'extraction du secret. Ce qui n'est pas le cas des méthodes existantes de la littérature. Le Tableau 3.6 illustre cette comparaison.

TABLE 3.6 – Comparaison de la robustesse entre les différentes méthodes

	Rescaling	Luminance change	Contrast enhancement	JPEG compression	Guassian noise adding
Zhou et al	0	0	0	3	2
Zhen et al	-	-	-	5	4
Wu et al.	1	2	1	1	60
Abdulsattar et al	-	-	-	-	26
Notre solution	0	0	0	0	0

La méthode proposée est robuste aux attaques par distorsion (altération) du support. L'extraction du secret pourra être difficile dans les cas suivants :

– l'interruption de la communication entre les parties
– la suppression de fichiers
– le renommage de fichiers

3.4.4 Evaluation de la sécurité

Dans cette évaluation nous allons proposer une attaque par force brute permettant de retrouver le message dissimulé dans les différents espaces de stockage des clouds. Puis nous déterminerons la complexité de cette attaque pour évaluer le niveau de sécurité du schéma stéganographique proposé.

3.4.4.1 Attaque par force brute sur les fichiers dissimulés

Dans cette attaque, nous supposons que l'attaquant a accès aux différents clouds. Il peut donc voir les fichiers distribués dans les espaces de stockage de chaque cloud. Nous supposons également qu'il ne possède pas la clé, c'est à dire les différentes listes de fichiers et la base B utilisés pour encoder le secret. L'attaquant ayant connaissance des fichiers déposés dans les clouds, il devra trouver le bon ordre des n clouds puis des k listes et enfin des B fichiers contenus dans chacune des k listes. L'algorithme de recherche du message secret par une attaque par force brute est le suivant :

Algorithme 3.3: Attaque par force brute sur les fichiers dissimulés

Entrées : C : l'ensemble des clouds ;
W : l'ensemble des informations d'authentification des clouds ;
Sorties : s : le message secret ;

début
  Déterminer le nombre de listes de fichiers utilisées pour la dissimulation noté k tel que : $k = min\{k_0, ..., k_{n-1}\}$, avec k_i le nombre le fichiers stockés dans le cloud c_i;
  Déterminer l'ordre de dissimulation du secret dans les n clouds : $(c_0...c_{n-1})$;
  Placer les fichiers de chaque cloud dans les k listes :$L^{(0)}, L^{(1)}, ..., L^{(k-1)}$;
  Attribuer les index aux B fichiers de chaque liste $L^{(i)}$;
  Extraire le secret à partir des index des fichiers;
fin

3.4.4.2 Evaluation de la complexité de l'attaque

Dans un premier temps, la complexité en temps est évaluée sur chacune des cinq étapes de l'attaque :

Etape 1 : La recherche du minimum se fait en n opérations soit une complexité en temps de $O(n)$;

Etape 2 : Déterminer le bon ordre des clouds revient à trouver la permutation utilisée au départ entre les n clouds. Toutes les permutations possibles s'obtiennent en $n!$ opérations, soit une complexité en temps de $O(n!)$;

Etape 2.1 : Ordonner les fichiers provenant du cloud à partir des k listes d'où ils proviennent se fait en determinant la bonne permutation des k listes. Ce qui nécessite $k!$ permutations possibles. Soit une complexité en $O(k!)$;

Etape 2.1.1 : Déterminer l'ordre initial des B fichiers de chaque liste s'obtient à partir des différentes permutations de B fichiers. Autrement dit cela nécessite $B!$ permutations, soit une complexité en $O(B!)$;

Etape 2.1.1.1 : Cette étape consiste à attribuer les bons index aux fichiers du cloud puis à les concaténer pour obtenir le secret. En considérant qu'il y a au total m fichiers transférés dans les espaces clouds, cela nécessite m opérations, soit une complexité en $O(m)$.

Puis dans un second temps nous évaluons la complexité totale :

$$\begin{aligned} T(n) &= n + n! * k! * B! * m \\ &= n! * k! * B! \\ &> B! \end{aligned}$$

Nous avons : $T(n) = B!$ car n, k et m sont petit devant B. Nous obtenons une complexité de type factorielle en $O(B!)$.

3.4.4.3 Evaluation du niveau de sécurité du schéma proposé

Le niveau de sécurité ou bits de sécurité [74] est une mesure qui sert à quantifier le nombre d'opérations minimum que devrait faire un adversaire pour casser un schéma cryptographique. Le niveau de sécurité est exprimée en bits par rapport à la meilleure attaque sur le schéma cryptographique, où la sécurité à n bits signifie que l'attaquant devrait effectuer 2^n opérations pour briser la sécurité. Le tableau 3.7 présente le niveau de sécurité du schéma de dissimulation de l'environnement multicloud face à une attaque par force brute qui est fonction de la valeur de la base B. Nous obtenons par exemple un niveau de sécurité de 128 bits pour $B = 34$. Le NIST (National Institute of Standards and Technology) recommande d'utiliser des clés ayant au moins 112 bits comme niveau sécurité[75]. Par conséquent, la base recommandée pour la dissimulation doit être au minimum de $B = 31$ pour avoir un bon niveau de sécurité. Notons également que nous avons 128 bits de sécurité pour une base B égale à 34.

TABLE 3.7 – Evaluation du niveau de sécurité en fonction de la base B

B	31	**34**	37	40	43
$\boldsymbol{log_2(B!)}$ **Niveau de sécurité**	112	**128**	143	159	175

3.4.5 Discussion

Dans cette thèse, nous proposons un schéma stéganographique de distribution du secret résistant à la détection. Par rapport aux travaux connexes, l'extraction du secret suppose que les fichiers ne subissent aucune modification lors de la distribution du secret dans un environnement de stockage multi-cloud, en masquant l'existence du canal secret entre les parties communicantes. Comme le montrent les exemples 1 à 2, les données réparties dans les clouds diminuent au fur et à mesure que la valeur de la base choisie augmente. Nous avons distribué respectivement 16 et 4 valeurs avec les bases B = 2 et 17 dans les exemples 1 à 2, en considérant une taille secrète et un nombre de clouds fixes. On observe également que le choix de la valeur de la base a un impact sur le nombre et la taille des listes de fichiers nécessaires à la dissimulation du secret. Chaque liste doit contenir au moins B fichiers et le nombre de listes doit être identique au nombre de blocs obtenus après découpage du message secret représenté en base B. Dans l'exemple 1, quatre listes sont utilisées, puis une seule liste est utilisée dans le dernier exemple. De plus, les fichiers disponibles dans les listes ne sont pas pleinement exploités. La valeur de la base indique le nombre de fichiers à gérer à la fois pour l'insertion et l'extraction du secret. En effet, seuls 2 et 17 fichiers sont respectivement pris en compte dans chacune des listes des exemples 1 à 2. Un argument supplémentaire qui plaide en faveur du schéma de sécurité proposé est la possibilité d'utiliser n'importe quelle extension de fichier pour établir le canal secret tels que des images (.png, .jpg, ...), des fichiers binaires (.exe, .bin, ...) des fichiers textes (.txt, .doc, ...), ... Dans ce cas, les fichiers word, excel, powerpoint et pdf ont été utilisés tout en conservant leur intégrité. Une autre remarque importante doit être soulevée lors de l'utilisation de l'environnement de stockage multi-cloud. Il peut s'agir de plusieurs comptes disponibles auprès d'un ou de plusieurs fournisseurs de stockage. Dans les exemples de ce travail, nous avons utilisé des comptes de quatre fournisseurs différents tels que : iCloud, Dropbox, OneDrive et Google Drive.

Au final, une comparaison de ces exemples révèle que l'exemple 4 présente une meilleure répartition du secret dans les quatre clouds choisis. Un seul fichier est déposé dans chaque cloud, ce qui n'est pas le cas dans les autres exemples. Cela facilite également l'extraction du secret en réduisant le temps de recherche des fichiers disponibles dans les listes. De manière générale, le programme d'intégration prend en entrée la valeur de la base, le nombre de clouds et les répertoires contenant les listes de fichiers, puis sort les fichiers à uploader sur chaque cloud dans des répertoires. Alors que le programme d'extraction prend en entrée un ensemble de fichiers des clouds, la valeur de la base, le numéro des clouds et les listes, puis retourne le secret. Pour une meilleure

répartition du secret, un choix optimal doit être fait sur les deux paramètres suivants : le nombre de clouds et la valeur de la base.

Le Tableau 3.8 illustre la comparaison entre les méthodes de l'état de l'art et notre solution suivant plusieurs critères. Les propriétés suivantes, illustre la plus value de notre solution pour éviter la détection comparée aux autres :

- Le mécanisme de transfert du grand volume de fichiers se fait par l'intermédiaire du cloud ;
- La communication est indirecte entre l'émetteur et le destinataire (clouds) ;
- Le mapping entre le secret et le support utilise tout type de support (BD quelconque) ;
- Il y a aucune dépendance entre le support et le secret.

TABLE 3.8 – Quelques éléments de comparaison entre les différentes méthodes

	Nombre de fichiers stégo	**Type de Com.**	**Type de support**	**Support et secret**
Zhou et al.	128	Directe	Identique	Corrélation
Zhen et al.	56	Directe	Identique	Corrélation
Wu et al.	3	Directe	Identique	Corrélation
Abdulsattar et al.	1	Directe	Identique	Corrélation
Notre solution	52	Indirecte	Variable	Indépendant

3.5 Conclusion

Dans ce chapitre, un nouveau modèle de communication sécurisé en stéganographie, transparent à tout attaquant et résistant à la détection et à l'extraction de secret a été proposé. Deux propriétés contribuent à atteindre ces objectifs : les fichiers ne subissent aucune modification tandis que la diffusion du secret dans l'environnement de stockage multi-cloud permet de masquer l'existence du canal secret entre les parties communicantes. Les travaux antérieurs cachent généralement des informations à l'intérieur des médias secrets. Dans ce travail, les médias secrets sont un pointeur vers l'information. Le fichier porte donc les informations sans être modifié et le seul moyen d'y accéder est d'avoir la clé. Les expériences réalisées ont montré que la distribution secrète dans les clouds diminue à mesure que la valeur de la base choisie augmente. De plus, le choix de la valeur de la base a un impact sur le nombre et la taille des listes de fichiers nécessaires à la dissimulation du secret. Un argument supplémentaire qui plaide en faveur du schéma de sécurité proposé est la possibilité d'utiliser n'importe quelle extension de fichier pour

établir le canal secret tout en maintenant leur intégrité. Bien que la méthode proposée permet de communiquer secrètement, il n'en demeure pas moins que l'on manipule un grand nombre de fichiers lors des transferts vers l'environnement multicloud. Pour réduire cette complexité, nous proposons une seconde méthode de dissimulation distribuée basée sur les séquences de requêtes HTTP qui fera l'objet du prochain chapitre.

Chapitre 4

Stéganographie distribuée basée sur des séquences de requêtes HTTP

4.1 Introduction

Les données distribuées cachées dans l'environnement de stockage multi-cloud proposée par Moyou et Ndoundam [76] consistent à transmettre les fichiers dans des clouds séparés tout en conservant leur intégrité. L'intérêt recherché est de répartir ces fichiers sur plusieurs clouds afin de masquer l'existence de la communication et aussi d'empêcher un attaquant d'établir un lien direct entre les personnes lors des échanges. Les limitations de ce schéma sont basées sur les opérations de chargement et de téléchargement de ces fichiers qui peuvent être coûteuses, notamment pour l'échange de données secrètes volumineuses. Alors, comment atteindre les mêmes objectifs de sécurité tout en réduisant la taille des clés et en effectuant des opérations simples lors du processus d'insertion et d'extraction du secret ? Ce chapitre met en lumière un nouveau canal secret, piloté par des opérations web en ligne qui ne laissent pas d'empreintes digitales. Les caractéristiques de conception de ce schéma sont basées sur des requêtes légères pour encoder le secret couplées à l'indépendance du support de couverture.

4.2 Modèle du canal secret

4.2.1 Présentation

L'idée de base est d'utiliser les opérations en ligne quotidiennes pour avoir une signature furtive indétectable. Ces opérations concernent la consultation de sites internet qui est d'une manière insoupçonnée et passe pratiquement inaperçue en cas de dissimulation d'informations secrètes. L'intérêt de ce nouveau canal secret léger est de réaliser des communications parfaitement indétectables sans l'échange habituel de fichiers multimédia stégo. L'expéditeur consulte les pages web et le destinataire contrôle le site web contenant ces pages web. Les URL des pages web sont balisées afin de masquer les informations lors de la navigation. Ainsi, la contribution se divise en trois points principaux :

– Un canal secret léger : le schéma emploie une opération simple à savoir l'usage des requêtes HTTP ;

– Indétectable par analyse stéganographique : la technique est basée sur la navigation internet habituelle ;
– Indépendant du média de couverture : la technique utilisée ne nécessite ni échange ni modification de fichiers multimédia.

Deux schémas basés sur une signature furtive sont proposés en utilisant les opérations de navigation web en ligne. Le premier schéma utilise des requêtes HTTP basées sur la dépendance à l'intérieur des séquences avec des permutations pour transférer le secret. A l'intérieur de chaque séquence, il existe une interdépendance entre les requêtes HTTP en raison des permutations des URL balisées. Concrètement, chaque adresse IP produit des séquences ordonnées de requêtes, alors que le deuxième schéma utilise des requêtes HTTP basées sur la dépendance entre les séquences et les applications pour transférer le secret. Le deuxième schéma produit un ensemble de séquences de requêtes, où dans chaque séquence, chaque adresse IP produit une seule requête HTTP. En effet, cela facilite la reconstruction du secret à partir des identificateurs d'adresses IP et par conséquent rend les requêtes au sein d'une séquence indépendantes.

4.2.2 Notations et hypothèses

Ces notations sont utiles à la fois pour l'insertion et l'extraction du secret :
– S : un message secret en binaire ;
– B : la valeur de la base ;
– π : une permutation d'ordre n telle que $\pi = (\pi_0, \pi_1, ..., \pi_{n-1})$;
– r : le rang d'une permutation de n éléments ;
– U : une liste de n URLs telle que $U = (U_0, U_1, ..., U_{n-1})$;
– P : une liste de m adresses IP telles que $P = (p_0, p_1, ..., p_{m-1})$;
– $U_0, U_1..., U_{n-1}$: une séquence d'URL telle que :
  – $\forall i_1, i_2,\ 0 \leq i_1, i_2 \leq n-1$;
  – si $i_1 \neq i_2$ alors $U_{i_1} \neq U_{i_2}$.
– U^i : le i^e bloc de n URL sélectionné : $U_0^i, U_1^i, ..., U_{n-1}^i$;

4.2.3 Requêtes HTTP basées sur la dépendance à l'intérieur des séquences

4.2.3.1 Description

Dans ce premier schéma, les séquences de requêtes HTTP sont exécutées par l'expéditeur de manière méthodique afin de transférer les données secrètes. En effet, les requêtes au sein d'une séquence sont ordonnées et donc dépendantes les unes des autres. Le modèle de communication proposé dans la Figure 4.1 montre l'expéditeur qui transfère les données secrètes en faisant des séquences de requêtes HTTP au destinataire, qui dispose d'un ensemble de pages web accessibles en ligne. Les URL d'accès à certaines de ces pages web sont étiquetées et appelées URL taguées. Ces URL sont utilisées par l'expéditeur pour coder le secret tandis que le destinataire utilise l'adresse IP source de l'expéditeur ainsi que les URL taguées pour décoder le secret. A noter que les nombres

présents dans ces requêtes représentent les identifiants des URL accessibles. Dans la Figure 4.1, plusieurs types de requêtes sont appliqués aux URL. Dans l'ensemble des URL disponibles sur le serveur web, une courte liste de n URL est identifiée pour former les URL taguées. Les requêtes HTTP peuvent être effectuées sur des URL taguées ainsi que sur des URL non taguées (couleur blanche, voir Fig 4.1). Une distinction est faite entre les requêtes HTTP contenant des URL taguées, provenant de l'adresse IP de l'expéditeur (couleur noire, voir Fig 4.1) et des utilisateurs normaux (couleur blanche avec des lignes noires, voir Fig 4.1).

Lors de l'encodage du secret, l'expéditeur peut insérer des requêtes HTTP incluant des URL non taguées parmi celles qui sont taguées sans perturber la communication secrète. Il en va de même pour le côté récepteur qui enregistre les requêtes HTTP pour les URL taguées et non taguées provenant d'utilisateurs normaux. Les séquences continues de n requêtes HTTP pour les URL taguées permettent de masquer des bits secrets. Le récepteur filtre ces requêtes HTTP et ne retient que celles qui contiennent les URL taguées ayant l'adresse IP de l'expéditeur. Après ce filtrage, le récepteur apporte la sémantique aux URL sélectionnées. Pour coder et décoder les bits de données cachés, les fonctions de classement et de déclassement de Myrvold et al. sont utilisées [77]. Ces deux fonctions sont décrites dans la section suivante.

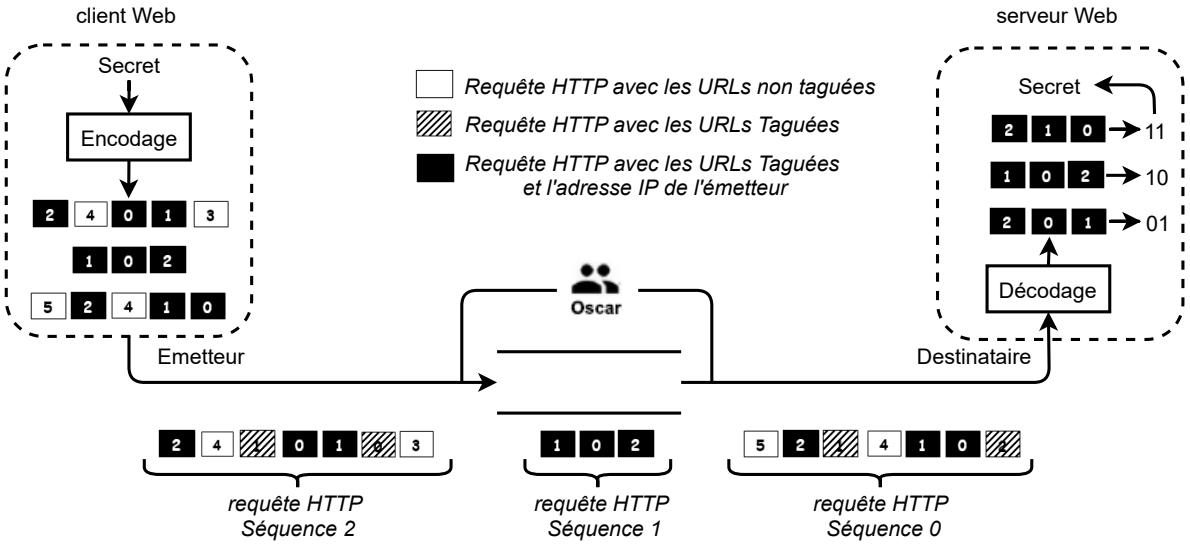


FIGURE 4.1 – Illustration du processus de dissimulation et d'extraction du secret

4.2.3.2 Fonctions de génération des permutations

Une permutation d'ordre n est un arrangement de n éléments. Ainsi, le nombre de permutations de n éléments est $n(n-1)...2.1 = n!$. Par conséquent, à chaque séquence de permutation $\pi_0, \pi_1..., \pi_{n-1}$ est associé un rang compris entre 0 et $n!-1$. Pour effectuer une telle indexation, nous utilisons une fonction de classement appelé *rank* et l'opération inverse nécessite la fonction de déclassement appelée *unrank*. En effet, pour chacune des $n!$ permutations de n symboles, la fonction de classement renvoie un entier dans l'intervalle $[0, n! - 1]$. Inversement, la fonction de déclassement prend en entrée un entier

compris entre 0 et $n!-1$ et renvoie la permutation de n symboles ayant ce rang. Certains travaux de la littérature [78] produisent ces permutations dans l'ordre lexicographique en temps $O(n^2)$. Les travaux de Myrvold et al. [77] qui sont pris en compte dans cet article, génèrent ces permutations en temps $O(n)$ mais pas dans l'ordre lexicographique, avec le temps d'exécution comme priorité. Leurs pseudo-codes respectifs sont décrits ci-dessous.

La fonction Unrank : pour déterminer la permutation de n symboles de rang r, nous devons d'abord initialiser le tableau π comme l'identité de permutation $\pi[i] = i$, pour $i = 0, 1, \cdots n-1$. Appeler ensuite la procédure unrank ci-dessous [77], avec les paramètres suivants : n, r et la permutation initiale π. Une fois la procédure terminée, la permutation attendue est disponible dans le tableau π. la fonction Echanger(a,b) permute les valeurs de a et b.

Procédure $unrank(n, r, \pi)$
Début
Si $n > 0$ **Alors**
$Echanger(\pi[n-1], \pi[r \bmod n])$;
$unrank(n-1, \lfloor r/n \rfloor, \pi)$;
fin ;
fin ;

La fonction Rank : pour déterminer le rang r associé à une permutation π de n symboles, on initialise d'abord le tableau π^{-1} comme ceci : $\pi^{-1}[\pi[i]] = i$, pour $i = 0, 1, \cdots, n-1$. Ensuite, la fonction rank est appelée avec les paramètres suivants : n, π et π^{-1}. Cela renvoie le rang associé à la permutation π.

Fonction $rank(n, \pi, \pi^{-1})$:Entier
Début
Si $n = 1$ **Alors** return(0) **fin** ;
$s := \pi[n-1]$;
$Echanger(\pi[n-1], \pi[\pi^{-1}[n-1]])$;
$Echanger(\pi^{-1}[s], \pi^{-1}[n-1])$;
retourner$(s + n.rank(n-1, \pi, \pi^{-1}))$;
fin ;

Les rangs correspondants pour les permutations de n= 4 sont illustrés dans le Tableau 4.1.

4.2.3.3 Illustration de l'encodage et du décodage du secret

Le nombre de permutations de n éléments est $n!$ et chaque permutation a un rang entre 0 et $n!-1$. En prenant $n = 3$ pour cette illustration, le nombre de permutations est de $3! = 6$. Nous énumérons tous les arrangements des éléments $0, 1, 2$. Par conséquent, chaque séquence de permutation de 3 éléments sera associée à un rang compris entre 0

TABLE 4.1 – Rangs et permutations pour n=4.

0 : 1 2 3 0	6 : 3 0 1 2	12 : 2 1 3 0	18 : 0 3 1 2
1 : 3 2 0 1	7 : 2 0 1 3	13 : 2 3 0 1	19 : 0 2 1 3
2 : 1 3 0 2	8 : 1 3 2 0	14 : 3 1 0 2	20 : 3 1 2 0
3 : 1 2 0 3	9 : 3 0 2 1	15 : 2 1 0 3	21 : 0 3 2 1
4 : 2 3 1 0	10 : 1 0 3 2	16 : 3 2 1 0	22 : 0 1 3 2
5 : 2 0 3 1	11 : 1 0 2 3	17 : 0 2 3 1	23 : 0 1 2 3

et 5. Le tableau 4.2 illustre les différentes permutations et leurs rangs respectifs. Ainsi, le codage du secret binaire 10 nécessite l'utilisation de la séquence : 1 0 2, car elle correspond au rang 2. De même, le décodage de la séquence 2 1 0, révèle la valeur secrète 3 qui est égale à 11 en binaire.

TABLE 4.2 – Rangs et séquences de permutation pour n = 3.

Rang	Permutation
0	1 2 0
1	2 0 1
2	1 0 2
3	2 1 0
4	0 2 1
5	0 1 2

4.2.3.4 Stégo clé

Deux paramètres doivent être partagés entre l'expéditeur et le destinataire :
- Une liste d'URL taguées : $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- Une liste d'adresses IP de l'expéditeur : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$.

