Université de Lille
Faculté des Sciences et Technologies

# Les protocoles de vote par internet

**Maxime Lalisse**

Master Informatique
Master mention Informatique


Département d'Informatique
Faculté des Sciences et Technologies

mars, 2025

*Ce mémoire satisfait partiellement les pré-requis du module de Mémoire de Master, pour la 2e année du Master mention Informatique.*

**Candidat:** Maxime Lalisse, Nº 42327741,
maxime.lalisse.etu@univ-lille.fr
**Encadrant(e):** Sylvain SALVATI, sylvain.salvati@univ-lille.fr


Département d'Informatique
Faculté des Sciences et Technologies
Campus Cité Scientifique, Bât. M3 extension, 59655 Villeneuve-d'Ascq

mars, 2025

# Résumé

De nombreux protocoles de vote par internet existent, et cela reste un sujet de recherche très actif. Ils font souvent un usage extensif de la cryptographie, qui est le sujet de la première partie de ce mémoire. Nous exporeront les differentes primitives utilisables et comment les assembler en protocoles.

Cela servira de base à la seconde partie où l'on étudiera plus précisément les protocoles de vote de vote électronique dit "par internet" pour bien le différencier du vote électronique "sur site" mais avec des machines à voter.

Dans cette seconde partie, nous étudieront déjà ce qu'on peut demander à ces protocoles : les propriétés de sécurités. Puis nous étudieront certains de ces protocoles afin de comparer les avantages et inconvénients de chacun.

**Mots-clés** : vote par internet, vote électronique, protocoles cryptographiques, cryptographie à clé publique, vérifiabilité de bout en bout, secret du vote, confidentialité, résistance à la coercition, chiffrement homomorphe, preuves à divulgation nulle de connaissance

# Abstract

Many internet voting protocols exist, and this remains a very active area of research. These protocols often make extensive use of cryptography, which is the subject of the first part of this thesis. We will explore the different cryptographic primitives that can be used and quickly seen how they can be assembled into protocols.

This will serve as a foundation for the second part, where we will examine in more detail electronic voting protocols known as "internet voting" (to clearly distinguish them from "on-site" electronic voting, which uses voting machines).

In this second part, we will first analyze the security properties that we require from these protocols. We will then study some of them to compare their advantages and disadvantages.

**Keywords**: internet voting, electronic voting, cryptographic protocols, public key cryptography, end-to-end verifiability, ballot secrecy, privacy, coercion resistance, homomorphic encryption, zero-knowledge proofs

# Indice

# Table des figures

# Liste des Acronymes

| | |
|---|---|
| **3DES** | *Triple DES* |
| **AE** | *Authenticated encryption* |
| **AES** | *Advanced Encryption Standard* |
| **DES** | *Data Encryption Standard* |
| **DSA** | *Digital Signature Algorithm* |
| **E2E-E** | *End-to-end Encryption* |
| **E2E-V** | *End-to-end Verifiability* |
| **ECC** | *Elliptic Curve Cryptography* |
| **IV** | *Initialisation Vector* |
| **KEM** | *Key encapsulation mechanism* |
| **MAC** | *Message Authentication Code* |
| **MITM** | *Machine-in-the-middle* |
| **NIST** | *National Institute of Standards and Technology* |
| **PGP** | *Pretty Good Privacy* |
| **PKI** | *Public key infrastructure* |
| **PRF** | *Pseudo Random Function* |
| **SRC** | *Signature on randomizable ciphertexts* |
| **SSH** | *Secure SHell* |
| **TLS** | *Transport Level Security* |
| **ZKP** | *Zero-Knowledge Proof* |

## Chapitre 1

# Introduction

Nous commencerons par un tour d'horizon de la cryptographie, un champ de l'informatique théorique et des mathématiques qui s'est considérablement développé au cours du siècle dernier.

Des algorithmes assurant la confidentialité des communications sécurisées ont vu le jour, même si l'algorithme est connu de l'adversaire (la clé restant secrète) ou, plus récemment, même si la clé de chiffrement est publique (la clé de déchiffrement, distincte, restant évidemment secrète).

En plus de la confidentialité, la cryptographie permet d'assurer d'autres propriétés comme l'intégrité et l'authentification.

Récemment, des innovations dans le champ des "preuves à divulgation nulle de connaissance" permettent de prouver des affirmations générales en ne dévoilant que ce que l'on souhaite, ouvrant ainsi la voie à de nouveaux protocoles plus puissants.