4.2.3.5 Schéma d'insertion du secret

Dans ce premier schéma, le secret représenté en binaire est découpé en blocs. Chaque bloc binaire est encodé en sélectionnant une séquence d'URL. Cette séquence d'ouverture d'URL est déterminée en appliquant la fonction unrank à la valeur décimale de chaque bloc. L'expéditeur envoie les requêtes HTTP au serveur Web dans l'ordre prédéfini des URLs, c'est à dire en séquentielle. L'attente de la réponse à une requêtes HTTP permet de s'assurer que le secret est bien reçu au niveau du serveur. Il est donc possible de passer à la requête suivante. L'algorithme 4.1 présente la dissimulation du secret à travers la séquence de requêtes HTTP en séquentielle.

Algorithme 4.1: Algorithme de dissimulation dans les requêtes HTTP en séquentielle

Entrées : S : le message secret ;
p_0 : l'adresse IP de l'expéditeur ;
π : le tableau de permutation ;
U : la liste des URL taguées ;
Sorties : U' : les URL sélectionnées après la navigation ;

début
  Le secret S est représenté en binaire comme suit : $S = (z_{q-1}...z_1z_0)_2$, où $z_i \in \{0, 1\}$;
  Le nombre d'URL taguées est déterminé comme suit : $n = |U|$;
  La taille d'un bloc secret est évaluée comme suit : $l = \lfloor log_2(n!) \rfloor$;
  Le nombre de blocs est évalué comme suit : $k = \lceil q/l \rceil$;
  Le secret est découpé en k blocs de l bits : $S[(i \times l) + j]$, $0 \leq i \leq k-1$ et $0 \leq j \leq l-1$;
  pour *chaque bloc secret* $i = 0, 1, ..., k-1$ *:* **faire**
    Calculer la valeur décimale du bloc : $r = \sum_{j=0}^{l-1}(S[(i \times l) + j] \times 2^j)$;
    Initialiser le tableau π pour être l'identité de permutation $\pi[j] = j$, $j = 0, 1, \cdots n-1$;
    Déterminer la séquence de permutation $\pi_0, \pi_1..., \pi_{n-1}$ en appliquant la fonction unrank avec les paramètres suivants : n, r et π;
    Déterminer la séquence des URL : $U_{\pi_0}, U_{\pi_1}, ..., U_{\pi_{n-1}}$;
    pour *chaque URL* U_{π_j}, $j = 0, 1, ..., n-1$ *:* **faire**
      Envoyer des requêtes HTTP sur l'URL U_{π_j} avec l'adresse IP de l'expéditeur p_0 ;
      Récupérer la réponse HTTP de la requête;
    fin
  fin
fin

4.2.3.6 Schéma d'extraction du secret

Côté serveur, le récepteur enregistre les liens demandés par le client ainsi que les adresses IP associées. Ensuite, il trie les requêtes entrantes provenant d'une adresse IP source connue. Les liens URL associés sont extraits par ordre d'ouverture. La fonction rank est appliquée aux différentes séquences de liens afin de reconstituer le bloc secret correspondant. La programmation dynamique des sites web en langage python utilise ces deux instructions pour déterminer respectivement l'URL des requêtes HTTP et l'adresse IP distante des clients :

- $request.META.get('SCRIPT\_URI')$;
- et $request.META.get('REMOTE\_ADDR')$.

De plus, la fonction *time()* est utilisée pour déterminer l'heure à laquelle une URL est visitée. L'algorithme 4.2 présente l'extraction du secret.

Algorithme 4.2: Algorithme d'extraction du secret à travers les requêtes HTTP en séquentielle

Entrées : p_0 : l'adresse IP de l'expéditeur ;
U : la liste des URL taguées ;
π : le tableau de permutation ;
π^{-1} : le tableau de permutation inverse ;
Sorties : S : le message secret ;

1 **début**
2 Déterminer le nombre d'URL taguées : $n = |U|$; Déterminer le nombre de bits cachés dans une liste de n URL : $l = \lfloor log_2(n!) \rfloor$;
3 Capturer les URL dans l'ordre d'ouverture des requêtes HTTP ;
4 Extraire celles qui sont taguées appartenant à la liste U;
5 Filtrer uniquement les URL provenant de la seule adresse IP connue p_0 ;
6 Les URL obtenues sont subdivisées en k blocs de n URL : $U^0, U^1, ..., U^{k-1}$;
7 Initialiser S avec la chaîne vide : $S = \varepsilon$;
8 **pour** *chaque bloc* $U^i = U^i_{\pi_0}, U^i_{\pi_1}, ..., U^i_{\pi_{n-1}}$, $i = 0, 1, ..., k-1$ *:* **faire**
9 Initialiser le tableau π comme suit : $\pi = \{\pi_0, \pi_1, ..., \pi_{n-1}\}$;
10 Initialiser le tableau π^{-1} comme suit : $\pi^{-1}[\pi[j]] = j$, pour $j = 0, 1, \cdots, n-1$;
11 Calculer le rang r de la séquence : $\pi_0, \pi_1, ..., \pi_{n-1}$ avec les paramètres suivants : n, π et π^{-1};
12 Convertir la valeur r en binaire sur l bits : $S' = (z_{l-1}...z_1z_0)_2$;
13 Concaténer la chaîne binaire S' à S : $S = S'||S$, où le symbole $||$ désigne l'opération de concaténation ;
14 Retourner le secret S;
15 **fin**
16 **fin**

Remarque : La méthode proposée ci-dessus génère n requêtes HTTP pour chaque bloc de $log_2(n!)$ bits à l'aide de la même adresse IP. Les requêtes reçues du serveur peuvent devenir très élevées. Nous proposons dans la section suivante une méthode améliorée qui distribue les requêtes à travers plusieurs adresses adresses IP et dont le rapprochement entre les différentes adresses IP est difficile.

4.2.4 Requêtes HTTP basées sur la dépendance entre les séquences

4.2.4.1 Description

Dans ce second schéma, les séquences de requêtes HTTP effectuées par l'expéditeur sont dépendantes les unes des autres méthodiquement afin de transférer le secret. En revanche, les requêtes au sein d'une séquence ne sont pas ordonnées et donc indépendantes les unes des autres. Le modèle de communication illustré à la figure 4.7, montre comment le secret est distribué sur plusieurs adresses IP appartenant à l'expéditeur. Chaque participant cache sa valeur discrète en effectuant une requête HTTP sur l'URL étiquetée avec la valeur discrète attribuée comme index. Les URL taguées contenues dans ces requêtes HTTP sont indexées de 0 à $n-1$. Ces requêtes sont donc identifiées par l'adresse IP de l'expéditeur et l'URL taguée sélectionnée. Au cours du processus, chaque participant peut sélectionner des URL non balisées sans entraver l'intégration du secret. De même, les utilisateurs normaux peuvent sélectionner à la fois des URL taguées et non taguées sans aucun impact sur le processus de dissimulation. Côté serveur, le récepteur filtre les requêtes HTTP en ne conservant que celles contenant l'adresse IP de l'expéditeur et l'URL taguée sélectionnée. Ensuite, ces requêtes sont ordonnées selon la liste des adresses IP de l'expéditeur : ip_0, ip_1, ..., ip_{m-1}. Enfin, le secret est obtenu en extrayant, dans le même ordre, les index des URL taguées contenues dans ces requêtes HTTP.

La valeur ajoutée de ce schéma par rapport au premier est qu'aucune contrainte d'ordre séquentiel n'est établie lors de la sélection des URL par différentes adresses IP. Ils effectuent des requêtes HTTP indépendamment les uns des autres sans perturber le processus de décodage. De plus l'utilisation d'un VPN (Virtual Private Network) sur chaque adresse IP permet masquer le lien entre les différentes adresse IP en fournissant l'anonymat dans les communications. Enfin, la répartition du secret entre plusieurs adresses IP permet de réduire le nombre de requêtes HTTP par adresse IP pour éviter la détection.

4.2.4.2 Stégo clé

Trois paramètres doivent être partagés entre l'expéditeur et le destinataire :

- Une liste de n URL taguées : $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- Une liste de m adresses IP de l'expéditeur : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$;
- La valeur de la base B telle que : $B \geq 2$ et $B = |U| = n$.

4.2.4.3 Schéma d'insertion du secret

Dans ce deuxième schéma, l'expéditeur contrôle plusieurs ordinateurs. Ces machines clientes sont identifiées par leurs adresses IP respectives. Côté serveur, le récepteur contrôle n pages web. L'émetteur transcode ainsi le secret en base B, puis répartit les valeurs discrètes du secret obtenu entre les machines clientes. Chaque machine cliente

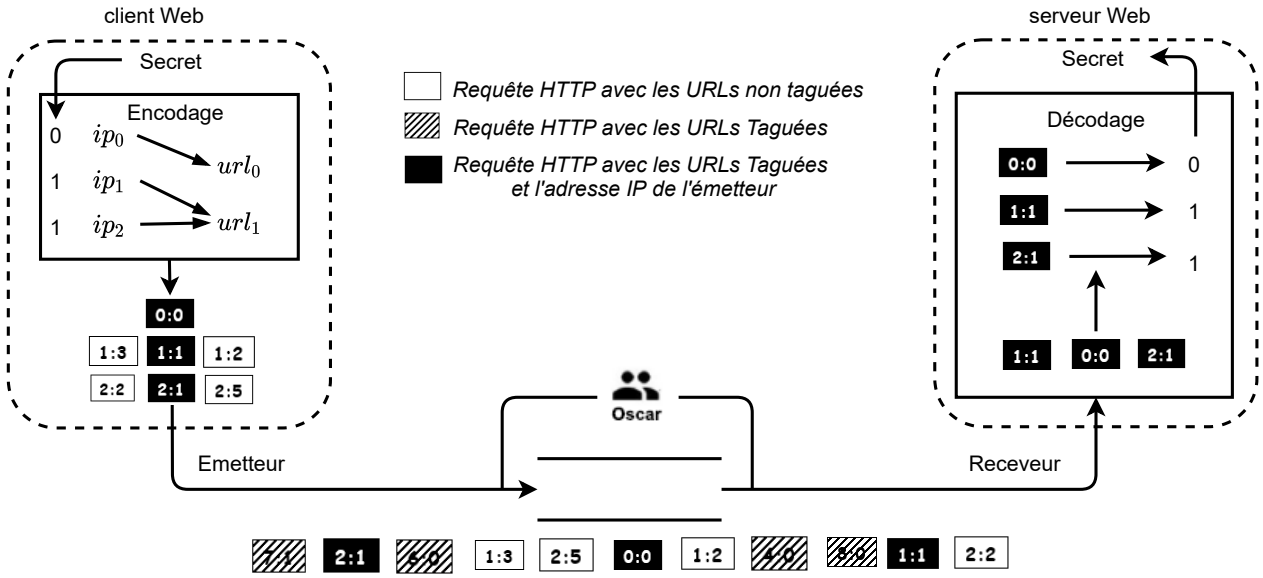


FIGURE 4.2 – Illustration du processus de dissimulation et d'extraction du secret

consulte l'URL qui est mappée à la valeur discrète attribuée. Les adresses IP effectuent ainsi les requêtes HTTP en parallèles. L'algorithme 4.3 dissimule le secret en parallèle à travers les différentes adresses IP de l'émetteur.

Algorithme 4.3: Algorithme de dissimulation du secret à travers les requêtes HTTP en parallèle

Entrées : S : le message secret ;
P_0 : l'adresse IP de l'expéditeur ;
U : la liste des URL taguées ;
B : la valeur de la base ;
Sorties : U' : les URL sélectionnées après navigation ;

début
  Le nombre d'adresses IP des expéditeurs est déterminé comme suit : $m = |P|$;
  Le nombre d'URL taguées est déterminé comme suit : $n = |U|$;
  Le secret S est transcodé en base B comme suit : $S = (z_{q-1}...z_1z_0)_B$, où $0 \leq z_i \leq B-1$ et $m \geq q$;
  Les adresses IP de l'expéditeur $p_0, p_1, ..., p_{q-1}$ envoient des requêtes HTTP respectivement aux URL $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$;
  pour $i = 0, 1, ..., q-1$ *:* **faire**
    Récupérer la réponse HTTP de chaque requête;
  fin
fin

4.2.4.4 Schéma d'extraction du secret

Le récepteur côté serveur lit les requêtes entrantes. Un filtre est ensuite appliqué pour extraire uniquement les URL taguées obtenues à partir des adresses IP sources connues. Ensuite, les URL sont triés dans l'ordre de numérotation des adresses IP sources de l'expéditeur. Puis, à chaque URL est associée sa position dans l'ensemble des URL balisées. Les valeurs discrètes ainsi obtenues sont assemblées pour former le secret en base B, qui sera ensuite converti en base 2 pour obtenir le secret initial. L'algorithme 4.4 présente l'extraction du secret à travers les requêtes HTTP en parallèle.

Algorithme 4.4: Algorithme d'extraction du secret à travers les requêtes HTTP en parallèle

Entrées : P : les adresses IP des expéditeurs ;
B : la valeur de la base ;
U : la liste des URL taguées ;
Sorties : S : le message secret ;

début
  Capturer les URL et les adresses IP sources dans l'ordre d'ouverture des requêtes HTTP ;
  Extraire uniquement celles qui sont taguées appartenant à la liste U ;
  Filtrer uniquement les URL de la liste d'adresses IP connue P ;
  Trier les URL dans l'ordre de numérotation des adresses IP sources de l'expéditeur : $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$;
  Faire correspondre chaque URL $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$ avec sa position respective : $z_0, z_1, ..., z_{q-1}$;
  Représenter les valeurs discrètes obtenues dans la base B : $S' = (z_{q-1}...z_1 z_0)_B$;
  Récupérer le secret S en convertissant S' en binaire;
fin

4.2.5 Analyse de la complexité en temps

4.2.5.1 Requêtes HTTP basées sur la dépendance à l'intérieur des séquences

Pour analyser la complexité en temps du premier schéma proposé, nous étudions d'abord la complexité en temps du processus d'insertion. Nous supposons que le secret S est réparti sur k séquences de n requête HTTP et que chaque séquence cache l bits. La représentation secrète en base 2 se fait en $O(log_2(S))$. La conversion de la valeur binaire d'un bloc en décimal se fait en $O(l)$. La détermination d'une séquence de n URL se fait en $O(n)$. La dissimulation du secret S en k séquences se fait en $O(k * (l + n))$. Si l est supérieur à n alors la complexité en temps du processus d'insertion est de $O(k * l)$ sinon elle est de $O(k * n)$.

Dans le schéma d'extraction du secret, nous supposons que le destinataire a filtré les m URL taguées à partir de l'adresse IP de l'expéditeur. La décomposition en k blocs de n URL se fait en $O(m)$. La détermination du rang associé à une séquence de n URL se fait en $O(n)$. La conversion du rang r en binaire sur l bits se fait en $O(log_2(r))$. Ensuite, la complexité en temps pour une séquence de requête HTTP est de $O(n)$. Enfin, la complexité en temps pour les k séquences de n URL est de $O(k * n)$. Si m est supérieur à $k * n$ alors la complexité en temps du processus d'extraction est de $O(m)$ sinon elle est de $O(k * n)$.

4.2.5.2 Requêtes HTTP basées sur la dépendance entre les séquences

Concernant la complexité en temps du second schéma de séquences de requêtes HTTP, nous supposons que le secret S est distribué en q valeurs discrètes dans la base B. Ensuite, dans le processus d'insertion, le secret est décomposé en k blocs de n valeurs discrètes. Pour n URL, l'envoi des requêtes HTTP se fait en $O(n)$. Par conséquent, la complexité en temps du schéma d'insertion pour k séquences de n URL est de $O(k*n)$.

De plus, le processus d'extraction du secret suppose que le destinataire a filtré les q URL taguées de l'expéditeur et les décomposer en k blocs de n URL. Ainsi, le tri des n URL selon les adresses IP se fait dans $O(nlogn)$. La correspondance entre les URL, leur index et leur représentation en base B se fait en $O(n)$. Par conséquent, la complexité en temps du processus d'extraction du secret pour k séquences de n URL est de $O(k * nlogn)$.

4.3 Evaluation

Cette section se concentre sur l'évaluation des deux schémas proposés en termes d'estimation du nombre de bits cachés et du mode de fonctionnement à travers des exemples numériques simples. Ensuite, les prochaines sous-sections présenteront l'analyse comparative, la discussion et enfin l'analyse de sécurité des schémas proposés par rapport à ceux existants.

4.3.1 Estimation des bits cachés

Dans la méthode proposée, un utilisateur considéré comme l'expéditeur identifié par son adresse IP, envoie des requêtes HTTP au serveur web, où exactement n URL sont étiquetées comme faisant partie de la clé du canal secret. Le nombre total de permutations sur ces n URL est de $n!$. Par conséquent, une séquence d'ouverture de n liens cache $\lfloor log_2(n!) \rfloor$ bits. Le nombre de bits cachés pour k séquences d'ouverture est : $k \times \lfloor log_2(n!) \rfloor$.

D'après la formule de Stirling [79], on a :

$$n! \sim \left(\frac{n}{e}\right)^n \times \sqrt{2\pi n} \tag{4.1}$$

Où $\pi = 3.14$ est l'aire du cercle de rayon en radius, $e = 2.718$ est la base du logarithme népérien et $\sim$ représente une égalité approximative.

$$\begin{aligned} log_2(n!) &\sim log_2(\left(\frac{n}{e}\right)^n \times \sqrt{2\pi n}) \\ &\sim nlog_2(\frac{n}{e}) + \frac{1}{2}log_2(2\pi n) \end{aligned}$$

Par conséquent, l'estimation des bits cachés pour k séquences est :

$$k \times \lfloor nlog_2(\frac{n}{e}) + \frac{1}{2}log_2(2\pi n) \rfloor \tag{4.2}$$

L'estimation du nombre de bits secrets de la seconde méthode est basée sur le principe selon lequel, le secret est distribué aux m ordinateurs clients de l'expéditeur. Ensuite, un mappage est effectué entre ces valeurs discrètes et les n URL taguées . À partir de là, chaque client peut faire n requêtes HTTP au serveur. Donc pour les m clients, n^m requêtes HTTP possibles. Enfin, le nombre de bits cachés sera :

$$\lfloor log_2(n^m) \rfloor = m \times \lfloor log_2(n) \rfloor \tag{4.3}$$

4.3.2 Exemples

Deux scénarios sont présentés ci-dessous pour chacune des méthodes stéganographiques proposées dans le but de montrer le processus d'insertion et d'extraction du secret.

4.3.2.1 Cas 1 : $S = (1110110110110111)_2$ et $n = 4$

Afin de décrire notre schéma de masquage de données proposé, des exemples numériques simples sont détaillés. Dans ces exemples, $S = (1110110110110111)_2$ un secret de 16 bits et l'adresse IP de l'expéditeur est p_0. Ensuite, un scénario est mit en évidence avec le nombre d'URL taguées prenant la valeur $n = 4$. Le Tableau 4.3 montre les quatre liens contrôlés par le récepteur. Le cas présente la répartition du secret entre les requêtes HTTP et explique en détail comment le secret est intégré et extrait en fonction de l'ordre des URLs sélectionnées. Notez que l'adresse IP de l'expéditeur et les URLs taguées sont partagées entre l'expéditeur et le destinataire.

TABLE 4.3 – Ensemble des URLs taguées et leur code d'identification

Code	URL
U_0	URL 0
U_1	URL 1
U_2	URL 2
U_3	URL 3
U_4	URL 4
U_5	URL 5
U_6	URL 6
U_7	URL 7
U_8	URL 8
U_9	URL 9

Dans cet exemple, les quatre entrées du Tableau 4.3 représentent les URLs taguées utilisées pour dissimuler les bits secrets. Tous les arrangements possibles de ces quatre liens et leurs rangs sont indiqués dans le Tableau 4.4.

Les étapes suivantes sont effectuées pour l'insertion du secret :

TABLE 4.4 – Vingt quatre permutations et leur rang

0	1 2 3 0	**6**	3 0 1 2	**12**	2 1 3 0	**18**	0 3 1 2
1	3 2 0 1	**7**	2 0 1 3	**13**	2 3 0 1	**19**	0 2 1 3
2	1 3 0 2	**8**	1 3 2 0	**14**	3 1 0 2	**20**	3 1 2 0
3	1 2 0 3	**9**	3 0 2 1	**15**	2 1 0 3	**21**	0 3 2 1
4	2 3 1 0	**10**	1 0 3 2	**16**	3 2 1 0	**22**	0 1 3 2
5	2 0 3 1	**11**	1 0 2 3	**17**	0 2 3 1	**23**	0 1 2 3

Etape 1 : le secret est déjà représenté en base 2 sur $m = 16$ bits : $s = (1111010110110111)_2$;

Etape 2 : déterminer le nombre d'URL balisées : $n = 4$;

Etape 3 : la taille d'un bloc secret est évaluée comme suit : $l = \lfloor log_2(n!) \rfloor = 4$;

Etape 4 : déterminer le nombre de blocs de quatre bits : $k = \lceil m/l \rceil = 4$;

Etape 5 : le secret est subdivisé en 4 blocs de 4 bits chacun :

Bloc #3	Bloc #2	Bloc #1	Bloc #0
1110	1101	1011	0111

Etape 6 : déterminer la valeur entière de chaque bloc :

Bloc #3	Bloc #2	Bloc #1	Bloc #0
14	13	11	7

Etape 7 : initialiser la permutation initiale dans le tableau $\pi = \{0, 1, 2, 3\}$:

Etape 8 : déterminer la séquence $\pi' = unrank(n, r, \pi)$ en fonction de chaque bloc. Le tableau **??** est utilisé pour déterminer les séquences d'URL selon le rang $7, 11, 13$ et 14.

	n	r	π	π'
Bloc #0	4	7	0 1 2 3	2 0 1 3
Bloc #1	4	11	0 1 2 3	1 0 2 3
Bloc #2	4	13	0 1 2 3	2 3 0 1
Bloc #3	4	14	0 1 2 3	3 1 0 2

Étape 9 : l'adresse IP de l'expéditeur, envoie les requêtes HTTP au destinataire dans l'ordre suivant :$U_2\ U_0\ U_1\ U_3$, $U_1\ U_0\ U_2\ U_3$, $U_2\ U_3\ U_0\ U_1$ et $U_3\ U_1\ U_0\ U_2$.

Les étapes suivantes doivent être effectuées pour récupérer le secret :

Etape 1 : déterminez le nombre d'URL balisées : $n = 4$;

Etape 2 : déterminer la taille d'un bloc secret : $l = \lfloor log_2(n!) \rfloor = 4$;

Etape 3 : capturer les URL des clients dans l'ordre des requêtes et filtrer uniquement celles qui sont taguées provenant de l'adresse IP de l'expéditeur : U_2, U_0, U_1, U_3, U_1 , U_0, U_2, U_3, U_2, U_3, U_0, U_1, U_3, U_1, U_0 et U_2 ;

Etape 4 : subdiviser la liste résultante en blocs de quatre URLs :

Bloc #0	Bloc #1	Bloc #2	Bloc #3
U_2, U_0, U_1, U_3	U_1, U_0, U_2, U_3	U_2, U_3, U_0, U_1	U_3, U_1, U_0, U_2

Etape 5 : initialiser le tableau π pour chaque séquence d'URL, avec l'index d'URL associé ;

Etape 6 : initialiser le tableau π^{-1} comme suit : $\pi^{-1}[\pi[i]] = i$, pour $i = 0, 1, 2, 3$;

Etape 7 : déterminer le rang r associé à chaque séquence d'URL : $r = rank(n, \pi, \pi^{-1})$

	n	π	π^{-1}	r
Bloc #0	4	2 0 1 3	1 2 0 3	7
Bloc #1	4	1 0 2 3	1 0 2 3	11
Bloc #2	4	2 3 0 1	2 3 0 1	13
Bloc #3	4	3 1 0 2	2 1 3 0	14

Etape 8 : convertir chaque rang en binaire sur 4 bits :

Bloc #0	Bloc #1	Bloc #2	Bloc #3
0111	1011	1101	1110

Etape 9 : concaténer les sous-chaînes binaires afin de récupérer le secret :

$$\begin{aligned} S &= Bloc\#3||Bloc\#2||Bloc\#1||Bloc\#0 \\ &= 1110||1101||1011||0111 \\ &= (1110110110110111)_2 \end{aligned}$$

4.3.2.2 Cas 2 : $S = (1110110110110111)_2$, $B = 4$, $m = 8$ et $n = 4$

Dans ce deuxième exemple appliqué sur des séquences de requêtes HTTP utilisant plusieurs adresses IP, le secret de 16 bits à cacher est $s = (1110110110110111)_2$. Les huit adresses IP contrôlées par l'expéditeur sont $P = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$. Les URLs taguées utilisées pour coder les bits secrets sont présentes dans le Tableau 4.3. ce cas utilise la base 4, 8 adresses IP et 4 URLs taguées. Les étapes pour intégrer le secret sont les suivantes :

Etape 1 : déterminer le nombre d'adresses IP : $m = 8$;

Etape 2 : déterminer le nombre d'URL : $n = 4$;

Etape 3 : représenter le secret dans la base $B = 4$:
$S = (1110110110110111)_2 = (32312313)_4$;

Étape 4 : distribuer les valeurs discrètes du secret aux huit adresses IP de droite à gauche :

$$\begin{array}{cccccccc} 3 & 1 & 3 & 2 & 1 & 3 & 2 & 3 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ ip_0 & ip_1 & ip_2 & ip_3 & ip_4 & ip_5 & ip_6 & ip_7 \end{array}$$

Etape 5 : remplacer les valeurs discrètes par les URLs taguées positionnées à ces index

$$\begin{array}{cccccccc} ip_0 & ip_1 & ip_2 & ip_3 & ip_4 & ip_5 & ip_6 & ip_7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ U_3 & U_1 & U_3 & U_2 & U_1 & U_3 & U_2 & U_3 \end{array}$$

Etape 6 : chaque adresse IP est utilisée pour formuler des requêtes HTTP sur les URLs attribuées. ip_0, ip_1, ip_2, ip_3, ip_4, ip_5, ip_6 et ip_7 effectuent une requête HTTP avec les URL respectives U_3, U_1, U_3, U_2, U_1, U_3,U_2 et U_3.

Le secret est extrait suivant les étapes suivantes :

Etape 1 : capturer les URLs et les adresses IP sources de l'expéditeur dans l'ordre des requêtes reçues et filtrer uniquement celles qui sont taguées provenant des adresses IP de l'expéditeur : (U_3,p_2), (U_3 , p_0), (U_3, p_7), (U_1,p_1), (U_2,p_3), (U_3,p_5), (U_1 ,p_4), (U_2,p_6).

Etape 2 : trier les URLs dans l'ordre de numérotation des adresses IP sources de l'expéditeur :

$$\begin{array}{cccccccc} ip_0 & ip_1 & ip_2 & ip_3 & ip_4 & ip_5 & ip_6 & ip_7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ U_3 & U_1 & U_3 & U_2 & U_1 & U_3 & U_2 & U_3 \end{array}$$

Step 3 : matcher ces URL avec leur index respectif :

$$\begin{array}{cccccccc} U_3 & U_2 & U_3 & U_1 & U_2 & U_3 & U_1 & U_3 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 3 & 1 & 3 & 2 & 1 & 3 & 2 & 3 \end{array}$$

Etape 4 : représenter les valeurs discrètes obtenues en base 4 de droite à gauche :
$S' = (32312313)_4$;

Étape 5 : convertir S' en binaire pour récupérer le secret :
$S = (32312313)_4 = (60855)_{10} = (1110110110110111)_2$.

4.3.3 Analyse comparative

Nous présentons dans cette sous-section les analyses comparatives pour démontrer l'efficacité des schémas stéganographiques proposés. Trois schémas sont pris en compte dans cette analyse à savoir le schéma 1, le schéma 2 et le schéma 3 désignant respectivement la méthode proposée basée sur la dépendance entre les requêtes HTTP incluses dans une séquence, la dépendance entre les séquences de requêtes HTTP et enfin la méthode récente de dissimulation des donnés secrètes dans un environnement de stockage multi-cloud [76]. Dans la suite, nous analysons quatre paramètres : l'influence du nombre d'URL taguées sur la capacité de bits secrets cachés et le nombre de séquences de requête HTTP et d'upload. De plus, le type d'opérations effectuées dans chaque séquence de requêtes et la taille des clés stégo sont comparés.

4.3.3.1 Comparaison de la capacité des bits secrets

Considérant un secret réparti sur une séquence de n URL étiquetées utilisant une seule adresse IP, la capacité des bits secrets du premier schéma est donnée dans le Tableau 4.5. De même, pour une séquence de n URL étiquetées employant m adresses IP, la capacité des bits secrets du second schéma est présentée dans le Tableau 4.6. La capacité des bits secrets du troisième schéma indiquée dans le Tableau 4.7 dépend de la valeur de la base B et du nombre de clouds c impliqués dans le processus de dissimulation des données.

En comparant ces trois tables, nous concluons que les capacités des bits secrets des schémas 2 et 3 sont identiques pour $n = B$ et $m = c$. De plus, le premier schéma a une capacité de bits secrets plus élevée que les deux autres schémas. En effet, les informations contenues dans les permutations augmentent rapidement par rapport à l'augmentation de l'espace du nombre de symboles n.

TABLE 4.5 – Capacité des bits secrets du premier schéma pour n URL taguées.

# d'URL taguées n	# permutations $n!$	Taille d'un bloc secret $log_2(n!)$	Capacité des bits secrets $\lfloor log_2(n!) \rfloor$
2	2	1	1
4	24	4,6	4
8	4,03 x 10^4	15,3	15
16	2,09 x 10^{13}	44,3	44
32	2.63 x 10^{35}	117,7	117
64	1,27 x 10^{89}	296,0	296

TABLE 4.6 – Capacité des bits secrets du deuxième schéma pour n URL taguées et m adresses IP.

# d'URL taguées n	# d'adresses IP m	# d'applications n^m	Taille d'un bloc secret $log_2(n^m)$	Capacité des bits secrets $\lfloor log_2(n^m) \rfloor$
2	1	2	1	1
4	2	16	4	4
8	4	4,09 x 10^3	12	12
16	8	4,29 x 10^9	32	32
32	16	1,21 x 10^{24}	80	80
64	32	6,28 x 10^{57}	192	192

TABLE 4.7 – Capacité des bits secrets du troisième schéma pour n clouds et la base B.

Base B	# de clouds c	# d'applications B^c	Taille d'un bloc secret $log_2(B^c)$	Capacité des bits secrets $\lfloor log_2(B^c) \rfloor$
2	1	2	1	1
4	2	16	4	4
8	4	4,09 x 10^3	12	12
16	8	4,29 x 10^9	32	32
32	16	1,21 x 10^{24}	80	80
64	32	6,28 x 10^{57}	192	192

4.3.3.2 Comparaison du nombre de séquences de requête/téléchargement HTTP

Pour cette comparaison, nous effectuons l'expérience avec 100 messages secrets aléatoires de 1024 bits et consignons la moyenne des résultats obtenus. Le nombre de séquences de requêtes HTTP est mesuré dans les schémas stéganographiques 1 et 2 en prenant les valeurs successives du nombre d'URL taguées $n = 2$, $n = 4$, $n = 8$ et $n = 16$. Les résultats sont tracés dans les Figures 4.3 et 4.4 pour un nombre d'adresses IP variant de 1 à 64. Alors que pour le troisième schéma stéganographique, nous mesurons le nombre de séquences de téléchargement de fichiers pour les valeurs successives de la base $b = 2$, $b = 4$, $b = 8$ et $b = 16$. La Figure 4.5 montre les résultats obtenus pour un nombre donné de clouds.

Une séquence de requêtes HTTP comprend la sélection de n URL . Un nombre précis de séquences de requêtes HTTP peut donc être nécessaire pour dissimuler des données secrètes. Lorsque ce nombre est réduit, le réseau devient moins surchargé. Le nombre de séquences de requêtes HTTP obtenues dans la Figure 4.3 appliquées au premier schéma diminue considérablement lorsque le nombre d'adresses IP utilisées augmente. Ce constat est le même pour les quatre courbes avec des valeurs différentes du nombre d'URL taguées n. Cependant, le nombre de séquences de requêtes HTTP devient constant (égal à 1) lorsque le nombre d'adresses IP est élevé. Par exemple, lorsque $n = 16$ URL

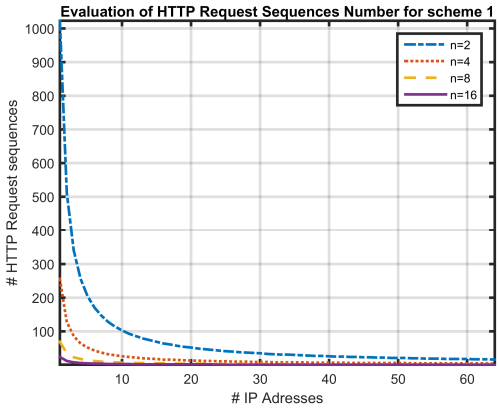


FIGURE 4.3 – Nombre de séquences de requêtes HTTP avec différentes valeurs de m et n pour le premier schéma.

tagués et le nombre d'adresses IP $m = 24$, chacune de ces adresses IP n'exécute qu'une seule séquence de requêtes HTTP. La formule (4.4) utilisée pour construire ces courbes confirme ce constat :

$$\text{Nombre de séquences de requêtes HTTP} = \left\lceil \frac{|(S)_B|}{log_2(n!) * m} \right\rceil, \tag{4.4}$$

Avec n le nombre d'URL taguées, m le nombre d'adresses IP, $|(S)_B|$ le nombre de valeurs discrètes obtenues après la représentation du secret S en base B et $\lceil x \rceil$ le plus petit entier supérieur ou égal à x. En effet lorsque m augmente, ce nombre approche 1. De même, le nombre de séquences de requêtes HTTP tend vers 1 lorsque le nombre d'URL augmente. Par exemple, en considérant une seule adresse IP et le nombre d'URL taguées $n = 2$, $n = 4$, $n = 8$, $n = 16$, nous avons respectivement $1023, 256, 69$ et 24 séquences de requêtes HTTP en moyenne.

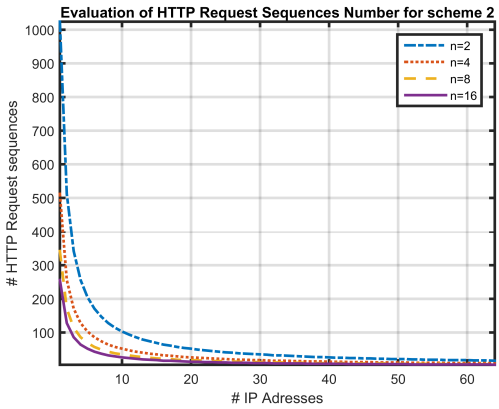


FIGURE 4.4 – Nombre de séquences de requêtes HTTP avec différentes valeurs de m et n pour le deuxième schéma.

Dans le cas du deuxième schéma stéganographique illustré à la Figure 4.4, le nombre

de séquences de requêtes HTTP devient petit à mesure que le nombre d'URL taguées augmente. La formule suivante est utilisée pour effectuer cette analyse :

$$\text{Nombre de séquences de requêtes HTTP} = \left\lceil \frac{|(S)_B|}{log_2(n^m)} \right\rceil, \tag{4.5}$$

Avec n le nombre d'URL taguées, m le nombre d'adresses IP et $|(S)_B|$ le nombre de valeurs discrètes obtenues après la représentation secrète S en base B. Par exemple pour $n = 4$ URL taguées et $m = 2$ adresse IP, le nombre moyen de séquences de requêtes HTTP est de 256. Par conséquent, avec ces paramètres, chaque adresse IP masque 2 bits par requête HTTP et donc 4 bits par séquence.

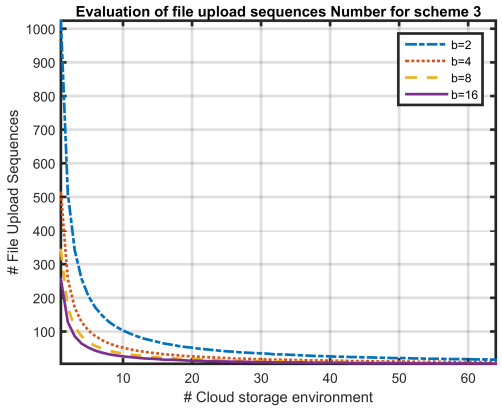


FIGURE 4.5 – Nombre de séquences de fichiers uploadés avec différentes valeurs de b et c pour le troisième schéma.

En revanche, le nombre de séquences d'uploads de fichiers présenté dans la Figure 4.5 du troisième schéma stéganographique basé sur la distribution du secret dans l'environnement multi-cloud, est identique au nombre de séquences de requêtes HTTP distribuées dans les URL taguées (Voir Fig. 4.4). En effet, un secret S représenté dans la base B est divisé en k séquences de c valeurs discrètes, avec c le nombre de clouds gérés. Ainsi, chaque valeur discrète est dissimulée dans un espace de stockage cloud séparé dans chaque séquence en uploadant un fichier multimédia. La formule (4.6) évalue le nombre de séquences d'uplaod de fichiers pour un secret S en base B :

$$\text{Nombre de séquences d'uploads de fichiers} = \left\lceil \frac{|(S)_B|}{log_2(B^c)} \right\rceil, \tag{4.6}$$

En comparant les formules (4.5) et (4.6), les graphiques de la Figure 4.4 et 4.5 sont similaires lorsque le nombre d'URL taguées est égal à la base ($n = B$) et lorsque le nombre d'adresses IP est égal au nombre de clouds gérés ($m = c$). En fin de compte, le premier schéma exécute le moins de séquences de requêtes par rapport aux deux autres schémas. Ceci peut être observé à travers la Figure 4.6, où les tracés des courbes des schémas 2 et 3 se chevauchent complètement.

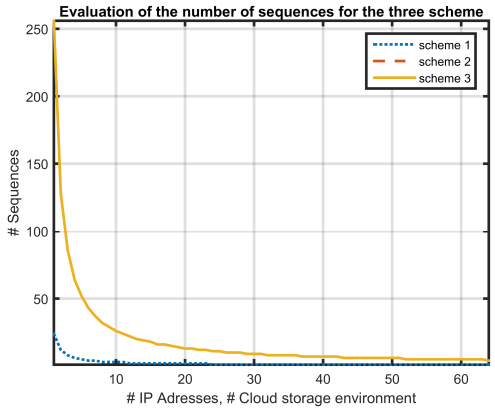


FIGURE 4.6 – Comparaison du nombre de séquences d'opérations sur les trois schémas pour n=B=16.

4.3.3.3 Comparaison du type d'opération

Les trois schémas exploitent les services populaires utilisés sur internet pour le processus de dissimulation de données afin de passer inaperçus. Les deux premiers schémas utilisent le service web via des requêtes HTTP lors de l'insertion et des calculs au niveau du serveur web pour l'extraction du secret. Tandis que le troisième exploite l'environnement de stockage des clouds pour uploader des fichiers lors de l'insertion et télécharger les mêmes fichiers lors de la phase d'extraction du secret.

En termes de comparaison, les requêtes HTTP sont plus simples et surchargent également très peu le réseau par rapport aux opérations de transfert de fichiers. De plus, les méthodes proposées dans les deux premiers schémas sont automatisées alors que pour le troisième schéma ce n'est pas le cas, c'est plutôt manuel. Parce que les utilisateurs ne disposent pas d'une interface de programmation à distance pour l'accès au stockage clouds. Sur la base de ces observations, les deux schémas proposés produisent un canal secret léger utilisant des séquences de requêtes HTTP.

4.3.3.4 Comparaison des clés stéganographiques

Les clés stéganographiques échangées entre les parties communicantes pour sécuriser les schémas de dissimulation des données sont importantes en fonction de leur taille. Plus ils sont simples, plus ils sont facilement mémorisés. Alors que lorsqu'elles sont complexes, se pose le problème de la conservation et de la confidentialité. Ainsi, le premier schéma gère deux listes de taille m et n comme clé stego représentant respectivement les URL taguées et les adresses IP utilisées. Dans le second schéma, deux listes de taille m et n, et un entier b forment la clé stego. Où, n, m et b représentant respectivement le nombre d'URL taguées, le nombre d'adresses IP et la base utilisée. Ces trois paramètres peuvent être réduits à deux car selon la méthode proposée, n est toujours égal à B. On peut donc utiliser l'un pour déduire l'autre. Concernant le troisième schéma, il y a quatre paramètres : la valeur de la base b, la liste des c clouds, les informations

d'authentification w (login et mot de passe) pour chaque espace cloud et les K listes de fichiers qui agissent comme pointeur vers les données secrètes. Hormis la base qui est un entier, les trois autres paramètres sont des listes de chaînes.

En fin de compte, les schémas 1 et 2 utilisent exactement deux listes comme clé tandis que le troisième utilise $k + 2$ listes et un entier. Par conséquent, les schémas 1 et 2 réduisent considérablement la taille des clés dans le processus de dissimulation des données.

4.3.4 Discussion

Dans cette thèse, nous proposons une dissimulation sécurisée des données basée sur des séquences de requêtes HTTP. Par rapport aux travaux connexes, l'insertion et l'extraction du secret supposent qu'aucun fichier n'est manipulé lors de la distribution du secret via les URL demandées, en masquant le canal secret entre les parties communicantes. Dans le premier schéma, le secret est réparti entre les différentes adresses IP, où chacune d'entre elles contribue à la réalisation des séquences de requêtes HTTP en fonction du nombre d'URL taguées. Notez qu'à l'intérieur de chaque séquence, il existe une interdépendance entre les requêtes HTTP en raison des permutations des URL taguées. Plus précisément, chaque adresse IP produit des séquences ordonnées de requêtes. D'autre part, le deuxième schéma produit un ensemble de séquences de requêtes, où dans chaque séquence une adresse IP produit une seule requête HTTP. En effet, cela facilite la reconstruction du secret à partir des identifiants d'adresse IP et rend par conséquent les requêtes au sein d'une séquence indépendantes. Dans ce cas, il y a plutôt interdépendance entre les séquences de requêtes HTTP. Comparativement, les deux schémas atteignent le même objectif mais avec un délai qui les différencie dans le séquencement des requêtes HTTP.

En comparant les deux schémas proposés, le 1er ayant une dépendance entre les requêtes HTTP au sein d'une séquence est plus lent en temps d'exécution car il suit un ordre d'exécution des requêtes pour transférer le secret. Cependant, il a une bonne capacité pour les bits secrets et un nombre réduit de séquences de requêtes HTTP. Au contraire, le deuxième schéma lève cette limite de temps avec l'absence de dépendance entre les requêtes HTTP à l'intérieur d'une séquence. Mais comme inconvénient la capacité des bits secrets est plus petite et le nombre de séquences de requêtes HTTP plus élevé.

Le Tableau 4.8 résume la comparaison fonctionnelle entre les deux méthodes proposées et la méthode de Moyou et Ndoundam [76]. Le point commun des trois schémas est qu'ils préservent complètement l'intégrité du média en tant que caractéristique de sécurité. Tandis que les deux schémas proposés réduisent considérablement la taille des clés stego sans manipuler les fichiers multimédias. Le nombre de séquences de requêtes générées dans le premier schéma est inférieur par rapport aux deux autres schémas. De plus, le premier schéma a une capacité de bits secrets plus élevée que les deux autres. Sur la base de cette synthèse, nous concluons que le premier schéma est très compétitif

TABLE 4.8 – Comparaison entre les méthodes de dissimulation de données proposées et celle récente.

Méthode	Capacité	# Séquences	Type de Requêtes	Média de Couverture	Taille des Clés	Intégrité des Média
schéma 1	élevée	faible	HTTP	URL	faible	Oui
schéma 2	faible	élevé	HTTP	URL	faible	Oui
Moyou et Ndoundam[76]	faible	élevé	Upload/ download	Fichier Multimédia	élevée	Oui

car en ce qui concerne les paramètres de comparaison, les performances sont meilleures, ce qui garantit une meilleure sécurité et capacité d'intégration.

4.3.5 Analyse de la sécurité

Les deux schémas proposés exploitent les opérations quotidiennes en ligne pour dissimuler des informations secrètes. Les modèles typiques cachent des informations dans les médias de couverture qui, en général, peuvent être analysées et détectées [68, 71]. Dans ce cas, rien n'est ajouté ou retiré du trafic Web. Seuls le nombre et l'ordre dans lequel les demandes sont envoyées diffèrent. Dans la figure 4.7, Alice envoie des séquences de requêtes HTTP à Bob, contenant des URL taguées. Alice peut insérer des URLs non taquées dans ces séquences de requêtes HTTP sans perturber la communication secrète. Cela permet de se concentrer sur tous les liens disponibles lors des échanges. Oscar qui analyse les communications se rendra compte que rien n'est ajouté ou modifié. Il est donc impossible pour un attaquant de détecter ou d'extraire des informations basées uniquement sur des requêtes HTTP de l'expéditeur.

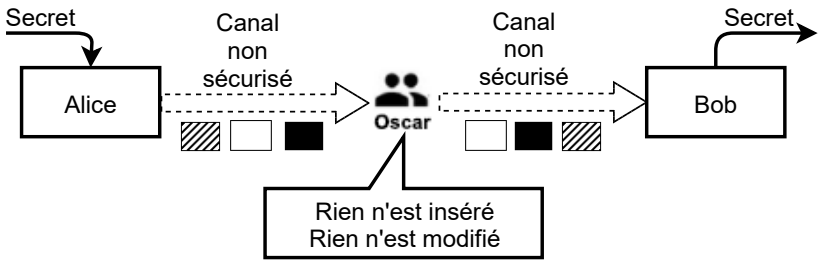


FIGURE 4.7 – Illustration du canal secret indétectable.

4.4 Conclusion

Dans ce chapitre, nous avons proposé deux nouveaux modèles de communication sécurisés, parfaitement piloté par des opérations web en ligne qui ne laissent pas d'empreintes digitales. Les caractéristiques de conception de ce schéma indétectable sont

basées sur des requêtes légères pour le codage secret couplées à l'indépendance du support de couverture. Le schéma proposé est simple et robuste quelle que soit la taille du secret à transmettre. Notre schéma diffère parfaitement des autres en ce qu'il ne modifie ni n'utilise de fichiers pour dissimuler des informations. Cela améliore considérablement le temps de calcul et la sécurité.

Concrètement, l'expéditeur consulte les pages Web et le destinataire contrôle le site Web contenant ces pages Web. Les URL des pages Web sont balisées afin de masquer les informations lors de la navigation. Le contrôleur de site apporte une sémantique aux pages sélectionnées en filtrant dans tous les liens sélectionnés, ceux qui sont tagués et qui proviennent de l'adresse IP connue de l'expéditeur. Ce travail s'inscrit dans le cadre de la recherche utilisant l'indirection pour préserver l'intégrité des communications.

Conclusion Générale

Rappels de la problématique

En stéganographie, la dissimulation du secret dans les supports doit se faire de la manière la pus imperceptible possible. C'est pourquoi les supports sont généralement modifiés pour intégrer les données confidentielles. Mais cette approche est aujourd'hui dépassée car la stéganalyse est capable de détecter les messages secrets dissimuler à l'intérieur de ces supports. Un nouveau courant de stéganographie a vu le jour pour évader les outils de stéganalyse appelé **coverless** (sans modification du support). L'idée est de sélectionner le support qui contient déjà tout le secret ou une partie du secret à transférer. Ainsi, le support original n'est pas modifié et ce dernier est envoyé au destinataire qui pourra utiliser une fonction de hachage robuste pour extraire le secret.

Nous nous sommes intéressés dans cette thèse aux mécanismes de prévention visant à empêcher toute violation de la confidentialité des données basée sur la stéganographie.

Ainsi, nous avons adressé les problématiques suivantes :

- une faible capacité d'embarquement ;
- la construction préalable de la base de données de fichiers permettant de transférer le secret. Les fichiers sélectionnés doivent avoir un lien avec le secret à envoyer ;
- un nombre élevé de fichiers stégo à transférer au destinataire ;
- le type de support utilisé est le même à chaque fois ;
- une communication directe entre les parties qui communiquent ce qui peut attirer l'attention ;
- la vulnérabilité à la distorsion (altération) du support.

Contributions

Dans le but de résoudre le problème de recherche défini dans cette thèse, puis de proposer des solutions efficaces pour palier aux limites des méthodes existantes, nous proposons de nouvelles méthodes pour la dissimulation du secret en considérant non seulement l'intégrité du support de couverture mais aussi les services de données en ligne. Ces considérations mettent en lumière l'intérêt en cours sur le camouflage des données sans modification du support de couverture. L'objectif principal est de définir une nouvelle approche pour faire correspondre les blocs du secret avec des fichiers de données sans compromettre le secret en cas de d'altération du fichier pendant le transfert. Ce qui est différent de la récente approche de la littérature basée sur le coverless qui s'appuie sur les fonctions de hachage pour faire correspondre le secret au contenu du fichier, ne permettant pas de garantir l'extraction de la totalité du secret en cas d'altération du support. Ces méthodes sont principalement limitées par le fait que malgré la transmission des données sans modification du support, l'analysent de la robustesse lors de la transmission du secret montre des taux d'erreurs non nuls en cas d'altération du support lors de l'extraction du secret. Dans le but d'assurer une transmission du secret et son extraction dans son entièreté, nous allons considérer que le secret n'est plus lié au contenu du fichier mais plutôt à une indexation des fichiers stockés dans des répertoires. Ainsi, les trois contributions majeures sont les suivantes :

1. **L'indexation des supports de couverture** : Nous proposons deux méthodes de dissimulation basées sur l'indexation des supports. La première méthode utilise les supports de fichiers multimédia. A chaque support sera associé un index ou un numéro en fonction d'un ordre de classement des fichiers à l'intérieur des répertoire. Cet ordre peut être par exemple : le classement des fichiers par nom, par type ou par date de création. Cela va permettre de faire correspondre chaque bloc du secret au moment de la dissimulation à un index de fichier et non plus à son contenu. Tandis que la seconde méthode utilise comme support les URL. Une liste de n URL d'un site web est sélectionnée pour cacher le secret. Ainsi, nous faisons correspondre chaque bloc du secret à une séquence d'URL en utilisant les permutations d'URL. Cela est possible grâce à l'utilisation de deux fonctions : Rank qui détermine la permutation de n URL connaissant son numéro (secret) et Unrank pour effectuer l'opération inverse. En somme, l'indexation de fichiers permet de garantir que le secret à transférer n'a pas de corrélation en terme de contenu avec les fichiers utilisés comme support lors de la dissimulation.

2. **Extensible à tout type de format de fichiers** : Avec l'indexation, un format de fichier précis n'est pas obligatoire. Car, les méthodes existantes fondent leur dissimulation soit sur un seul type de format de fichiers par exemple les images. Aini, la nature du format du fichier importe peu, puisqu'il sera possible de combiner à la fois les images, les documents textes, videos, audio ou même les URL en leur associant un numéro d'ordre. Cela crée une rupture avec l'existant en terme de format de fichier et contribue à rendre les méthodes de dissimulation imperceptible et qui n'attire pas l'attention par rapport aux schémas classiques.

3. **L'indirection entre l'émetteur et le destinataire** : Le secret une fois dissimulé dans le support n'est pas directement transféré au destinataire. Un intermédiaire est placé entre l'émetteur et le destinataire pour couper tout lien direct entre les deux et ainsi effacer toute trace de communication. Car s'il est établi qu'il y a communication, les conséquences possibles sont la destruction des supports ou l'interruption de la communication par quelque moyen que ce soit. La première méthode proposée utilise comme intermédiaire le stockage du cloud à travers le partage de fichiers entre les communicants. L'émetteur stocke des fichiers et les partage au destinataire. Ce dernier peut y avoir accès en utilisant le même compte (si l'émetteur à partagé ses accès avec lui) ou à travers son compte personnel. Tandis que la seconde méthode utilise un serveur web à travers l'ouverture des pages web chez l'émetteur pour transférer le secret et l'extraction du secret au niveau du serveur web.

Résultats obtenus

Les principaux résultats obtenus sont les suivants :

– la communication est imperceptible par conservation de l'intégrité des supports ;

– la diffusion du secret et le masquage du canal secret ;

– les communications sont robustes aux attaques par distorsion.

Perspectives

Les travaux futurs consisteront à tenir compte des aspects suivants :

– diminuer la taille de la base de données de fichiers nécessaire à l'encodage du secret. L'expérimentation a montré qu'il faut dans chaque dossier 1 048 576 fichiers pour dissimuler 20 bits par fichier. Cette estimation est élevée. Proposer une méthode qui diminue à la fois ce nombre de fichiers par dossier et augmente la capacité d'embarquement demeure un challenge.

– diminuer le nombre de fichiers déplacés dans l'environnement de stockage multi-cloud lors de la dissimulation. En effet selon l'expériemention, il a fallu deplacer 52 fichiers vers les espaces de stockage des clouds pour un secret de 1024 bits. La méthode sera plus performante si ce nombre de fichiers à déplacer se rapproche de 1.

– entrainer une IA sur les opérations d'upload de fichiers dans le but de montrer que ces opérations sont indistinguables de celles effectuées lors de la dissimulation du secret dans les différents clouds. En effet, l'idée est de comparer les opérations de manipulation des fichiers lors de la dissimulation et les opérations classiques utilisateurs dans le cloud. Et de montrer que ces deux types de manipulations sont indistinguables et par conséquent indétectables.

Références Bibliographiques

[1] D. Craigen, N. Diakun-Thibault, and R. Purse, "Defining cybersecurity," *Technology Innovation Management Review*, vol. 4, no. 10, 2014.

[2] D. R. Steve Kremer, Ludovic Mé and V. Roca, *Cybersecurity Current : challenges and Inria's research directions*. Inria white book, 2019.

[3] M. Gayathri and V. Preethi, "Study : Cryptography for information security," in *AIP Conference Proceedings*, vol. 3000, AIP Publishing, 2024.

[4] P. Aberna and L. Agilandeeswari, "Digital image and video watermarking : methodologies, attacks, applications, and future directions," *Multimedia Tools and Applications*, vol. 83, no. 2, pp. 5531–5591, 2024.

[5] J. Guaña-Moya, Y. Borja-López, G. Gutiérrez-Constante, P. Jaramillo-Flores, and O. Basurto-Guerrero, "Information security vulnerabilities using steganography as the art of hiding information," in *International Conference on Information Technology & Systems*, pp. 107–116, Springer, 2024.

[6] Z. Zhou, H. Sun, R. Harit, X. Chen, and X. Sun, "Coverless image steganography without embedding," in *International Conference on Cloud Computing and Security*, pp. 123–132, Springer, 2015.

[7] S. Zheng, L. Wang, B. Ling, and D. Hu, "Coverless information hiding based on robust image hashing," in *International conference on intelligent computing*, pp. 536–547, Springer, 2017.

[8] Z. Zhou, Y. Mu, and Q. Wu, "Coverless image steganography using partial-duplicate image retrieval," *Soft Computing*, vol. 23, no. 13, pp. 4927–4938, 2019.

[9] F. S. Abdulsattar, "Towards a high capacity coverless information hiding approach," *Multimedia Tools and Applications*, vol. 80, no. 12, pp. 18821–18837, 2021.

[10] L. M. Metcheka and R. Ndoundam, "Distributed data hiding in multi-cloud storage environment," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–15, 2020.

[11] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*, pp. 51–67, Springer, 1984.

[12] S. G. R. Ekodeck and R. Ndoundam, "Pdf steganography based on chinese remainder theorem," *Journal of Information Security and Applications*, vol. 29, pp. 1–15, 2016.

[13] A. K. Sahu and G. Swain, "Reversible image steganography using dual-layer lsb matching," *Sensing and Imaging*, vol. 21, no. 1, pp. 1–21, 2020.

[14] S. Jiang, D. Ye, J. Huang, Y. Shang, and Z. Zheng, "Smartsteganogaphy : Lightweight generative audio steganography model for smart embedding application," *Journal of Network and Computer Applications*, vol. 165, p. 102689, 2020.

[15] U. Pilania and P. Gupta, "Analysis and implementation of iwt-svd scheme for video steganography," in *Micro-Electronics and Telecommunication Engineering*, pp. 153–162, Springer, 2020.

[16] R. Bohme and R. Bèohme, *Advanced statistical steganalysis*, vol. 284. Springer Berlin, 2010.

[17] K. N. Choudry and A. Wanjari, "A survey paper on video steganography," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 2335–2338, 2015.

[18] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography : Survey and analysis of current methods," *Signal processing*, vol. 90, no. 3, pp. 727–752, 2010.

[19] T. Rabie and I. Kamel, "High-capacity steganography : a global-adaptive-region discrete cosine transform approach," *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 6473–6493, 2017.

[20] M. Dalal and M. Juneja, "Evaluation of orthogonal and biorthogonal wavelets for video steganography," *Information Security Journal : A Global Perspective*, vol. 29, no. 1, pp. 40–50, 2020.

[21] D. Kahn, *The Codebreakers : The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996.

[22] S.-J. Corke, *US covert operations and Cold War strategy : Truman, secret warfare and the CIA, 1945-53*. routledge, 2007.

[23] M. Gardner, *Codes, ciphers and secret writing*. Courier Corporation, 1984.

[24] M. Dalal and M. Juneja, "Steganography and steganalysis (in digital forensics) : a cybersecurity guide," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5723–5771, 2021.

[25] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, *Information hiding in communication networks : fundamentals, mechanisms, applications, and countermeasures*. John Wiley & Sons, 2016.

[26] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in dna microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, 1999.

[27] S. Neuner, A. G. Voyiatzis, M. Schmiedecker, S. Brunthaler, S. Katzenbeisser, and E. R. Weippl, "Time is on my side : Steganography in filesystem metadata," *Digital Investigation*, vol. 18, pp. S76–S86, 2016.

[28] W. Mazurczyk, M. Karas, and K. Szczypiorski, "Skyde : a skype-based steganographic method," *arXiv preprint arXiv :1301.3632*, 2013.

[29] P. Kopiczko, W. Mazurczyk, and K. Szczypiorski, "Stegtorrent : a steganographic method for the p2p file sharing service," in *2013 IEEE security and privacy workshops*, pp. 151–157, IEEE, 2013.

[30] W. Frączek, W. Mazurczyk, and K. Szczypiorski, "Hiding information in a stream control transmission protocol," *Computer Communications*, vol. 35, no. 2, pp. 159–169, 2012.

[31] I. Grabska and K. Szczypiorski, "Steganography in long term evolution systems," in *2014 IEEE Security and Privacy Workshops*, pp. 92–99, IEEE, 2014.

[32] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding : Steganography and Watermarking-Attacks and Countermeasures : Steganography and Watermarking : Attacks and Countermeasures*, vol. 1. Springer Science & Business Media, 2001.

[33] T. Jamil, "Steganography : the art of hiding information in plain sight," *IEEE potentials*, vol. 18, no. 1, pp. 10–12, 1999.

[34] R. T. Mercuri, "The many colors of multimedia security," *Communications of the ACM*, vol. 47, no. 12, pp. 25–29, 2004.

[35] S. Balu, C. N. K. Babu, and K. Amudha, "Secure and efficient data transmission by video steganography in medical imaging system," *Cluster Computing*, vol. 22, no. 2, pp. 4057–4063, 2019.

[36] A. Khalifa and A. Atito, "High-capacity dna-based steganography," in *2012 8th International Conference on Informatics and Systems (INFOS)*, pp. BIO–76, IEEE, 2012.

[37] I. J. Kadhim, P. Premaratne, P. J. Vial, and B. Halloran, "Comprehensive survey of image steganography : Techniques, evaluations, and trends in future research," *Neurocomputing*, vol. 335, pp. 299–326, 2019.

[38] R. J. Bagnall, "Reversing the steganography myth in terrorist operations : The asymmetrical threat of simple intelligence dissemination techniques using common tools," *SANS Information Security Reading Room*, vol. 19, 2002.

[39] B. News, "Steganography : how al-qaeda hid secret documents in a porn video," 2002, [En ligne] : http ://news.bbc.co.uk/2/hi/science/nature/2082657.stm. [Consulté le : 27-02-2021].

[40] Peerlyst, "Juan carlos, columbian drug trafficker," 2008, [En ligne] : https ://www.peerlyst.com/posts/using-digital-steganography-to-protect-national-security-information-ian-barwise-m-s-cissp-ceh-cnda. [Consulté le : 27-02-2021].

[41] D. News and Analysis, "Mumbai police fail to crack july 11 suspects' mail," 2009, [En ligne] :https ://www.dnaindia.com/mumbai/report-mumbai-police-fail-to-crack-july-11-suspects-mail-1058716. [Consulté le : 27-02-2021].

[42] N. Scientist, "Russian spy ring hid secret messages on the web," 2010, [En ligne] : https ://www.newscientist.com/article/dn19126-russian-spy-ring-hid-secret-messages-on-the-web/. [Consulté le : 27-02-2021].

[43] A. Technica, "Steganography : how al-qaeda hid secret documents in a porn video," 2012, [En ligne] : https ://arstechnica.com/information-

technology/2012/05/steganography-how-al-qaeda-hid-secret-documents-in-a-porn-video/. [Consulté le : 27-02-2021].

[44] C. W. Kurak Jr and J. McHugh, "A cautionary note on image downgrading.," in *ACSAC*, pp. 153–159, Citeseer, 1992.

[45] I.-S. Lee and W.-H. Tsai, "A new approach to covert communication via pdf files," *Signal processing*, vol. 90, no. 2, pp. 557–565, 2010.

[46] Z.-H. Wang, H.-R. Yang, T.-F. Cheng, and C.-C. Chang, "A high-performance reversible data-hiding scheme for lzw codes," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2771–2778, 2013.

[47] A. K. Sahu and M. Sahu, "Digital image steganography and steganalysis : A journey of the past three decades," *Open Computer Science*, vol. 10, no. 1, pp. 296–342, 2020.

[48] S. Khan, N. Ahmad, and M. Wahid, "Varying index varying bits substitution algorithm for the implementation of vlsb steganography," *Journal of the Chinese Institute of Engineers*, vol. 39, no. 1, pp. 101–109, 2016.

[49] R. Koikara, D. J. Deka, M. Gogoi, and R. Das, "A novel distributed image steganography method based on block-dct," in *Advanced Computer and Communication Engineering Technology*, pp. 423–435, Springer, 2015.

[50] A. Wibisurya *et al.*, "Distributed steganography using five pixel pair differencing and modulus function," *Procedia computer science*, vol. 116, pp. 334–341, 2017.

[51] A. Gutub, N. Al-Juaid, and E. Khan, "Counting-based secret sharing technique for multimedia applications," *Multimedia Tools and Applications*, vol. 78, no. 5, pp. 5591–5619, 2019.

[52] M. Al-Ghamdi, M. Al-Ghamdi, and A. Gutub, "Security enhancement of shares generation process for multimedia counting-based secret-sharing technique," *Multimedia Tools and Applications*, vol. 78, no. 12, pp. 16283–16310, 2019.

[53] A. Gutub and T. AlKhodaidi, "Smart expansion of target key for more handlers to access multimedia counting-based secret sharing," *Multimedia Tools and Applications*, pp. 1–29, 2020.

[54] T. AlKhodaidi and A. Gutub, "Trustworthy target key alteration helping counting-based secret sharing applicability," *Arabian Journal for Science and Engineering*, pp. 1–21, 2020.

[55] A. Gutub and K. Alaseri, "Hiding shares of counting-based secret sharing via arabic text steganography for personal usage," *Arabian Journal for Science and Engineering*, pp. 1–26, 2019.

[56] A. A.-A. Gutub and K. A. Alaseri, "Refining arabic text stego-techniques for shares memorization of counting-based secret sharing," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[57] A. Gutub and M. Al-Ghamdi, "Hiding shares by multimedia image steganography for optimized counting-based secret sharing," *Multimedia Tools and Applications*, pp. 1–35, 2020.

[58] A. Gutub and M. Al-Ghamdi, "Image based steganography to facilitate improving counting-based secret sharing," *3D Research*, vol. 10, no. 1, p. 6, 2019.

[59] X. Liao, Q.-y. Wen, and S. Shi, "Distributed steganography," in *2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 153–156, IEEE, 2011.

[60] J. Yang and X. Liao, "An embedding strategy on fusing multiple image features for data hiding in multiple images," *Journal of Visual Communication and Image Representation*, vol. 71, p. 102822, 2020.

[61] X. Liao, J. Yin, M. Chen, and Z. Qin, "Adaptive payload distribution in multiple images steganography based on image texture features," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[62] I. W. C. Easttom, "Method and apparatus of performing distributed steganography of a data message," Apr. 11 2017. US Patent 9,619,656.

[63] J. Mielikainen, "Lsb matching revisited," *IEEE signal processing letters*, vol. 13, no. 5, pp. 285–287, 2006.

[64] M. K. Khan, M. Naseem, I. M. Hussain, and A. Ajmal, "Distributed least significant bit technique for data hiding in images," in *2011 IEEE 14th International Multitopic Conference*, pp. 149–154, IEEE, 2011.

[65] T.-Y. Liu and W.-H. Tsai, "A new steganographic method for data hiding in microsoft word documents by a change tracking technique," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 24–30, 2007.

[66] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new synonym text steganography," in *2008 international conference on intelligent information hiding and multimedia signal processing*, pp. 1524–1526, IEEE, 2008.

[67] A. Malik, G. Sikka, and H. K. Verma, "A high capacity text steganography scheme based on lzw compression and color coding," *Engineering Science and Technology, an International Journal*, vol. 20, no. 1, pp. 72–79, 2017.

[68] Z. Yang, Y. Huang, and Y.-J. Zhang, "A fast and efficient text steganalysis method," *IEEE Signal Processing Letters*, vol. 26, no. 4, pp. 627–631, 2019.

[69] S. Samanta, S. Dutta, and G. Sanyal, "A real time text steganalysis by using statistical method," in *2016 IEEE international conference on engineering and technology (ICETECH)*, pp. 264–268, IEEE, 2016.

[70] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu, "A review on blind detection for image steganography," *Signal Processing*, vol. 88, no. 9, pp. 2138–2157, 2008.

[71] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimedia tools and applications*, vol. 77, no. 9, pp. 10437–10453, 2018.

[72] A. M. Abdullah *et al.*, "Advanced encryption standard (aes) algorithm to encrypt and decrypt data," *Cryptography and Network Security*, vol. 16, no. 1, p. 11, 2017.

[73] M. L. Mossebo Stephane and N. René, "Distributed data hiding in single cloud storage environment," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–13, 2021.

[74] J. Katz and Y. Lindell, *Introduction to modern cryptography : principles and protocols*. Chapman and hall/CRC, 2014.

[75] E. Barker, *Recommendation for Key Management : Part 1 – General*. NIST SP 800-57 Part 1 Rev. 5, 2020.

[76] L. Moyou and R. Ndoundam, "Distributed data hiding in multi-cloud storage environment," *Journal of Cloud Computing : Advances, Systems and Applications*, 2020, DOI : https ://doi.org/10.1186/s13677-020-00208-4.

[77] W. Myrvold and F. Ruskey, "Ranking and unranking permutations in linear time," *Information Processing Letters*, vol. 79, no. 6, pp. 281–284, 2001.

[78] C. T. Djamegni and M. Tchuenté, "A cost-optimal pipeline algorithm for permutation generation in lexicographic order," *Journal of Parallel and Distributed Computing*, vol. 44, no. 2, pp. 153–159, 1997.

[79] A. T. Benjamin, "Discrete mathematics : Elementary and beyond," 2004.

Publications

Article de Journal

1. **Leonel Moyou Metcheka** and Rene Ndoundam. Distributed data hiding in multicloud storage environment. *Journal of Cloud Computing*, Springer : pp. 1-15, 2020. DOI : https ://doi.org/10.1186/s13677-020-00208-4

Article de Conférence

1. **Leonel Moyou Metcheka**, Stephane Gael Ekodeck, Rene Ndoundam. Two secure online steganography schemes based on HTTP request sequences. *Proceedings of CARI 2022*, pp.1-12. HAL : https ://hal.archives-ouvertes.fr/hal-03706829

Moyou Metcheka and Ndoundam *Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:68
https://doi.org/10.1186/s13677-020-00208-4

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Distributed data hiding in multi-cloud storage environment

Leonel Moyou Metcheka[1,2,3] and René Ndoundam[1,2,3*]

Abstract

Classical or traditional steganography aims at hiding a secret in cover media such as text, image, audio, video or even in network protocols. Recent research has improved this approach called distributed steganography by fragmenting the secret message and embedding each secret piece into a distinct cover media. The major interest of this approach is to make the secret message detection extremely difficult. However, these file modifications leave fingerprints which can reveal a secret channel to an attacker. Our contribution is a new steganography paradigm transparent to any attacker and resistant to the detection and the secret extraction. Two properties contribute to achieve these goals: the files do not undergo any modification while the distribution of the secret in the multi-cloud storage environment allows us to hide the existence of the covert channel between the communicating parties. Information's are usually hidden inside the cover media. In this work, the covert media is a pointer to information. Therefore the file carries the information without being modified and the only way to access it is to have the key. Experiments show interesting comparison results with remarkable security contributions. The work can be seen as a new open direction for further research in the field.

Keywords: Distributed steganography, Multi-cloud environment, Multimedia files, Data integrity

Introduction

The rapid development of internet leads to an increase dependence in almost all areas of life. In the same way that it promotes communications, it also poses a threat both to users and to state institutions with the spying and theft of information. To tackle this annoyance, two security techniques are used: steganography and cryptography. The main difference between the two is the secret access mechanism. One goes unnoticed while the second is unreadable by transformations [1].

Steganography is mainly used for secret communications by setting up a cover channel [2]. Steganography is the art of writing secret data so that no one except the recipient is aware of the existence of the secret message [3]. Popular steganography techniques hide the secret in digital content such as text, image, video and audio files [4–7]. While other techniques insert the secret in the network protocols [8]. The study of steganographic systems is evaluated with three main characteristics: capacity, robustness and security. Capacity measures the volume of secret data that is hidden in the cover media. Robustness concerns resistance to destruction or modification of the hidden data (steganogram). While security assesses the ability of an eavesdropper to detect hidden information. Although, security is usually the most desirable characteristic [9].

A successful steganography depends on the carrier medium that does not raise attention [10]. This is why the best carrier choice for steganogram must be popular as well as transparent when inserting the steganogram [11]. Thus, the cloud is an ideal candidate for enabling secret communications. A number of steganographic techniques are used to transfer and secure the data stored in the cloud storage [12]. These techniques generally respond to the problems of confidentiality of user data stored in cloud servers [13].

These methods refer to classical steganography compared to distributed steganography [14] which aims to

* Correspondence: ndoundam@yahoo.com
[1]Team GRIMCAPE, Yaounde, Cameroon
[2]Sorbonne University, IRD, UMMISCO, F-93143 Bondy, France
Full list of author information is available at the end of the article

hide the secret in several covert medium instead of one. The distributed steganography has the merit of making difficult the secret message detection. This is only possible by a meticulous modification of each cover medium. However, these modifications leave fingerprints which can reveal a secret channel by an attacker [15, 16]. Moreover, the simple fact that the attacker knows that there exists an exchange between the two can raise suspicion. When these scenarios are realized the whole steganographic model collapses. Therefore, here is an overview of the usual features of classical steganography:

- The communicating parties are known: the files are sent from one to the other [4, 17, 18];
- The transferred files are altered for secret insertion: the embedding and extraction process use different approaches such as a special character like non breaking space(A0), punctuation marks, word synonyms and linguistic properties [4, 17–19];
- The exchanged files are commonly subject to steg analysis: the covert channel establishment will be detected [15, 16, 20–22]. Consequently, the deletion or modification of the covert media leads to the loss of the entire secret;

In this paper, our contribution consists in proposing a new distributed steganography scheme based on the processed files integrity. Thus, it guarantees that the exchanged files do not undergo any modification. This ensures a good security level with an undetectable secret communication. The technique used exploits file storage in several cloud service providers. This solution is an excellent tool useful for the organizations for data exfiltration in case of espionage or to keep secure the participants shared keys involved in a secret sharing. This work contributions are summarized as follow:

- The communicating parties are not known: there is no direct link between them during the communication process. One uploads files in the cloud storage while the second exploits these files;
- The transferred files are not altered for secret insertion: each file implicitly holds a part of the secret data;
- The exchange files are robust against steg analysis: the proposed technique focuses on maximum resiliency against secret detection and extraction.
- The ability to use any file extension to establish the covert channel while maintaining their integrity.

The rest of the paper is organized as follows: section 2 and section 3 study classical and distributed steganography respectively. The new distributed scheme is presented in section 4. Experimental results are done in section 5 and finally section 6 is devoted to the conclusion.

Classical steganography

In this section, we will revisit some shortcomings related to classical steganography.

In 1984, Simmons presents the interest of steganography through the prisoner's problem [23], where Alice and Bob plan to escape from prison. Their communication goes through the warden who searches for any hidden communication between the two and if he detects one, he will separate them and cause their failure to escape. Then, the two prisoners must use ingenuity to keep their communication undetectable. Alice and Bod must use a channel that is invisible to the warden. This secret channel is known as the covert channel. Figure 1 shows the security model applied to cryptography [24]. This communication conveys secret information in a manner that it can be observed by an adversary. It uses the unsecured regular communication channel.

The general process of steganography achieves unsuspecting communications with two current algorithms: the embedding algorithm which consists in altering the cover media (text, image, video, audio or network protocol) to insert the secret message using a key shared between the parties in communication. This produced a stego media. The second is the extracting algorithm which consists in dissociating the covert message from the media carrier according to the stego media and the secret key. The extraction is said to be reversible when the cover media obtained is identical to the initial.

The main concern of steganography is stealth, because if an attacker, passive or active, can detect the presence of the secret message, from there, he can try to extract it and to decrypt it, if it is encrypted. Thus, steganography techniques must focus on maximum resiliency against detection and extraction. In contrast, two unexpected actions can militate in favour of the detection and extraction of the message. The first is the attacker's ability to know the existence of a communication between Alice and Bob. The stego media is generally sent to Bob via an open or insecure channel. The simple fact that the attacker knows there exists an exchange between the two can raise suspicion. The second action is the attacker's ability to capture and to study in depth the content of the messages exchanged to reveal the existence of a covert channel. The attacker can achieve this objective by performing steg analysis and finally detects or extracts the secret message. The attacker can even go further by deactivating the hidden message so that the recipient cannot extract it and / or modify the hidden message to send incorrect information to the recipient [25].

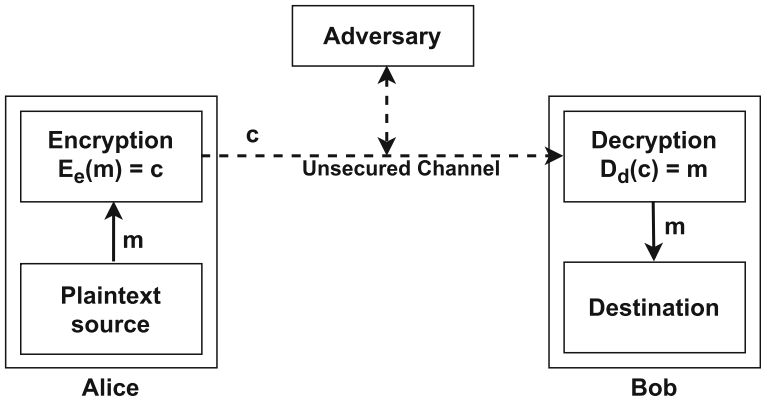


Fig. 1 The model of a two-party communication using encryption [24]

When these scenarios are realized the whole steganographic model collapses. These covert channel detection are motivated and carried out mainly by computer forensic examiners which collect evidence related to a past crime. But also by secret government and corporate services to prevent espionage.

Some steganography detection tools on image and audio files have been described using advanced statistical tests [26–29] such as higher-order statistics, Markov random fields, linear analysis, wavelet statistics, and much more [20]. Regarding data hiding in network communications, several studies are capable of detecting covert channels while others prevent all secret communications. Goudar's work [30] uses second-order statistical tools such as the adjacency histogram and the normalized adjacency histogram to detect secret communications. While Muawia's work [31] uses crafting and replaying packet tools to disable hidden messages.

Distributed steganography

In this section, we will revisit some shortcomings related to distributed steganography.

Distributed steganography [32, 33] is an improvement of classical steganography which aims at fragmenting a secret and then hidden in several covert media, which makes it more difficult to detect the whole secret message. This is applied when there are various independent senders and only one recipient. Thus the receiver acquires the union of distinct inputs. With the advent of cloud technologies, it is common nowadays for users to hide secret information in a mass of images and stored them in the cloud space. These stego images are then shared with the recipient to extract the secret. In general, these embedding algorithms deal with the issue of distributing the payload in a sequence of images to avoid detection. This explains the emergence of embedding strategies of payload distribution in multiple images by fusing multiple features to describe image complexity [14]. Other recent strategies are based on the image texture complexity and the distortion distribution as indicator for secure capacity of each cover image [34]. These strategies are applied on single image steganographic algorithms and experiments shown better resistance to modern universal pooled steganalysis compared to existing methods.

Xin Liao et al. [35] have proposed a model of the distributed steganography (see Fig. 2), where n sending parties $p_1, p_2, ..., p_n$ wishes to establish a covert channel with a receiver R. Each party only knows its covert message m_i while no one else apart from the receiver R can retrieve the combination of these secrets messages by receiving it through a public channel. The fact of protecting a secret data through several people is named as secret sharing [36]. Secret sharing schemes require three main phases: generation of the target key, distribution of the share keys to participants and reconstruction of the secret. The target key can thus be obtained from a set of n shares determined automatically by the dealer of the system. The latter distributes a share key to each participant via a private channel. Finally, the system combines the different share keys to access the system from the target key obtained [37]. This principle was originally proposed by Shamir [38] and Blakley [39]. Secret sharing is applied in critical areas requiring access controlled by multiple users such as rocket launching, opening of bank safes, proof correctness of electronic voting systems [40].

Improvements in secret sharing have been proposed for allowing access to a restricted number of participants, called (k, n) secret sharing. The k value is known as the secret share threshold and must be less than or equal to n. As a constraint, the

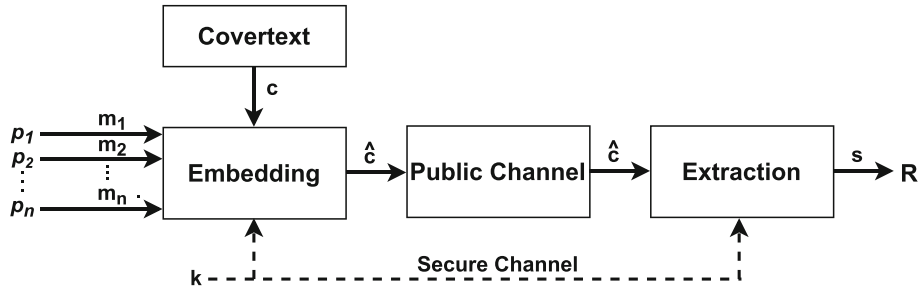


Fig. 2 The model of distributed steganography [34]

reconstruction of the target key requires at least k participants [41]. One of the approaches which is inspired by this principle is the counting based secret sharing [40], consisting in generating the share keys by replacing at various positions of the secret key one or two 0-bits by a 1-bit. Therefore, the reconstruction of the target key is obtained by adding bit by bit the share keys of the same position, so that the result bit is equal to 1 if the sum is equal to the value of the threshold k and 0 otherwise. This technique requires less computation as a strength, however it generates a reduced number of shares. Several optimizations have been made to guarantee the security of shared keys [42] as well as to enhance the number of generated shares [43, 44]. These schemes are excellent tools useful for cryptographic protocols. Moreover, they are also used in steganography to hide in a distributed manner the share keys of each participant. To this end, methods are provided for concealing the target key in specific covert media such as text [45, 46] or images [47, 48].

Distributed steganography is interesting because it makes detection task more complicated by spreading the secret in various media and store them in random places. However, simply modifying a media to incorporate a secret can raise attention for establishing a covert channel. This can be noted by performing steg analysis [49, 50]. More seriously, if the suspicious is proved to be true, the deletion or modification of a single covert media can lead to the loss of the entire secret.

To illustrate these limits we will take images considered as one of the most used media in steganography. If an algorithm can determine the presence of a secret message in a covert media then the whole steganographic system used is considered broken [28]. However, it is hardly practical for a steganalizer to know the algorithm used in all the variety of existing steganographic system. This is why universal methods emerge named as blind steganalisis, are capable of detecting any new or unknown embedding algorithm [15, 16]. The process is performed in two main phases: a training phase using original images with features extraction and a second phase for the images classification. This technique detects original images and stego images. It motivates us to develop a steganographic method perfectly undetectable with current steganalized methods by exchanging covert media without modifying them. In addition, the covert channel is capable of handling all types of covert media. Table 1 shows a comparison of steganographic techniques based on two criteria: the covert media type used as well as the modification of the cover media.

Table 1 Steganographic techniques comparaison based on the covert media type and covert media modification

Classical Steganography					
References	**Text**	**Image**	**Audio**	**Video**	**Covert Media Modification**
Liu et al. [51] Lee et al. [19] Ekodeck et al. [4] Khosravi et al. [17]	✓	×	×	×	✓
Sahu et al. [5] Su et al. [52]	×	✓	×	×	✓
Jiang et al. [6] Ali et al. [53]	×	×	✓	×	✓
Pilania et al. [7] Baziyad et al. [54]	×	×	×	✓	✓
Distributed Steganography					
References	Text	Image	Audio	Video	Covert Media Modification
Gutub et al. [45] Gutub et al. [46]	✓	×	×	×	✓
Yang et al. [14] Liao et al. [34] Gutub et al. [47] Gutub et al. [48]	×	✓	×	×	✓
Our proposal	✓	✓	✓	✓	×

The comparison ends with the positioning of the method to be proposed.

The proposed scheme

The proposed covert channel is a new paradigm, transparent to secret communication between the two parties. The solution uses the cloud storage space to store files. The uploaded files undergo no alteration, the information associated with each of them is their classification order in a list of files. Thereby, the file selection and uploading to the cloud depends on the secret to be shared with the receiver.

The original idea of steganography was proposed by Simmons [23]. The basic idea was to hide secret data inside the cover media to go unnoticed. Related works [4–7] based the design of their steganographic schemes on this idea.

The original contribution is the proposal of a steganographic scheme where the files used as covert media carry the information without being modified. The cover media is a pointer to secret data. This secret data is found in the key. This key consists of the following sets: the list of clouds and login credentials, the lists of files and the base used. The sender and the receiver exchange this key before initiating their secret communication. The keys exchange can be done in a physical meeting or using encrypted communications. As a concrete example, a secret agent C is employed by his origin country A. During a physical meeting in the country's government agency A, the agency officials give to the secret agent a removable memory which contains the key. Country A sends the secret agent in the country B. During his spy mission in country B, the secret agent sends confidential information from country B to country A.

Overview

The communication model proposed in Fig. 3, shows the sender and the receiver sharing identical lists of documents. The sender transcodes the secret in a specific base and group them into k blocks of n values: $b_0, b_1, ..., b_{k-1}$. For each value of a given bloc, the files at this index value are sent to the cloud. The process is repeated for every secret block. To each block is assigned a new list. The receiver having the same access to each cloud, browses them to recover the saved files. Thereafter, he reconstructs the secret from these file positions found in the lists. The originality of this scheme is the opponent disability to make any communication link between the sender and the receiver.

The covert channel model

The covert object

These are any extension files, selected to be dropped in multiple clouds storage environment. The cloud storage providers used are named as: $c_0, c_1, ..., c_{n-1}, n \geq 2$.

The embedded message

Any message format can be concealed. The preliminary step requires that the secret be encoded in a given base.

The key

Three elements are shared between the sender and the receiver:

- The cloud order $c_0, c_1, ..., c_{n-1}$;
- The authentication accounts (user name and password) for cloud access named as: $w_0, w_1, ..., w_{n-1}$;
- A set of disjointed lists $L^{(0)}, L^{(1)}, ..., L^{(k-1)}$, where each list i contains exactly B files: $L_0^{(i)}, L_1^{(i)}, ..., L_{B-1}^{(i)}$, $i =$

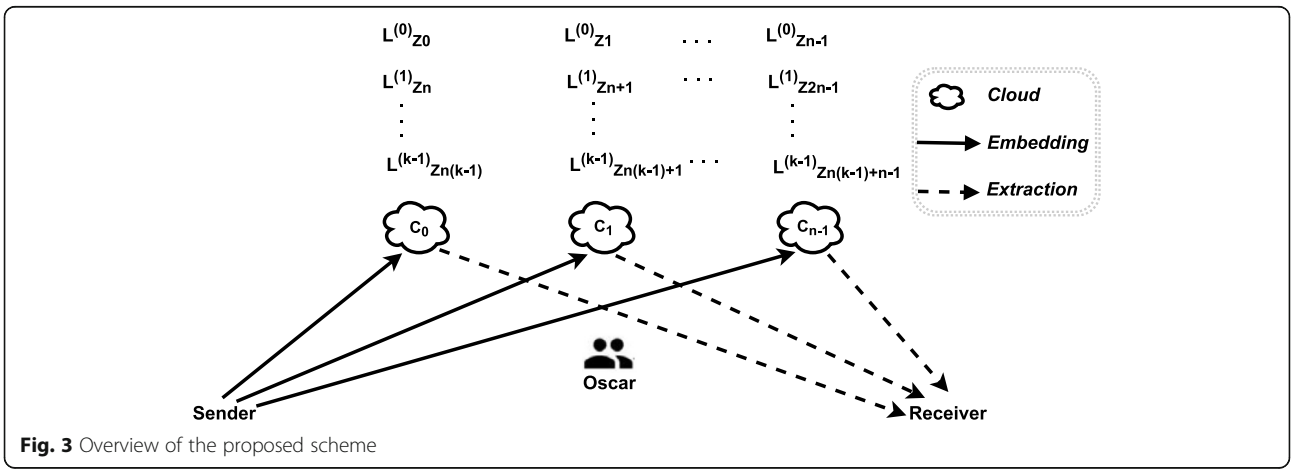


Fig. 3 Overview of the proposed scheme

0,1, ..., k-1. These files can take any format type such as text, image, audio, video, application, archive, ...
- The base B such that: $|L^{(0)}| = |L^{(1)}| = \ldots = |L^{(k-1)}| = B$.

Notations and hypothesis

These notations are useful for the embedding algorithm as well as for the secret extraction:

- s: the input secret formatted in base *2* or *10*;
- B: the base used such that: $B \geq 2$;
- $(z_{q-1} \ldots z_1 z_0)_B$ is the secret representation in base B;
- *Mat [i]*: is the i^{th} block of the secret;
- *Mat [i,j]*: is the value at position j of the block number i;
- n: is the number of clouds handled;
- k: is the secret bloc number or the number of lists used;
- $L^{(i)}$: is the i^{th} files list. Each secret block uses a distinct list i, $0 \leq i \leq k\text{-}1$;
- $L_j^{(i)}$: is the j^{th} file in the list number i, $0 \leq i \leq k\text{-}1$ *and* $0 \leq j \leq B\text{-}1$;

Here are hypothesis of the proposed scheme:

- Lists:$L^{(0)}$, $L^{(1)}$, ..., $L^{(k-1)}$ are disjointed:
  i_1, i_2 designating Lists, $0 \leq i_1, i_2 \leq k\text{-}1$,
  j_1, j_2 designating files, $0 \leq j_1, j_2 \leq B\text{-}1$,
  if $i_1 \neq i_2$ then $L_{j_1}^{(i_1)} \neq L_{j_2}^{(i_2)}$.

Embedding algorithm

This is done by performing the following steps in the sender side. The corresponding flowchart is presented in Fig. 4.

1) The secret is converted in base B
2) The secret representation in base B is split in block of n values;
3) For each secret block;
   a) Open the first cloud storage with the associated user name and password;
   b) For each value of the block.
      i. Find in the list the file having these value index;
      ii. Send to the cloud the related file ;
      iii. Open the next cloud storage with the associated user name and password.

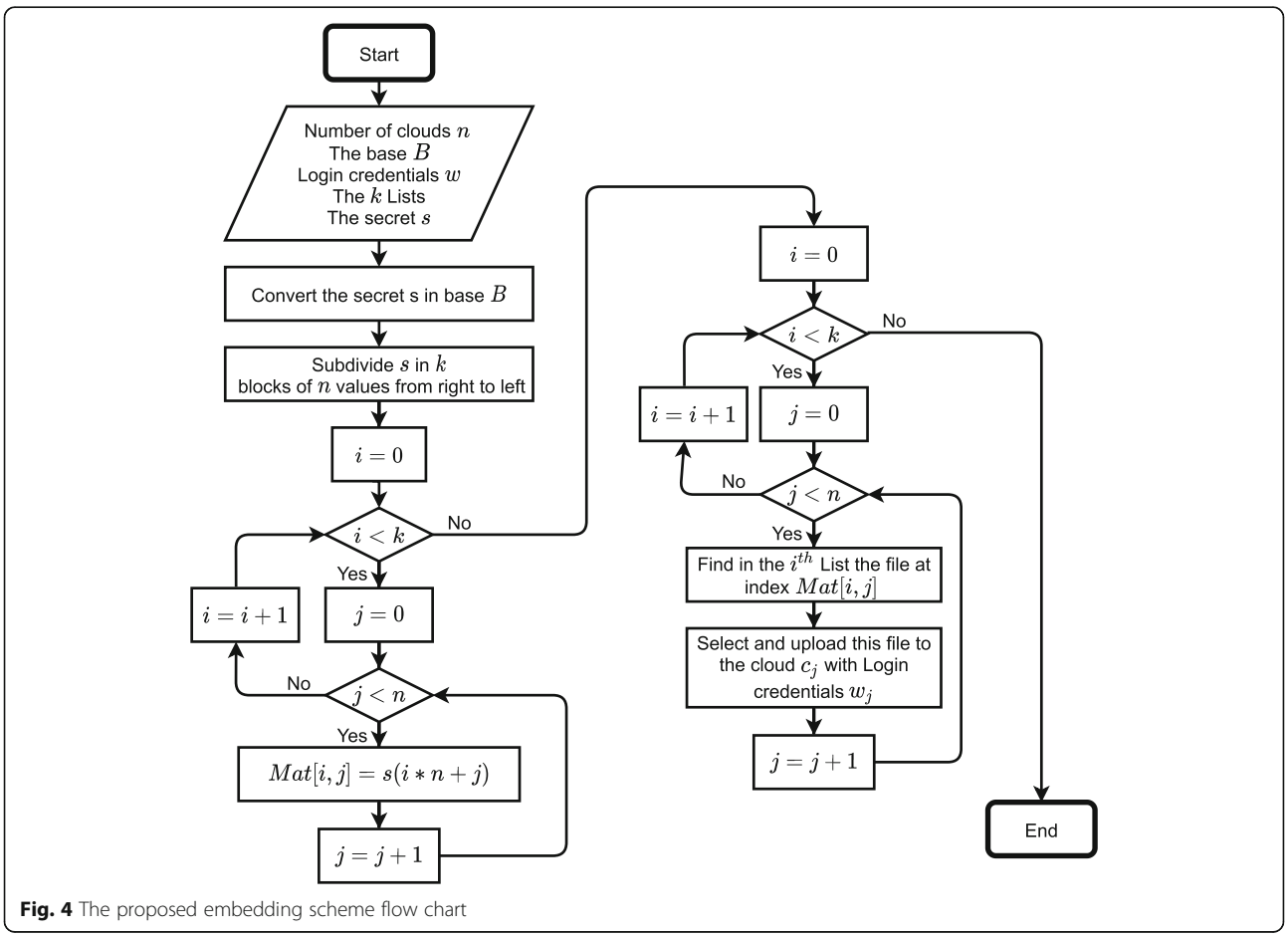


Fig. 4 The proposed embedding scheme flow chart

The embedding algorithm is as follows:

Input

C: the set of clouds;

W: the set of cloud authentication accounts;

L: the set of files lists;

B: the base;

s: the secret message;

Output

C': the cloud lists after file storage;

Begin

1. Convert the secret *s* to base *B* such that:$s = (z_{q-1}...z_1z_0)_B$, where $0 \le z_i \le B\text{-}1$;
2. Split *s* in *k* blocks of length *n* stored in the matrix *Mat* such that :
   $Mat[i,j] = s[(i \times n) + j], 0 \le i \le k\text{-}1$ and $0 \le j \le n\text{-}1$;
3. For each block of *Mat* : $0 \le i \le k\text{-}1$:
   3.1. For each value $Mat[i,j]$: $j = 0,1,\ldots,n-1$:
      3.1.1. $u = Mat[i,j]$;
      3.1.2. Find in the i^{th} list $L^{(i)}$ the files at index $Mat[i,j]$;
      3.1.3. Select and upload to the cloud c_j the file $L_u^{(i)}$.

End

Extraction algorithm

This is done by performing the following steps in the receiver side. The corresponding flowchart is presented in Fig. 5.

1. Browse each managed cloud storage;
   a. Authenticate with the user name and password;
   b. Recover the files found in the list shared between the sender and the receiver;
   c. Determine the indexes of these files;
   d. Sort the indexes in the numbering order of the lists where they come from;
   e. Store the indexes by column in a matrix;
2. Use the secret base value *B* to retrieve the secret in decimal stored in the matrix;
3. Transcode the secret from the decimal representation to binary;
4. Delete from the clouds all the files used to recover the secret.

The extraction algorithm is as follows:

Input

C: the set of clouds;

W: the set of cloud authentication accounts;

L: the set of filelists;

B: the base;

Output

s: the secret message;

Begin

1. For each cloud $c_j, j = 0,1,\ldots,n-1$:
   1.1. Extract to the cloud c_j the files contained in the lists $L^{(0)}, L^{(1)}, \ldots, L^{(k-1)}$;
   1.2. Find the indexes associated to these files named as:$z_0, z_1, \ldots, z_{k-1}$;
   1.3. For each block $i = 0,1, \ldots, k-1$:
      1.3.1.$Mat[i,j] = z_i$;
2. Compute $m = \sum_{i=0}^{k-1}\sum_{j=0}^{n-1}(Mat[i,j] \times B^{(i \times n)+j})$;
3. Convert *m* to binary and get the secret message *s*.
4. Remove from the clouds all the files used to recover the secret.

End

Time complexity analysis

In this subsection, we investigate the time complexity analysis of the proposed steganographic scheme. We assume that we have a secret *s* to distribute between *n* cloud storages using the base value *B*. We also assume that the secret *s* is hidden using *k* file lists, each containing *B* files. The proposed embedding scheme converts the secret s in base B in $O(log_B(s))$. The subdivision of s into blocks and the choice of files to be stored in the cloud are done in $O(n * k)$. Consequently, the embedding scheme time complexity is $O(n * k)$.

In the proposed extraction scheme, we assume that the cloud contains m files. The files extraction from the cloud contained in the lists is done in $O(m * k * B)$. Moreover, the secret is converted to decimal in $O(n * k)$. Finally, the secret decimal value conversion to base 2 is done in $O(log_2\ (s))$. Therefore, the time complexity of the secret extraction scheme is $O(m * k * B)$.

Evaluation

In order to assess the performance of the proposed method, a theoretical estimation of the hidden bits in multi-cloud storage environment is given. Then experiments will show in detail the steps necessary to realize the covert channel. Finally, discussion and security analysis are performed on the proposed scheme.

Hidden secret bits estimation in the clouds storage

The focus here is to hide secret bits in a set of *n* cloud. Each cloud embeds a value in base *B*, and this value can vary from *0* to *B-1*, so *B* possibilities. Then for a set of n clouds, there is B^n possibilities. So, for a secret with *k* blocks, the number of hidden bits is:

$$\boldsymbol{k \times log_2(B^n) = k \times n \times log_2(B)}.$$

Example

To describe our proposed data hiding scheme, simple numerical examples are detailed below. In this examples, s = 1, 111,101,101,000,001 a 16-bit secret and the number of managed clouds is set to four ($n = 4$). The storage providers used and their respective IDs are: SugarSync (c_0), Dropbox (c_1), OneDrive (c_2) and Google Drive (c_3). Table 2 shows associated cloud login credentials. The four lists $L^{(0)}$, $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$ used to embed the secret are presented in Table 3. Then, four scenarios are highlighted with the base taking these successive values: $B = 2$, $B = 4$, $B = 9$ and $B = 17$. Each case presents the secret distribution between these cloud storage environments and explains in detail how the secret is embedded and extracted using file lists. Note that the file lists, the base value, the set of clouds and their login credentials are the key, shared between the sender and the receiver.

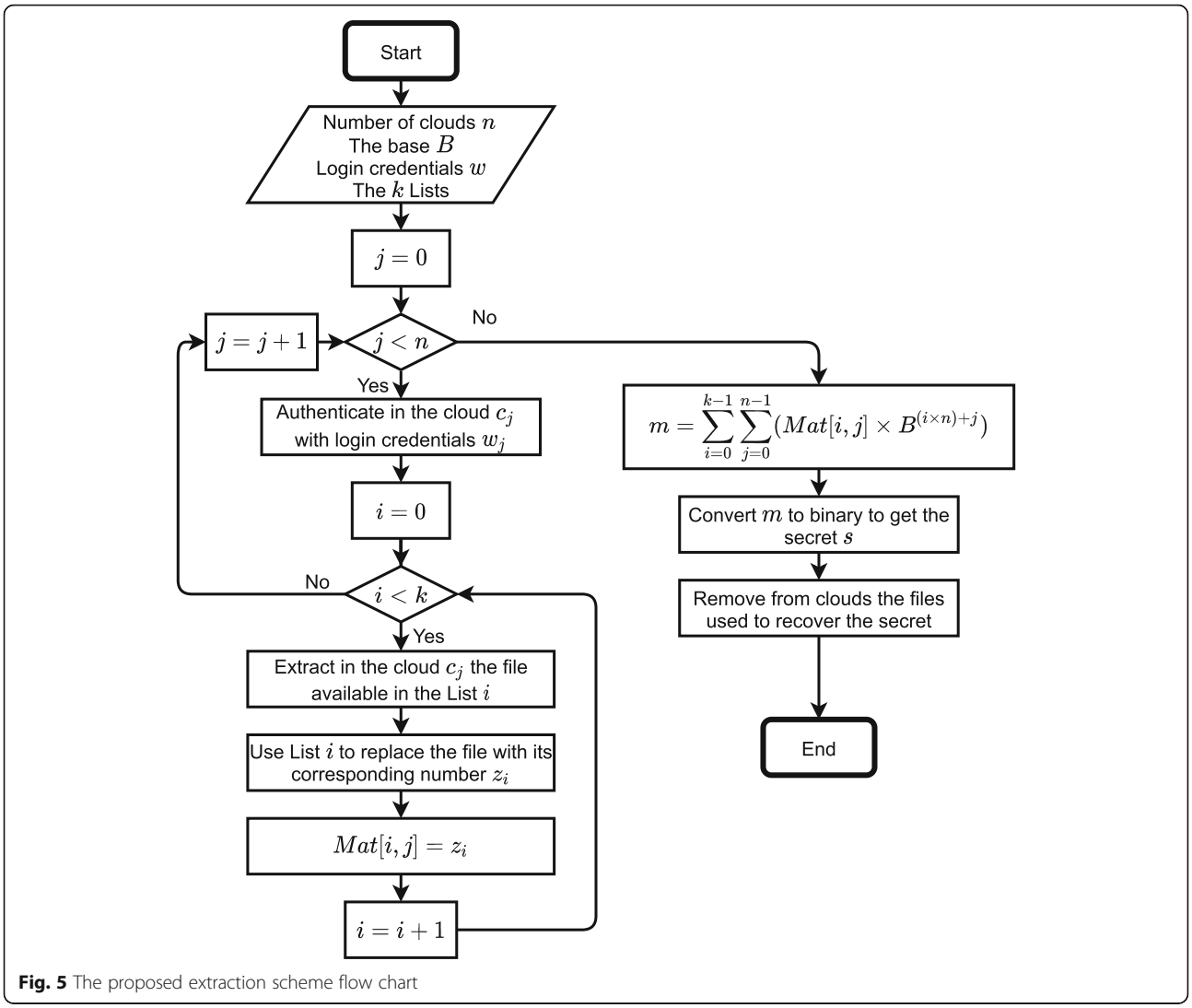


Fig. 5 The proposed extraction scheme flow chart

Case 1: s = 1,111,101,101,000,001, n = 4 and B = 2

Let's follow these steps to embed the secret key:

- Step 1: The secret is already represented in base 2, $s = (1111101101000001)_2$;
- Step 2: The secret is subdivided into groups of 4 bits, because of the four clouds available. From right to left this gives 4 blocks: 0001 0100 1011 1111;
- Step 3: For each block, each bit is linked to a distinct cloud in the order c_0, c_1, c_2 and c_3:

Bloc #0				Bloc #1				Bloc #2				Bloc #3			
0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0

- Step 4: Bits of the same cloud are grouped together. Therefore, the secret parts of each cloud are:

c_0	c_1	c_2	c_3
1	0	0	0
0	0	1	0
1	1	0	1
1	1	1	1

Table 2 The set of 4 cloud service providers handled and their login credentials

Code	Cloud Name	Login	Password
c_0	SugarSync	userlogin0@gmail.com	User-pwd0
c_1	Dropbox	userlogin1@gmail.com	User-pwd1
c_2	OneDrive	userlogin2@gmail.com	User-pwd2
c_3	Google Drive	userlogin3@gmail.com	User-pwd3

Table 3 The four file lists and their index number

	(a) $L^{(1)}$		(b) $L^{(2)}$		(c) $L^{(3)}$		(d) $L^{(4)}$
0	thesis.docx	**0**	scheduling.xlsx	**0**	conference.pptx	**0**	dataHiding.pdf
1	article.docx	**1**	statistics.xlsx	**1**	results.pptx	**1**	cryptography.pdf
2	balanceSheet.docx	**2**	budget.xlsx	**2**	slideshow.pptx	**2**	deepLearning.pdf
3	report.docx	**3**	data.xlsx	**3**	marketing.pptx	**3**	linearAlgebra.pdf
4	meeting.docx	**4**	bill.xlsx	**4**	management.pptx	**4**	dataScience.pdf
5	opportunities.docx	**5**	Evaluation.xlsx	**5**	slides.pptx	**5**	publications.pdf
6	lesson.docx	**6**	gradebook.xlsx	**6**	animation.pptx	**6**	sourceCode.pdf
7	chapter.docx	**7**	stocks.xlsx	**7**	overview.pptx	**7**	dataAnalysis.pdf
8	introduction.docx	**8**	simulation.xlsx	**8**	speech.pptx	**8**	modelingLife.pdf
9	tutorial.docx	**9**	project.xlsx	**9**	seminar.pptx	**9**	cloudComputing.pdf
10	redaction.docx	**10**	analysis.xlsx	**10**	symposium.pptx	**10**	masterDegree.pdf
11	news.docx	**11**	curves.xlsx	**11**	resume.pptx	**11**	bachelorDegree.pdf
12	book.docx	**12**	quotation.xlsx	**12**	shopping.pptx	**12**	bithdayCerti cate.pdf
13	exercise.docx	**13**	finance.xlsx	**13**	accounts.pptx	**13**	passport.pdf
14	anthem.docx	**14**	classes.xlsx	**14**	clinical.pptx	**14**	human.pdf
15	journal.docx	**15**	salaries.xlsx	**15**	aviation.pptx	**15**	contacts.pdf
16	editor.docx	**16**	phonebook.xlsx	**16**	audition.pptx	**16**	awards.pdf

- Step 5: The four lists of Table 3 are used to hide the four secret blocks. Hide respectively the 1st, 2nd, 3rd and 4th row of the matrix obtained in step 4 with the list $L^{(0)}$, $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$. More specifically, each value is replaced by the file having this index in the corresponding list. These files act as pointers to the data to be kept secret. The stego files to be uploaded in each cloudare allocated as follows:

List	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	article.docx	thesis.docx	thesis.docx	thesis.docx
$L^{(1)}$	scheduling.xlsx	scheduling.xlsx	statistics.xlsx	scheduling.xlsx
$L^{(2)}$	results.pptx	results.pptx	conference.pptx	results.pptx
$L^{(3)}$	cryptography.pdf	cryptography.pdf	cryptography.pdf	cryptography.pdf

- Step 6: The last embedding step is to transfer the files article.docx, scheduling.xlsx, results.pptx and cryptography.pdf to the cloud c_0; thesis.docx, scheduling.xlsx, results.pptx and cryptography.pdf to the cloud c_1; thesis.docx, statistics.xlsx, conference.pptx and cryptography.pdf to the cloud c_2; thesis.docx, scheduling.xlsx, results.pptx and cryptography.pdf to the cloud c_3.

Let's follow these steps to extract the secret:

- Step 1: The files of each cloud are compared to those available in the four lists $L^{(0)}$, $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$. When the names are identical, these files are retrieved and sorted in ascending order of list numbering. The files extracted by cloud and by list are as follows:

Cloud	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
c_0	article.docx	scheduling.xlsx	results.pptx	cryptography.pdf
c_1	thesis.docx	scheduling.xlsx	results.pptx	cryptography.pdf
c_2	thesis.docx	statistics.xlsx	conference.pptx	cryptography.pdf
c_3	thesis.docx	scheduling.xlsx	results.pptx	cryptography.pdf

- Step 2: The files in each list are then replaced by their number. The sequence of each cloud obtained is:

Cloud	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
c_0	1	0	1	1
c_1	0	0	1	1
c_2	0	1	0	1
c_3	0	0	1	1

- Step 3: Each binary sequence belonging to a cloud is stored in column inside a matrix called *Mat*:

$$Mat = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

- Step 4: Compute m, the secret in decimal using the base value (B = 2). The variables i and j respectively scan the rows and columns of the matrix *Mat*. The conversion is done as follows:

$$m = \sum_{i=0}^{3}\sum_{j=0}^{3} Mat[i,j] \times 2^{(i\times 4)+j}$$

$$= 1{*}2^0 + 0{*}2^1 + 0{*}2^2 + 0{*}2^3 + 0{*}2^4 + 0{*}2^5 + 1{*}2^6 + 0{*}2^7 +$$

$$1{*}2^8 + 1{*}2^9 + 0{*}2^{10} + 1{*}2^{11} + 1{*}2^{12} + 1{*}2^{13} + 1{*}2^{14} + 1{*}2^{15}$$

$$= 1 + 64 + 256 + 512 + 2\,048 + 4\,096 + 8\,192 + 16\,384 + 32\,768$$

$$= 64,321$$

- Step 5: The secret s is obtained by converting m to base 2: $(64321)_{10} = (1111101101000001)_2$
- Step 6: All the files retrieved in step 1 of extraction are removed from the cloud storage.

Case 2: s = 1,111,101,101,000,001, n = 4 and B = 4

In this case, the secret s and the number of clouds n remain unchanged. The base considered is B = 4. Let's follow these steps to embed the secret:

- Step 1: The secret is converted to base 4:

$$(1111101101000001)_2 = (64321)_{10} = (33231001)_4;$$

- Step 2:The secret is subdivided into groups of 4 values, because of the four clouds available. From right to left this gives 2 blocks: 1001 3323
- Step 3:For each block, each value is linked to a distinct cloud in the order c_0, c_1, c_2 and c_3:

Bloc #0				Bloc #1			
1	0	0	1	3	3	2	3
↓	↓	↓	↓	↓	↓	↓	↓
c_3	c_2	c_1	c_0	c_3	c_2	c_1	c_0

- Step 4: Values of the same cloud are grouped together. Therefore, the secret parts of each cloud are:

c_0	c_1	c_2	c_3
1	0	0	1
3	2	2	3

- Step 5: Two lists of Table 3 are used to hide the two secret blocks. Hide respectively the 1st and 2nd row of the matrix obtained in step 4 with the list $L^{(0)}$ and $L^{(1)}$. Each value is replaced by the file having this index in the corresponding list. The stego files to be uploaded in the clouds are allocated as follows:

List	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	article.docx	thesis.docx	thesis.docx	article.docx
$L^{(1)}$	data.xlsx	budget.xlsx	data.xlsx	data.xlsx

- Step 6: The last embedding step is to transfer the files *article.docx* and *data.xlsx* to the cloud c_0; *thesis.docx* and *budget.xlsx* to the cloud c_1; *thesis.docx* and *data.xlsx* to the cloud c_2; *article.docx* and *data.xlsx* to the cloud c_3.

Let's follow these steps to extract the secret:

- Step 1: The files of each cloud are compared to those available in the two lists $L^{(0)}$and$L^{(1)}$. When the names are identical, these files are retrieved and sorted in ascending order of list numbers. The files extracted by cloud and by list is as follows:

Cloud	$L^{(0)}$	$L^{(1)}$
c_0	article.docx	data.xlsx
c_1	thesis.docx	budget.xlsx
c_2	thesis.docx	data.xlsx
c_3	article.docx	data.xlsx

- Step 2: The files in each list are then replaced by their number. The sequence of each cloud obtained is:

Cloud	$L^{(0)}$	$L^{(1)}$
c_0	1	3
c_1	0	2
c_2	0	3
c_3	1	3

- Step 3: Each sequence belonging to a cloud is stored in column inside the matrix *Mat*:

$$Mat = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 3 & 2 & 3 & 3 \end{pmatrix}$$

- Step 4: Compute m, the secret in decimal using the base value (B = 4). The conversion is done as follows:

$$m = \sum_{i=0}^{1} \sum_{j=0}^{3} Mat[i,j] \times 4^{(i\times 4)+j}$$

$$= 1*4^0 + 0*4^1 + 0*4^2 + 0*4^3 + 3*4^4 + 2*4^5 + 3*4^6 + 3*4^7$$

$$= 1 + 64 + 768 + 2\,048 + 12\,288 + 49\,152$$

$$= 64,321$$

- Step 5: The secret s is obtained by converting m to base 2: $s = (64321)_{10} = (1111101101000001)_2$
- Step 6: All the files retrieved in step 1 of extraction are removed from the cloud storage.

Case 3: s = 1,111,101,101,000,001, n = 4 and B = 9

In this third case, the secret s and the number of clouds n remain unchanged. The base considered is B = 9. Let's follow these steps to embed the secret:

- Step 1: The secret is converted to base 9:

$$(1111101101000001)_2 = (64321)_{10} = (107207)_9;$$

- Step 2: The secret is subdivided into groups of 4 values, because of the four clouds available. From right to left this gives 2 blocks: 7207 10.
- Step 3: For each block, each value is linked to a distinct cloud in the order c_0, c_1, c_2 and c_3:

Bloc #0				Bloc #1	
7	2	0	7	1	0
↓	↓	↓	↓	↓	↓
c_3	c_2	c_1	c_0	c_1	c_0

- Step 4: Values of the same cloud are grouped together. Therefore, the secret parts of each cloud are:

c_0	c_1	c_2	c_3
7	0	2	7
0	1		

- Step 5: Two lists of Table 3 are used to hide the two secret blocks. Hide respectively the 1st and 2nd row of the matrix obtained in step 4 with the list $L^{(0)}$ and $L^{(1)}$. Each value is replaced by the file having this index in the corresponding list. The stego files to be uploaded in the clouds are allocated as follows:

List	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	chapter.docx	thesis.docx	balanceSheet.docx	chapter.docx
$L^{(1)}$	scheduling.xlsx	statistics.xlsx		

- Step 6: The last embedding step is to transfer the files *chapter.docx* and *scheduling.xlsx* to the cloud c_0; *thesis.docx* and *statistics.xlsx* to the cloud c_1; *balanceSheet.docx* to the cloud c_2 and *chapter.docx* to the cloud c_3.

Let's follow these steps to extract the secret:

- Step 1: The files of each cloud are compared to those available in the two lists $L^{(0)}$ and $L^{(1)}$. When the names are identical, these files are retrieved and sorted in ascending order of list numbers. The files extracted by cloud and by list is as follows:

Cloud	$L^{(0)}$	$L^{(1)}$
c_0	chapter.docx	scheduling.xlsx
c_1	thesis.docx	statistics.xlsx
c_2	balanceSheet.docx	
c_3	chapter.docx	

- Step 2: The files in each list are then replaced by their number. The sequence of each cloud obtained is:

Cloud	$L^{(0)}$	$L^{(1)}$
c_0	7	0
c_1	0	1
c_2	2	
c_3	7	

- Step 3: Each sequence belonging to a cloud is stored in column inside the matrix *Mat*. Empty entries in

the matrix are replaced by zeros. The resulting matrix looks like this:

$$Mat = \begin{pmatrix} 7 & 0 & 2 & 7 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- Step 4: Compute m, the secret in decimal using the base value (B = 9). The conversion is done as follows:

$$m = \sum_{i=0}^{1} \sum_{j=0}^{3} Mat[i,j] \times 9^{(i\times 4)+j}$$

$$= 7*9^0 + 0*9^1 + 2*9^2 + 7*9^3 + 0*9^4 + 1*9^5 + 0*9^6 + 0*9^7$$

$$= 7 + 162 + 5\,103 + 59\,049$$

$$= 64,321$$

- Step 5: The secret s is obtained by converting m to base 2:

$s = (64321)_{10} = (1111101101000001)_2$

- Step 6: All the files retrieved in step 1 of extraction are removed from the cloud storage.

Case 4: s = 1,111,101,101,000,001, n = 4 and B = 17

In this last case, the secret s and the number of clouds n remain unchanged. The base considered is B = 17. Let's follow these steps to embed the secret:

- Step 1: The secret is converted to base 17:

$$(1111101101000001)_2 = (64321)_{10} = (D19A)_{17};$$

- Step 2: The secret is subdivided into groups of 4 values, because of the four clouds available. This gives one block: D19A.
- Step 3: For each block, each value is linked to a distinct cloud in the order c_0, c_1, c_2 and c_3:

Bloc #0			
D	1	9	A
↓	↓	↓	↓
c_3	c_2	c_1	c_0

- Step 4: Values of the same cloud are grouped together. Letters in the block are also replaced by their equivalent: D = 13 and A = 10. Therefore, the secret parts of each cloud are:

c_0	c_1	c_2	c_3
10	9	1	13

- Step 5: As the subdivision gave one block, just one list is used. Hide the vector value obtained in step 4 with the list $L^{(0)}$. The stego files to be uploaded in the clouds are allocated as follows:

List	Cloud c_0	Cloud c_1	Cloud c_2	Cloud c_3
$L^{(0)}$	redaction.docx	tutorial.docx	article.docx	exercise.docx

- Step 6: The last embedding step is to transfer the files *redaction.docx*to the cloud c_0; *tutorial.docx* to the cloud c_1; *article.docx* to the cloud c_2 and *exercise.docx* to the cloud c_3.

Let's follow these steps to extract the secret:

- Step 1: The files of each cloud are compared to those available in the list $L^{(0)}$. When the names are identical, these files are retrieved and sorted in the order in which the lists were created. The files extracted by cloud and by list is as follows:

Cloud	$L^{(0)}$
c_0	redaction.docx
c_1	tutorial.docx
c_2	article.docx
c_3	exercise.docx

- Step 2: The files in each list are then replaced by their number. The sequence of each cloud obtained is:

Cloud	$L^{(0)}$
c_0	10
c_1	9
c_2	1
c_3	13

- Step 3: The sequence stored in column in the matrix *Mat*. The resulting vector looks like this:

$$Mat = (10\ \ 9\ \ 1\ \ 13)$$

- Step 4: Compute m, the secret in decimal using the base value (B = 17). The conversion is done as follows:

$$m = \sum_{i=0}^{0}\sum_{j=0}^{3} Mat[i,j] \times 17^{(i\times 4)+j}$$

$$= 10*17^0 + 9*17^1 + 1*17^2 + 13*17^3$$

$$= 10 + 153 + 289 + 63\,869$$

$$= 64,321$$

- Step 5: The secret s is obtained by converting m to base 2:

$$s = (64321)_{10} = (1111101101000001)_2$$

- Step 6: All the files retrieved in step 1 of extraction are removed from the cloud storage.

Discussion

In this paper, we propose a steganographic scheme for secret distribution resistant to detection. Compared to related work, secret extraction assumes that files do not undergo any modification when distributing the secret in multi-cloud storage environment, by hiding the existence of the covert channel between the communicating parties. As examples 1 to 4 show, the data distributed in the clouds decrease as the value of the chosen base increases. We have respectively distributed 16, 8, 6 and 4 values with base $B = 2$, 4, 9 and 17 in examples 1 to 4, considering a fixed secret size and clouds number. We also observe that the base value choice has an impact on the number and the size of the file lists necessary for the secret dissimulation. Each list must contain at least B files and the number of lists must be identical to the blocks number obtained after splitting the secret message represented in base B. In example 1, four lists are used, then in examples 2 and 3 only two lists are used and finally a single list in the last example. Moreover, the files available in the lists are not fully used. The base value indicates the number of files to manage both for secret insertion and extraction. Indeed, only 2, 4, 9 and 17 files are respectively taken into account in each of the lists of examples 1 to 4. An additional argument that argues in favour of the proposed scheme security is the ability to use any file extension to establish the covert channel such as images (.png, .jpg, ...), binary files (.exe, .bin, ...) text files (.txt, .doc, ...), and so on. In this case, the *word, excel, powerpoint* and *pdf* files were used while maintaining their integrity. Another important note should be raised when using the multi-cloud storage environment. These can be multiple accounts available from a single or multiple storage providers. In the examples of this work we used accounts from four different providers name as: SugarSync, Dropbox, OneDrive and Google Drive.

Ultimately, a comparison of these examples reveals that example 4 has a better distribution of the secret in the four chosen clouds. Only one file is deposited in each cloud, which is not the case in the other examples. This also facilitates the secret extraction by reducing the search time for files available in the lists. In general, the embedding program takes as input the base value, the number of clouds and the directories containing the lists of files, then outputs the files to be uploaded to each cloud in directories. While the extraction program takes as input a set of files from the clouds, the base value, the clouds number and the lists, then outputs the secret. For a better distribution of the secret, an optimal choice must be made on the following two parameters: the clouds number and the base value.

Security analysis

In the literature, some techniques [4, 17–19] use a special character insertion to hide information like: space, ASCII code character *A0*, characters coloring, text justification. In our method, no special character is used. Thus, by just observing the file content, there will be no suspicious items. Therefore, the file content doesn't attract the adversary attention.

Security analysis is done by considering two attacks hypothesis:

Hypothesis 1:an attack by an adversary who doesn't have the ability to access the cloud accounts. This adversary has the following limits:

- He doesn't know the key (the cloud user name and password, the files lists, the base);
- He doesn't know the clouds storage content;
- He doesn't know that a secret communication is taking place by observing only the files transfer between the clouds. Nothing can reveal the secret communication existence because there is no addition of special information in the exchange files.

Hence, it does not attract the adversary's attention. In short, he can't do anything.

Hypothesis 2: an attack by an adversary who can partially or fully access the accounts of the different clouds. This adversary has the following limits:

• He doesn't know the key (the files lists and the base). If the adversary gets the file lists by accessing the cloud accounts, he must find the correct order of the secret distribution in the clouds as well as the numbering of the lists and files contained in these lists. Therefore, he must perform $B! * k! * n!$ permutations in a case of exhaustive search for a successful attack. Unfortunately, this number of permutations is exponential.
• He can't make a link between Alice and Bob. The only connection between Alice and Bod is during the key exchange. After that, there is no direct communication between them. In the proposed steganographic scheme, there is only communication between each party and the cloud. As presented in the Fig. 6, the secret channel doesn't make a direct connection between Alice and Bob. Thus, the usual security model as mentioned in the Fig. 1 is broken.

Just like Hypothesis 1, he can't do anything.

Conclusion

In this paper a new steganography paradigm, transparent to any attacker and resistant to the detection and the secret extraction was proposed. Two properties contribute to achieve these goals: the files do not undergo any modification while the distribution of the secret in the multi-cloud storage environment allows us to hide the existence of the covert channel between the communicating parties. Related works hide usually information inside the covert media. In this work, the covert media is a pointer to information. Therefore the file carries the information without being modified and the only way to access it is to have the key. The experiments carried out have shown that the secret distribution in the clouds decreases as the value of the chosen base increases. Moreover, the base value choice has an impact on the number and the size of the file lists necessary for the secret dissimulation. An additional argument that argues in favour of the proposed scheme security is the ability to use any file extension to establish the covert channel while maintaining their integrity. Another important note should be raised when using the multi-cloud storage environment. These can be multiple accounts available from a single or multiple storage providers.

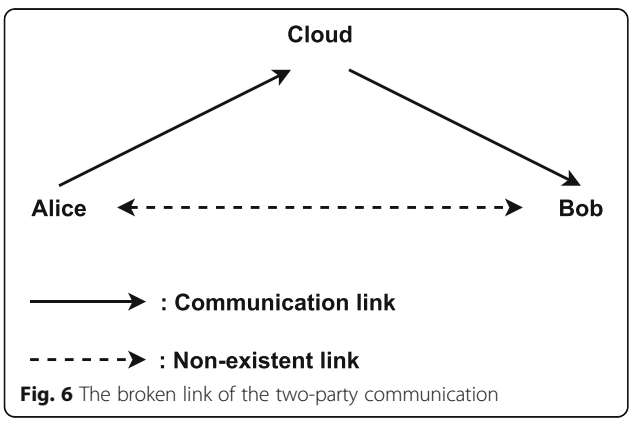


Fig. 6 The broken link of the two-party communication

The paper shows interesting comparison results with remarkable security contributions. The work can be seen as a new open direction for further distributed stego research. Future work will consist in improving the scheme by proposing optimal parameters allowing a better distribution of the secret. We will also study the robustness of the proposed technique in the face of very large secret data. We are also motivated to design new steganographic schemes resistant to detection by preserving the shared files integrity.

Acknowledgements
This work was supported by UMMISCO and the University of Yaounde 1.We thank the European Commission through the Erasmus+ and DFG projects for funding our research visit to the Brandenburg University of Technology and the IHP - Leibniz Institutfür innovative Mikroelektronik, in Germany.

Authors' contributions
RenéNdoundam Conceived, designed and directed this research. Leonel MoyouMetcheka investigated, implemented and wrote the paper. All authors reviewed and approved the final manuscript.

Authors' information
Leonel MoyouMetcheka Ph.D. candidate in Computer Science at University of Yaounde I. He obtained his Master degree of Computer Science at University of Yaounde I in 2014. His research interests include steganography.
René Ndoundam is Associate Professor in Computer Science. He received hisDoctoratd'Etat in Computer Science from the University of Yaounde I, in 2005. His research interests include automata, complexity, recommendation systems and steganography.

Funding
This work has no funding.

Availability of data and materials
No data or models were generated during the study. However, a code wrote in C language was used to distribute the secret in the clouds handled.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Team GRIMCAPE, Yaounde, Cameroon. [2]Sorbonne University, IRD, UMMISCO, F-93143 Bondy, France. [3]Department of Computer Science, University of Yaounde I, P.o. Box 812, Yaounde, Cameroon.

Received: 2 August 2020 Accepted: 20 October 2020
Published online: 09 December 2020

References
1. Mazurczyk W, Wendzel S, Zander S, Houmansadr A, Szczypiorski K (2016) Information hiding in communication networks: fundamental, Mechanisms, Applications, and Countermeasures. Wiley
2. Chang C-C, Kieu TD (2010) A reversible data hiding scheme using complementary embedding strategy. Inf Sci 180(16):3045–3058
3. Por LY, Delina B (2008) Information hiding: a new approach in text steganography. In: WSEAS international conference. In: Proceedings. Mathematics and computers in science and engineering, vol 7. World Scientific and Engineering Academy and Society
4. Ekodeck SGR, Ndoundam R (2016) Pdf steganography based on chinese remainder theorem. J Inform Security Appl 29:1–15

5. Sahu AK, Swain G (2020) Reversible image steganography using dual-layer lsb matching. Sensing Imaging 21(1):1
6. Jiang S, Ye D, Huang J, Shang Y, Zheng Z (2020) Smartsteganogaphy: Light-weight generative audio steganography model for smart embedding application. J Netw Comput Appl 102689
7. Pilania U, Gupta P (2020) Analysis and implementation of IWT-SVD scheme for video steganography. In: Micro-Electronics and Telecommunication Engineering. Springer, Singapour, pp 153–162
8. Singh N, Bhardwaj J, Raghav G (2017) Network steganography and its techniques: A survey. Int J Comput Appl 174:2
9. Moskowitz IS, Chang L, Newman RE (2002) Capacity is the wrong paradigm. In: Proceedings of the 2002 Workshop on New Security Paradigms, pp 114–126
10. Sajedi H, Jamzad M (2010) Bss: Boosted steganography scheme with cover image preprocessing. Expert Syst Appl 37(12):7703–7710
11. Kopiczko P, Mazurczyk W, Szczypiorski K (2013) Stegtorrent: a steganographic method for the p2p file sharing service. In: 2013 IEEE security and privacy workshops, p 151 157 IEEE
12. El-Latif AAA, Abd-El-Atty B, Elseuofi S, Khalifa HS, Alghamdi AS, Polat K, Amin M (2020) Secret images transfer in cloud system based on investigating quantum walks in steganography approaches. Physica A: Stat Mech Appl 541:123687
13. Ali M, Khan SU, Vasilakos AV (2015) Security in cloud computing: Opportunities and challenges. Inf Sci 305:357–383
14. Yang J, Liao X (2020) An embedding strategy on fusing multiple image features for data hiding in multiple images. J Vis Commun Image Represent 71:102822
15. Luo X-Y, Wang D-S, Wang P, Liu F-L (2008) A review on blind detection for image steganography. Signal Process 88(9):2138–2157
16. Wu S, Zhong S, Liu Y (2018) Deep residual learning for image steganalysis. Multimed Tools Appl 77(9):10437–10453
17. Khosravi B, Khosravi B, Khosravi B, Nazarkardeh K (2019) A new method for pdf steganography in justified texts. J Inf Secur Appl 45:61–70
18. Malik A, Sikka G, Verma HK (2017) A high capacity text steganography scheme based on lzw compression and color coding. Engineering Science and Technology. Int J 20(1):72–79
19. Lee I-S, Tsai W-H (2010) A new approach to covert communication via pdf files. Signal Process 90(2):557–565
20. Bhattacharyya S (2011) A survey of steganography and steganalysis technique in image, text, audio and video as cover carrier. J Global Res Comput Sci 2(4)
21. Wang Z, Chen M, Yang Y, Lei M, Dong Z (2020) Joint multi-domain feature learning for image steganalysis based on cnn. EURASIP J Image Video Process 2020(1):1–12
22. Qiao T, Retraint F, Cogranne R, Zitzmann C (2015) Steganalysis of jsteg algorithm using hypothesis testing theory. EURASIP J Inf Secur 2015(1):1–16
23. Simmons GJ (1984) The prisoners' problem and the subliminal channel. In: advances in cryptology. Springer, Boston, pp 51–67
24. Menezes AJ, Katz J, Van Oorschot PC, Vanstone SA (1996) Handbook of Applied Cryptography. CRC press
25. Jackson JT, Gunsch GH, Claypoole RL, Lamont GB (2003) Blind steganography detection using a computational immune system: a work in progress. Int J Digit Evid 4(1):19
26. Farid H, Lyu S (2003) Higher-order wavelet statistics and their application to digital forensics. In: 2003 Conference on computer vision and pattern recognition workshop, vol 8. IEEE, pp 94–94
27. Farid H (2001) Detecting steganographic messages in digital images. Report TR2001–412. Dartmouth College, Hanover
28. Fridrich J, Goljan M (2002) Practical steganalysis of digital images: state of the art. In: security and Watermarking of Multimedia Contents IV, vol 4675. International Society for Optics and Photonics, pp 1–13
29. Ozer H, Avcibas I, Sankur B, Memon ND (2003) Steganalysis of audio based on audio quality metrics. In: Security and Watermarking of Multimedia Contents V, vol 5020, pp 55–66 International Society for Optics and Photonics
30. Goudar R, Patil A (2012) Packet length based steganography detection in transport layer. Int J Sci Res Publ 2:12
31. Elsadig MA, Fadlalla YA (2018) Packet length covert channels crashed. J Comput Sci Comput Math 8(4):55–62
32. Koikara R, Deka DJ, Gogoi M, Das R (2015) A novel distributed image steganography method based on block-dct. In: Advanced Computer and Communication Engineering Technology. Springer, Cham, pp 423–435
33. Wibisurya A et al (2017) Distributed steganography using five pixel pair differencing and modulus function. Procedia Comput Sci 116:334–341
34. Liao X, Yin J, Chen M, Qin Z (2020) Adaptive payload distribution in multiple images steganography based on image texture features. IEEE Trans Dependable Secure Comput
35. Liao X, Wen Q-Y, Shi S (2011) Distributed steganography. In: 2011 Seventh international conference on intelligent information hiding and multimedia signal processing. IEEE, pp 153–156
36. Ito M, Saito A, Nishizeki T (1989) Secret sharing scheme realizing general access structure. Electron Commun Jpn (Part III: Fundamental Electronic Science) 72(9):56–64
37. Iftene S (2006) Secret sharing schemes with applications in security protocols. Sci Ann Cuza Univ 16:63–96
38. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613
39. Blakley GR (1979) Safeguarding cryptographic keys. In: 1979 international workshop on managing requirements knowledge (MARK). IEEE, pp 313–318
40. Gutub A, Al-Juaid N, Khan E (2019) Counting-based secret sharing technique for multimedia applications. Multimed Tools Appl 78(5):5591–5619
41. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M. (1995) Proactive secret sharing or: how to cope with perpetual leakage. In: annual international cryptology conference, pp. 339-352. Springer
42. Al-Ghamdi M, Al-Ghamdi M, Gutub A (2019) Security enhancement of shares generation process for multimedia counting-based secret-sharing technique. Multimed Tools Appl 78(12):16283–16310
43. Gutub A, AlKhodaidi T (2020) Smart expansion of target key for more handlers to access multimedia counting-based secret sharing. Multimed Tools Appl 1:29
44. AlKhodaidi T, Gutub A (2020) Trustworthy target key alteration helping counting-based secret sharing applicability. Arab J Sci Eng 1:21
45. Gutub A, Alaseri K (2019) Hiding shares of counting-based secret sharing via arabic text steganography for personal usage. Arab J Sci Eng 1:26
46. Gutub AA-A, Alaseri KA (2019) Refining arabic text stego-techniques for shares memorization of counting-based secret sharing. J King Saud Univ Comput Inf Sci
47. Gutub A, Al-Ghamdi M (2019) Image based steganography to facilitate improving counting-based secret sharing. 3D Res (1):6
48. Gutub A, Al-Ghamdi M (2020) Hiding shares by multimedia image steganography for optimized counting-based secret sharing. In: Multimedia Tools and Applications, pp 1–35
49. Li B, He J, Huang J, Shi YQ (2011) A survey on image steganography and steganalysis. J Inf Hiding Multimedia Signal Process 2(2):142–172
50. Karampidis K, Kavallieratou E, Papadourakis G (2018) A review of image steganalysis techniques for digital forensics. J Inf Secur Appl 40:217–235
51. Liu T-Y, Tsai W-H (2007) A new steganographic method for data hiding in microsoft word documents by a change tracking technique. IEEE Trans Inf Forensic Secur 2(1):24–30
52. Su W, Ni J, Hu X, Fridrich J (2020) Image steganography with symmetric embedding using gaussianmarkov random field model. IEEE Trans Circuits Syst Video Technol
53. Ali AH, George LE, Mokhtar MR (2020) An adaptive high capacity model for secure audio communication based on fractal coding and uniform coefficient modulation. Circuits Syst Signal Process 39(10):5198–5225
54. Baziyad M, Rabie T, Kamel I (2020) Directional pixogram: a new approach for video steganography. In: 2020 advances in science and engineering technology international conferences (ASET), Dubai, United Arab Emirates, pp 1–5

Publisher's Note

Two secure online steganography schemes based on HTTP request sequences

Leonel Moyou Metcheka[1,2,3], **Stephane Gael Raymond Ekodeck**[1,2,3], **René Ndoundam**[*1,2,3]

[1]Team GRIMCAPE
[2]Unversity, IRD, UMMISCO, F-93143, Bondy, France
[3]Department of Computer Science, University of Yaounde I, Yaounde, 812, Cameroon

*E-mail : ndoundam@yahoo.com

Abstract

The interest of steganography is to find ways to leak confidential information without being detected. Several methods proposed in the literature implement strategies for secret embedding in one or more covert media with increased security performance, by enhancing alteration resistance. Nevertheless, these embedding strategies are vulnerable to detection with blind universal steganalysis. This paper sheds light on two new secure covert channels, perfectly driven by online web operations that do not leave fingerprints. The design features of this undetectable scheme are based on lightweight requests for secret encoding coupled to the cover media independence. Experiments on this new approach have shown remarkable security results compared to existing ones. The results are significant and present promising research contributions.

Keywords

Undetectability ; Lightweight covert channel ; Tagged URLs ; HTTP request sequences

I INTRODUCTION

Steganography is the art and science of hiding information so that its presence cannot be detected. Its aim is to set up a covert channel for secret communications [14] so that no one except the recipient is aware of the existence of the secret message [12]. Classic steganography techniques conceal secret messages in digital content such as text, image, video and audio files [18, 31, 33, 35]. Hence, three properties are used to evaluate the performance of a steganographic scheme [29]. Firstly the capacity, which refers to the amount of secret bits that could be hidden inside a stego-covert. Secondly the robustness, which assesses the ability of the scheme to resist modification or destruction of the secret message. And finally the security, which assesses the inability of an intruder to detect the secret information exchanged. Ultimately, security is the most critical property desired to achieve undetectable communications [6].

Recently, distributed data hiding in multi-cloud storage environment [32] has been introduced based on the files integrity. This secret channel allows to bypass steg analysis since nothing is inserted or modified in the multimedia files handled. However, the content of the key and the exposed method are complex. Indeed, four parameters must be defined in the key while the process of inserting and extracting the secret requires managing multiple clouds with many multimedia files. In this paper, we propose two lightweight covert channels independent to

multimedia files that performs secret communications using HTTP request sequences. The two main parameters involved in the key are the tagged URLs and the IP addresses used in the communication. The method is based on the simple operations of opening web pages from the selected URLs. The secret is distributed via URLs opened by the sender in a methodical way, depending on the embedding method used.

The rest of the paper is organized as follows : section 2 describes existing steganographic schemes and their limits. The lightweight cover secure channels contribution are presented in section 3. Experimental results are done in section 4 and finally section 5 is devoted to the conclusion.

II COVERT CHANNELS DETECTION

This section discusses the steganographic schemes security that insert confidential information inside the covert media. Existing work is highlighted to show that an attacker is able to perform progressive scans of these covert media in order to detect or destroy the presence of the secret. The cover channels detection based on text, images, video and audio are presented.

Text steganography is classified into three categories [20]: format-based methods, linguistic methods as well as random and statistical generation methods. Format-based methods modify the format or the layout features of the cover text without altering the sentences or the words. Features consist of word spacing, line spacing, font style, text color [16, 18, 30]. Linguistic or natural language processing-based methods modify the syntax and semantics of text content using features such as synonyms, abbreviations, word similarity to hide secret bits [27, 37]. Random and statistical generation methods are used to generate automatically covert text with regard to the statistical properties of the secret message. To produce covert text in the natural language, data compression techniques or random covert can be use [17, 21]. Assuming an attacker who has access to the stego text, he can intentionally insert, modify, delete, reorder words or characters or even change the formatting of the text (color, font) and the character encoding (ASCII, UNICODE, UTF8). This inevitably leads to the destruction of the secret message. Furthermore, statistical analysis can be carried out to detect the covert channel [10, 28].

Image steganography is classified into three types: spatial domain techniques, transform domain techniques and adaptive techniques [36]. The spatial domain based image steganography directly modify the image pixels to integrate the secret bits. Usually, the techniques employed are: Least Significant Bit (LSB), palette based techniques, Pixel Value Differencing (PVD), multiple bit-plane based techniques and histogram shifting [22]. Instead of directly manipulating the pixels, secret insertion is done by modifying the coefficients in the transform domain. Here, the techniques used include: Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) [23]. Adaptive methods are designed to target specific regions or pixels of the cover image to insert secret data to improve the efficiency of steganographic techniques. Human Visual System (HVS) is used for embedding as well as a threshold value [25]. Likewise, the detection of secret messages in images requires some advanced statistical testing such as higher-order statistics, Markov random fields, linear analysis and wavelet statistics [34]. Other recent techniques use blind universal detection methods to identify any new or unknown embedding schemes used in an image to hide a secret [11, 26].

Video steganography employs techniques applied to images for the embedding of secrecy, particularly in the spatial and transform domain [36]. However, the LSB technique of the spatial

domain is the most used to insert the secret bits in the videos [24]. Methods employing machine learning are implemented to detect secret messages inserted in video media [19].

Audio steganography is subdivided into four main categories: low-bit encoding, phase encoding, spread spectrum coding and echo hiding [15]. Regarding the low-bit coding technique, the secret is inserted into the audio sample using the LSB technique [13]. Phase encoding uses the phase components of sound so that it is not perceptible by human listening [1]. The secret bits are integrated as phase shifts in the phase spectrum of the audio signal. Spread spectrum coding consists in randomly spreading the secret data bits across the frequency spectrum of the audio signal [7]. Finally, the echo hiding technique hides the secret by introducing an echo in the discrete audio signal [5]. Detection methods can be applied for a specific type of audio steganography [9]. Also, universal detection methods are able to signal the presence of secret messages in audio media [8].

The common point of steganography techniques applied to multimedia files is the modification properties of the cover media for secret embedding, therefore leaving traces which can be detected by steg analyzes. Likewise, if an algorithm can determine the presence of a secret message in a covert media, then the whole steganographic system used is considered broken[4]. Considering these limits, we want to propose a secure steganographic scheme independent of multimedia files because it can be detected by current steganlysis methods, while preserving the integrity of the medium used.

III CONTRIBUTION

The basic idea is to use everyday online operations to have a stealthy signature that is undetectable. These operations concern the consultation of websites which is in a way unsuspected and goes practically unnoticed in case of concealment of secret information. The interest of this new lightweight covert channels is to achieve perfectly undetectable communications without the usual exchange of stego multimedia files. The sender consults the web pages and the receiver controls the website containing these web pages. The URLs of web pages are tagged in order to conceal information when browsing. Therefore, the contribution is divided into the following three main points :

- The lightweight covert channel: the scheme needs simple operation;
- Undetectable by steg analysis : the technique is based on usual internet browsing;
- Covert media independent : the technique used does not require the exchange or modifications of multimedia files.

Two schemes based on a stealth signature are proposed using online web browsing operations. The first scheme uses HTTP requests based on dependency inside sequences with permutations to transfer the secret. While the second scheme uses HTTP requests based on dependency between sequences and applications to transfer the secret.

3.1 Notations and hypothesis

These notations are useful both for the secret insertion and extraction:

- S : a secret message in binary ;
- B : the base value ;
- π : a permutation of order n such that $\pi = (\pi_0, \pi_1, ..., \pi_{n-1})$;
- r : the rank of a permutation of n elements;
- U : a list of n URL such that $U = (U_0, U_1, ..., U_{n-1})$;
- P : a list of m IP addresses such that $P = (p_0, p_1, ..., p_{m-1})$;

- $U_0, U_1..., U_{n-1}$: a sequence of URL such that :
  - $\forall U_i \in U, 0 \leq i \leq n-1$;
  - $\forall i_1, i_2, 0 \leq i_1, i_2 \leq n-1$;
  - if $i_1 \neq i_2$ then $U_{i_1} \neq U_{i_2}$.
- U^i : the i^{th} block of n URL selected : $U_0^i, U_1^i, ..., U_{n-1}^i$;

3.2 HTTP requests based on dependency inside sequences

3.2.1 Overview

In this first scheme, the HTTP request sequences are carried out by the sender in a methodical way in order to transfer the secret data. Indeed, the requests within a sequence are ordered and therefore dependent on each other. The communication model proposed in Figure 1, shows the sender who transfers the secret data by making sequences of HTTP requests to the receiver, which has a set of web pages accessible online. The access URLs to some of these web pages are labeled and called as tagged URLs. These URLs are used by the sender to encode the secret while the receiver uses the source IP address of the sender as well as the tagged URLs to decode the secret. Note that the numbers present in these requests represent the identifiers of the accessible URLs. Throughout Figure 1, several types of requests are applied to URLs. In the set of URLs available in the web server, a short list of n URLs is identified to form the tagged URLs. HTTP requests can be made on tagged URLs as well as on untagged URLs. A distinction is made between HTTP requests containing tagged URLs, coming from the IP address of the sender and normal users. To encode and decode the hidden data bits, the ranking and unranking functions of Myrvold et al. are used [3]. These two functions are described in the next section.

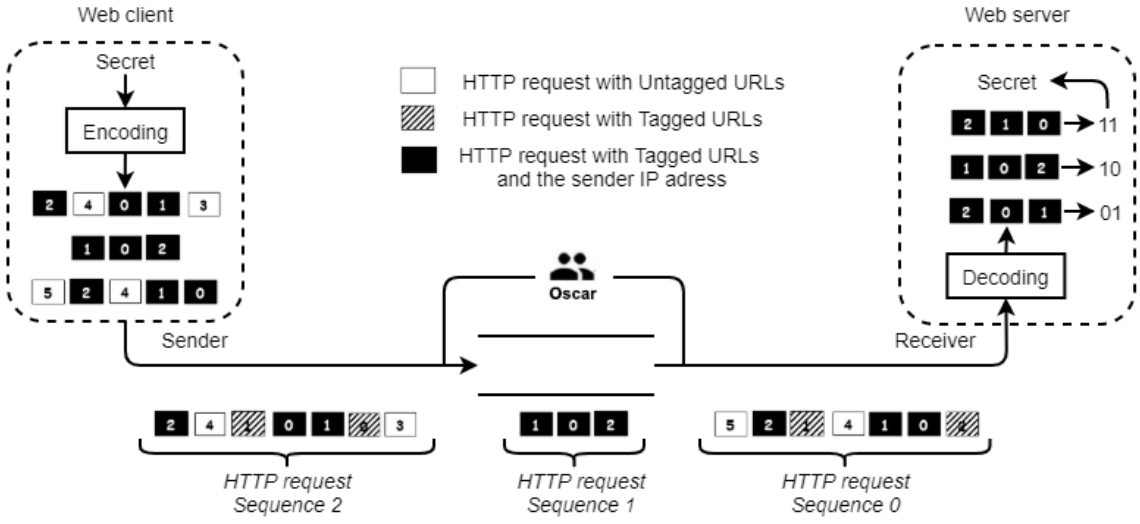


Figure 1: Overview of the proposed scheme.

3.2.2 Permutation generation functions

A permutation of order n is an arrangement of n elements. Thus, the number of permutations of n elements is $n(n-1)...2.1 = n!$. Consequently, at each permutation sequence $\pi_0, \pi_1..., \pi_{n-1}$ is associated a rank between 0 and $n! - 1$. To do such indexing, we use a ranking function and the reverse operation requires the unranking function. Indeed, for each of the $n!$ permutations of n symbols, the ranking function returns an integer in the interval $[0, n! - 1]$. Inversely, the unranking function takes as input an integer between 0 and $n! - 1$ and returns the permutation of n

symbols having this rank. Some related works [2] produce these permutations in lexicographic order. The works of Myrvold et al. [3] which are taken into account in this paper, generate these permutations but not in lexicographic order. Their respective pseudo-codes are described below.

Unranking function : to determine the permutation of n symbols of rank r, we must first initialize the array π to be the permutation identity $\pi[i] = i$, for $i = 0, 1, \cdots n-1$. Then call the unrank procedure below [3], with the following parameters: n, r and the initial permutation π. Once the procedure is completed, the expected permutation is available in the array π.

Procedure $unrank(n, r, \pi)$
begin
if $n > 0$ **then**
$swap(\pi[n-1], \pi[r \bmod n])$;
$unrank(n-1, \lfloor r/n \rfloor, \pi)$;
end;
end;

Ranking function : to determine the rank r associated with a permutation π of n symbols, we first initialize the array π^{-1} like this: $\pi^{-1}[\pi[i]] = i$, for $i = 0, 1, \cdots, n-1$. Then the rank function is called with the following parameters: n, π and π^{-1}. This returns the rank associated with the permutation π.

function $rank(n, \pi, \pi^{-1})$:integer
begin
if $n = 1$ **then** return(0) **end**;
$s := \pi[n-1]$;
$swap(\pi[n-1], \pi[\pi^{-1}[n-1]])$;
$swap(\pi^{-1}[s], \pi^{-1}[n-1])$;
return$(s + n.rank(n-1, \pi, \pi^{-1}))$;
end;

3.2.3 Secret encoding and decoding illustration

The number of permutations of n elements is $n!$ and each permutation has a rank between 0 and $n! - 1$. Taking $n = 3$ for this illustration, the number of permutations is $3! = 6$. We enumerate all the arrangements of the elements $0, 1, 2$. Consequently, each permutation sequence of 3 elements will be associated with a rank between 0 and 5. Table 1 illustrates the different permutations and their respective ranks. Thus, the coding of binary secret 10 requires the use of the sequence: 1 0 2, because it corresponds to rank 2. Similarly, the decoding of the sequence 2 1 0, reveals the secret value 3 which is equal to 11 in binary.

Table 1: Ranks and permutation sequences for n = 3.

0	1 2 0	**2**	1 0 2	**4**	0 2 1	**1**	2 0 1	**3**	2 1 0	**5**	0 1 2

3.2.4 The stego key

Two parameters must be shared between the sender and the receiver :

- A list of tagged URLs: $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- A list of sender IP addresses : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$.

3.2.5 *Embedding scheme*

In this first scheme, the secret represented in binary is divided into blocks. Each binary block is encoded by selecting a sequence of URLs. This opening sequence of URLs is determined by applying the unrank function to the decimal value of each block. The sender makes HTTP requests to the web server in this predefined order. Therefore, the secret embedding algorithm is as follows :

Input

S : the secret message;

p_0 : the sender IP address ;

π : the permutation array ;

U : the tagged URLs list;

Output

U' : the URL selected after browsing;

Begin

1. The secret S is represented in binary as follows : $S = (z_{q-1}...z_1z_0)_2$, where $0 \leq z_i \leq 1$;
2. The number of tagged URLs is determined as follows : $n = |U|$;
3. The size of a secret block is evaluated as follows : $l = \lfloor log_2(n!) \rfloor$;
4. The number of blocks is evaluated as follows: $k = \lceil q/l \rceil$;
5. The secret is split into k blocks of l bits : $S[(i \times l) + j]$, $0 \leq i \leq k-1$ and $0 \leq j \leq l-1$;
6. For each secret block $i = 0, 1, ..., k-1$:
   - 6.1. Compute the decimal value of the block : $r = \sum_{j=0}^{l-1}(S[(i \times l) + j] \times 2^j)$;
   - 6.2. Initialize the array π to be the permutation identity $\pi[j] = j$, $j = 0, 1, \cdots n-1$.
   - 6.3. Determine the permutation sequence $\pi_0, \pi_1..., \pi_{n-1}$ by applying the unrank function with the following parameters : n, r and π;
   - 6.4. Determine the URLs sequence : $U_{\pi_0}, U_{\pi_1}, ..., U_{\pi_{n-1}}$
   - 6.5. For each URL U_{π_j}, $j = 0, 1, ..., n-1$:
     - 6.5.1. Send HTTP requests on the URL U_{π_j} with the sender IP address p_0 ;
     - 6.5.2. Get the HTTP response of the request.

End

3.2.6 *Extraction scheme*

On the server side, the receiver records the links requested by the client as well as the associated IP addresses. Then, it sorts through the incoming requests those coming from a known source IP address. The related URL links are extracted in order of opening. The rank function is applied to the different sequences of links in order to reconstitute the corresponding secret block. The dynamic programming of websites in python language uses these two instructions to determine respectively the URL of HTTP requests and the remote IP address of the clients : $request.META.get('SCRIPT\_URI')$ and $request.META.get('REMOTE\_ADDR')$. In addition the *time()* function is used to determine the time at which a URL is visited. Therefore, the secret extraction algorithm is as follows :

Input

p_0 : the sender IP address ;

U : the tagged URLs list;

π : the permutation array ;

π^{-1} : the inverse permutation array ;

Output

S : the secret message;

Begin

1. Determine the number of tagged URLs : $n = |U|$;
2. Determine the amount of bits hidden in a list of n URL : $l = \lfloor log_2(n!) \rfloor$;
3. Capture URLs in order of opening HTTP requests;
4. Extract the tagged one belonging to the list U;
5. Filter only URLs from the single known IP address p_0;
6. The URLs obtained is subdivided into k blocks of n URLs : $U^0, U^1, ..., U^{k-1}$
7. Initialize S with the empty string : $S = \varepsilon$;
8. For each block $U^i = U^i_{\pi_0}, U^i_{\pi_1}, ..., U^i_{\pi_{n-1}}$, $i = 0, 1, ..., k-1$:
   - 8.1. Initialize the array π as follows: $\pi = \{\pi_0, \pi_1, ..., \pi_{n-1}\}$;
   - 8.2. Initialize the array π^{-1} as follows: $\pi^{-1}[\pi[j]] = j$, for $j = 0, 1, \cdots, n-1$;
   - 8.3. Compute the rank r of the sequence: $\pi_0, \pi_1, ..., \pi_{n-1}$ with the following parameters: n, π and π^{-1};
   - 8.4. Convert the value r to binary on l bits : $S' = (z_{l-1}...z_1 z_0)_2$;
   - 8.5. Concatenate the binary string S' to S: $S = S'||S$, where the symbol $||$ denotes the concatenation operation;
9. Return the secret S.

End

3.3 HTTP requests based on dependency between sequences

3.3.1 Overview

In this second scheme, the sequences of HTTP requests carried out by the sender are dependent on each other methodically in order to transfer the secret. In contrast, the queries within a sequence are unordered and therefore independent of each other. The communication model illustrated in Figure 2, shows how the secret is distributed on several IP addresses belonging to the sender. Each participant hides its discrete value by making an HTTP request on the tagged URL with the assigned discrete value as index. The tagged URLs contained in these HTTP requests are indexed from 0 to $n-1$. These requests are therefore identified by the IP address of the sender and the selected tagged URL. During the process, each participant can select untagged URLs without hindering the secret embedding. Likewise, normal users can select both tagged and untagged URLs without any impact in the dissimulation process. On the server-side, the receiver filters HTTP requests by retaining only those containing the IP address of the sender and the selected tagged URL. Then, these requests are ordered according to the list of IP addresses of the sender: ip_0, ip_1, ..., ip_{m-1}. Finally, the secret is obtained by extracting, in the same order, the indexes of the tagged URLs contained in these HTTP requests.

The added value of this scheme compared to the first one is that no sequential order constraint is established during the selection of the URLs by different IP addresses. They perform HTTP requests independently of each other without disturbing the decoding process.

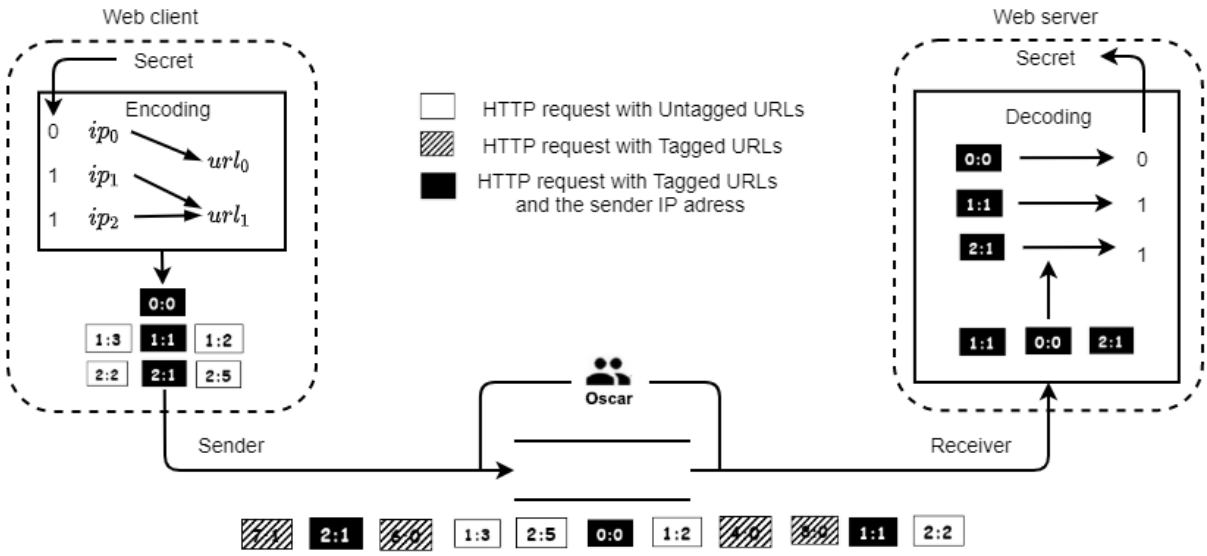


Figure 2: Overview of the proposed scheme.

3.3.2 The stego key

Two parameters must be shared between the sender and the receiver :

- A list of n tagged URLs: $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- A list of m sender IP addresses : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$;
- The base value B such that : $B \geq 2$ and $B = |U| = n$.

3.3.3 Embedding scheme

In this second scheme, the sender has control over multiple computers. These client machines are identified by their respective IP addresses. On the server-side, the receiver controls n web pages. The sender thus transcodes the secret to base B, then distributes the discrete values of the secret obtained between the client machines. Each client machine consults the URL that is mapped to the assigned discrete value. Therefore, the secret embedding algorithm is as follows:

Input
S : the secret message;

P : the IP addresses of the senders ;

U : the tagged URL list ;

B : the base value;
Output
U' : the URL selected after browsing;
Begin

1. The number of IP addresses of the senders is determined as follows : $m = |P|$;
2. The number of tagged URL is determined as follows : $n = |U|$;
3. The secret S is transcoded in base B as follows : $S = (z_{q-1}...z_1z_0)_B$,
   where $0 \leq z_i \leq B - 1$ and $m \geq q$;
4. The sender IP addresses $p_0, p_1, ..., p_{q-1}$ send HTTP requests respectively to URLs $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$.
5. For $i = 0, 1, ..., q - 1$:
   5.1. Get HTTP response of each request.

End

3.3.4 Extraction scheme

The server side receiver reads the incoming requests. A filter is then applied to extract only the tagged URLs from known source IP addresses. Subsequently, each URL is associated with the corresponding position in the set of tagged URLs. The discrete values thus obtained are assembled to form the secret in base B, which will then be converted into base 2 to obtain the initial secret. Therefore, the secret extraction algorithm is as follows:

Input

P : the IP addresses of the senders ;

B : the base value;

U : the tagged URL list ;

Output

S : the secret message;

Begin

1. Capture URLs and source IP addresses in order of opening HTTP requests;
2. Extract only the tagged one belonging to the list U;
3. Filter only URLs from the known IP address list P;
4. Sort the URLs in the numbering order of the source IP addresses of the sender : $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$;
5. Match each URL $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$ with its respective position : $z_0, z_1, ..., z_{q-1}$;
6. Represent the discrete values obtained in base B : $S' = (z_{q-1}...z_1 z_0)_B$;
7. Recover the secret S by converting S' to binary.

End

IV EVALUATION

This section focuses on the evaluation of the two proposed schemes in terms of estimating the capacity of hidden bits and the number of HTTP request sequences necessary for a given secret embedding.

4.1 The sequence Capacity and number of HTTP request sequences

In the two proposed methods, the sender makes sequences of n HTTP requests using m distinct IP addresses. The secret S to transfer is represented in the base B and the number of discrete values obtained is noted: $|(S)_B|$. Table 2 evaluates the capacity of the secret bits of an HTTP request sequence as well as the number of HTTP request sequences generated for n tagged URLs.

Table 2: The Sequence capacity and number of HTTP request sequences of the two proposed schemes

	Sequence capacity	# HTTP request sequences
Scheme 1	$\lfloor log_2(n!) \rfloor$	$\left\lceil \frac{\lvert(S)_B\rvert}{log_2(n!)*m} \right\rceil$
Scheme 2	$\lfloor log_2(n^m) \rfloor$	$\left\lceil \frac{\lvert(S)_B\rvert}{log_2(n^m)} \right\rceil$

As shown in Table 3 the first scheme has a higher secret bit capacity compared to the second scheme for a secret of 1024 bits. This is because the information contained in the permutations increase rapidly with respect to the increase in symbol space n. Similarly, the number of HTTP request sequences generated for the 1st scheme is very low compared to the second.

Table 3: Comparison example of the two proposed schemes

	n	m	Sequence capacity	# HTTP request sequences
Scheme 1	64	8	296	1
Scheme 2	64	8	48	22

To generalize, the higher the values of n and m, the greater the capacity and the lower the number of request sequences. The latter is visible in Table 2, with fractions that only depend on n and m.

4.2 Discussion

By comparing the two proposed schemes, the 1st having a dependency between HTTP requests within a sequence is slower in execution time because it follows an order of execution of requests to transfer the secret. However, it has a good capacity for secret bits and a reduced number of HTTP request sequences. On the contrary, the second scheme lifts this time limit with the absence of dependency between HTTP requests inside a sequence. But as a disadvantage the capacity of the secret bits is smaller and the number of HTTP request sequences higher. Finally, the two proposed schemes are complementary. Nevertheless, the 1st scheme is better than the second in terms of capacity of secret bits to transfer and the number of HTTP requests generated.

V CONCLUSION

In this paper, we have proposed two new secure covert channels, perfectly driven by online web operations that do not leave fingerprints. The design features of this undetectable scheme are based on lightweight requests for the secret encoding coupled to the cover media independence. The proposed scheme is simple and robust regardless of the size of the secret to be transmitted. Our scheme differs perfectly from others in that it does not modify or use files to conceal information. This considerably improves the scheme security. The comparative analysis showed that the 1st scheme is more efficient than the second in terms of capacity of secret bits to transfer and the number of HTTP requests generated.

REFERENCES

Publications

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. “Techniques for data hiding”. In: *IBM systems journal* 35.3.4 (1996), pages 313–336.

[2] C. T. Djamegni and M. Tchuenté. “A cost-optimal pipeline algorithm for permutation generation in lexicographic order”. In: *Journal of Parallel and Distributed Computing* 44.2 (1997), pages 153–159.

[3] W. Myrvold and F. Ruskey. “Ranking and unranking permutations in linear time”. In: *Information Processing Letters* 79.6 (2001), pages 281–284.

[4] J. Fridrich and M. Goljan. “Practical steganalysis of digital images: state of the art”. In: *Security and Watermarking of Multimedia Contents IV*. Volume 4675. International Society for Optics and Photonics. 2002, pages 1–13.

[5] D.-Y. Huang and T. Y. Yeo. “Robust and inaudible multi-echo audio watermarking”. In: *Pacific-Rim Conference on Multimedia*. Springer. 2002, pages 615–622.

[6] I. S. Moskowitz, L. Chang, and R. E. Newman. “Capacity is the wrong paradigm”. In: *Proceedings of the 2002 workshop on New security paradigms*. 2002, pages 114–126.

[7] D. Kirovski and H. S. Malvar. “Spread-spectrum watermarking of audio signals”. In: *IEEE transactions on signal processing* 51.4 (2003), pages 1020–1033.

[8] M. K. Johnson, S. Lyu, and H. Farid. “Steganalysis of recorded speech”. In: *Security, Steganography, and Watermarking of Multimedia Contents VII*. Volume 5681. International Society for Optics and Photonics. 2005, pages 664–672.

[9] W. Zeng, H. Ai, and R. Hu. “A novel steganalysis algorithm of phase coding in audio signal”. In: *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*. IEEE. 2007, pages 261–264.

[10] Z. Chen, L. Huang, Z. Yu, W. Yang, L. Li, X. Zheng, and X. Zhao. “Linguistic steganography detection using statistical characteristics of correlations between words”. In: *International Workshop on Information Hiding*. Springer. 2008, pages 224–235.

[11] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu. “A review on blind detection for image steganography”. In: *Signal Processing* 88.9 (2008), pages 2138–2157.

[12] L. Y. Por and B. Delina. “Information hiding: A new approach in text steganography”. In: *WSEAS international conference. Proceedings. Mathematics and computers in science and engineering*. Volume 7. World Scientific, Engineering Academy, and Society. 2008.

[13] R. Sridevi, A. Damodaram, and S. Narasimham. “EFFICIENT METHOD OF AUDIO STEGANOGRAPHY BY MODIFIED LSB ALGORITHM AND STRONG ENCRYPTION KEY WITH ENHANCED SECURITY.” In: *Journal of Theoretical & Applied Information Technology* 5.6 (2009).

[14] C.-C. Chang and T. D. Kieu. “A reversible data hiding scheme using complementary embedding strategy”. In: *Information Sciences* 180.16 (2010), pages 3045–3058.

[15] P. Jayaram, H. Ranganatha, and H. Anupama. “Information hiding using audio steganography–a survey”. In: *The International Journal of Multimedia & Its Applications (IJMA) Vol* 3 (2011), pages 86–96.

[16] K. F. Rafat and M. Sher. “Secure digital steganography for ASCII text documents”. In: *Arabian Journal for Science and Engineering* 38.8 (2013), pages 2079–2094.

[17] Z.-H. Wang, H.-R. Yang, T.-F. Cheng, and C.-C. Chang. “A high-performance reversible data-hiding scheme for LZW codes”. In: *Journal of Systems and Software* 86.11 (2013), pages 2771–2778.

[18] S. G. R. Ekodeck and R. Ndoundam. “PDF steganography based on Chinese Remainder Theorem”. In: *Journal of Information Security and Applications* 29 (2016), pages 1–15.

[19] M. Fan, P. Liu, H. Wang, and X. Sun. “Cross correlation feature mining for steganalysis of hash based least significant bit substitution video steganography”. In: *Telecommunication Systems* 63.4 (2016), pages 523–529.

[20] S. Sharma, A. Gupta, M. C. Trivedi, and V. K. Yadav. “Analysis of different text steganography techniques: a survey”. In: *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE. 2016, pages 130–133.

[21] A. Malik, G. Sikka, and H. K. Verma. “A high capacity text steganography scheme based on LZW compression and color coding”. In: *Engineering Science and Technology, an International Journal* 20.1 (2017), pages 72–79.

[22] K. Muhammad, J. Ahmad, N. U. Rehman, Z. Jan, and M. Sajjad. "CISSKA-LSB: color image steganography using stego key-directed adaptive LSB substitution method". In: *Multimedia Tools and Applications* 76.6 (2017), pages 8597–8626.

[23] M. Saidi, H. Hermassi, R. Rhouma, and S. Belghith. "A new adaptive image steganography scheme based on DCT and chaotic map". In: *Multimedia Tools and Applications* 76.11 (2017), pages 13493–13510.

[24] S. Chen and Z. Qu. "Novel quantum video steganography and authentication protocol with large payload". In: *International Journal of Theoretical Physics* 57.12 (2018), pages 3689–3701.

[25] T. Luo, G. Jiang, M. Yu, H. Xu, and W. Gao. "Sparse recovery based reversible data hiding method using the human visual system". In: *Multimedia Tools and Applications* 77.15 (2018), pages 19027–19050.

[26] S. Wu, S. Zhong, and Y. Liu. "Deep residual learning for image steganalysis". In: *Multimedia tools and applications* 77.9 (2018), pages 10437–10453.

[27] L. Xiang, W. Wu, X. Li, and C. Yang. "A linguistic steganography based on word indexing compression and candidate selection". In: *Multimedia Tools and Applications* 77.21 (2018), pages 28969–28989.

[28] X. Zuo, H. Hu, W. Zhang, and N. Yu. "Text semantic steganalysis based on word embedding". In: *International Conference on Cloud Computing and Security*. Springer. 2018, pages 485–495.

[29] A. Gutub and K. Alaseri. "Hiding shares of counting-based secret sharing via Arabic text steganography for personal usage". In: *Arabian Journal for Science and Engineering* (2019), pages 1–26.

[30] B. Khosravi, B. Khosravi, B. Khosravi, and K. Nazarkardeh. "A new method for pdf steganography in justified texts". In: *Journal of information security and applications* 45 (2019), pages 61–70.

[31] S. Jiang, D. Ye, J. Huang, Y. Shang, and Z. Zheng. "SmartSteganogaphy: Light-weight generative audio steganography model for smart embedding application". In: *Journal of Network and Computer Applications* (2020), page 102689.

[32] L. Moyou and R. Ndoundam. "Distributed data hiding in multi-cloud storage environment". In: *Journal of Cloud Computing: Advances, Systems and Applications* (2020).

[33] U. Pilania and P. Gupta. *Analysis and Implementation of IWT-SVD Scheme for Video Steganography*. 2020.

[34] A. K. Sahu and M. Sahu. "Digital image steganography and steganalysis: A journey of the past three decades". In: *Open Computer Science* 10.1 (2020), pages 296–342.

[35] A. K. Sahu and G. Swain. "Reversible image steganography using dual-layer LSB matching". In: *Sensing and Imaging* 21.1 (2020), page 1.

[36] M. Dalal and M. Juneja. "Steganography and Steganalysis (in digital forensics): a Cybersecurity guide". In: *Multimedia Tools and Applications* 80.4 (2021), pages 5723–5771.

[37] G. Deepthi, N. V. SriLakshmi, P. Mounika, U. Govardhani, P. L. Prassanna, S. Kavitha, and A. D. Kumar. *Linguistic Steganography Based on Automatically Generated Paraphrases Using Recurrent Neural Networks*. 2022.