La preuve formelle appliquée aux primitives ainsi qu'aux protocoles cryptographiques permet de prouver une fois pour toutes leur sécurité (dans le modèle considéré), afin d'éviter de mauvaises surprises, comme ce fut le cas avec le protocole de Needham–Schroeder,

où une attaque ne fut découverte que 17 ans plus tard par Gavin Lowe [1].

Cette première partie nous permettra d'entamer sérieusement la seconde, où nous parlerons du vote par internet, un domaine également en pleine expansion, expansion rendue possible par les avancées en cryptographie, dont il fait un usage intensif.

Nous parlons ici de vote (électronique) "par internet" afin de bien le différencier du vote électronique "sur site" avec des machines à voter. Dans notre cas, le vote s'effectue typiquement depuis chez soi avec son propre appareil.

Les conditions et l'environnement du vote sont donc très différents de ceux des systèmes modernes utilisant un isoloir, où l'on peut cacher son choix dans une enveloppe opaque avant de la glisser dans une urne transparente.

Les protocoles de vote par internet visent également à garantir le *secret du vote* (la confidentialité). Pour ce faire, les votes sont généralement chiffrés.

Afin que le votant puisse voter selon son choix, il faut également garantir une autre propriété, spécifique au monde du vote, appelée *résistance à l'influence* (ou *résistance à la coercition*). Il existe en effet un risque important que de tels protocoles facilitent la vente de votes, en permettant désormais une preuve de la transaction. Ainsi, dans les scénarios où la vente de votes constitue une menace sérieuse, des mécanismes visant à l'empêcher sont nécessaires.

Paradoxalement, nous souhaitons assurer la transparence du processus, ce qui implique doter les protocoles d'une nouvelle propriété, également issue du domaine du vote : la *vérifiabilité*. Nous y reviendrons plus loin.

Des outils permettant de prouver la sécurité de ces protocoles existent, ce qui est essentiel étant donné leur caractère critique. Ils sont quasi exclusivement automatisés, car le nombre de scénarios à considérer est souvent très élevé (exponentiel en fonction du nombre de participants). Ces outils constituent également un domaine de recherche très actif.

# Chapitre 2

# Cryptographie

## 2.1 Vue d'ensemble

La cryptographie a des origines très anciennes. Dès l'Antiquité, des techniques simples étaient employées pour protéger les messages. Les Grecs utilisaient, par exemple, la scytale, un bâton autour duquel était enroulé un ruban de cuir pour écrire un message crypté. Les Romains, quant à eux, utilisaient des techniques comme le chiffre de César, qui consiste à décaler chaque lettre d'un certain nombre de positions dans l'alphabet.

Au Moyen Âge, des méthodes plus complexes furent développées, comme le chiffre de Vigenère, qui repose sur l'utilisation d'une clé de chiffrement de longueur variable. Il est resté difficile à casser pendant plusieurs siècles, avant que de nouvelles techniques de cryptanalyse ne soient développées.

La révolution industrielle et les deux guerres mondiales ont marqué une étape majeure dans l'évolution de la cryptographie. Des machines comme Enigma, utilisée par l'Allemagne nazie, ont poussé les Alliés à développer des moyens sophistiqués pour casser ces systèmes. Le travail des cryptanalystes, notamment celui d'Alan

Turing, a été déterminant pour accélérer la fin de la Seconde Guerre mondiale.

Avec l'arrivée de l'informatique, la cryptographie est entrée dans une nouvelle ère. Les systèmes manuels et mécaniques ont laissé place à des algorithmes exécutés par des ordinateurs, ouvrant la voie à des techniques beaucoup plus complexes.

L'arrivée de l'informatique a aussi donné naissance à une nouvelle manière de faire de la cryptographie en raison de l'apparition de standards, à commencer par l'un des tout premiers grands standards : *Data Encryption Standard* (DES) en 1977. Les standards introduisent un nouveau paradigme : on utilise des algorithmes connus de tous, la sécurité du système ne reposant que sur le secret de la clé. C'est le principe de Kerckhoffs.

Peu après l'introduction de DES, des avancées en cryptanalyse et la trop courte longueur des clés de chiffrement (56 bits) ont conduit à l'introduction de *Triple DES* (3DES), qui consiste à appliquer le chiffrement DES trois fois de suite avec des clés différentes. La cryptanalyse de 3DES et ses faibles performances poussèrent le *National Institute of Standards and Technology* (NIST) à organiser un concours pour trouver son successeur. Rijndael remporta ce concours et fut donc désigné pour devenir *Advanced Encryption Standard* (AES).

Les algorithmes de chiffrement par bloc comme 3DES et AES doivent être utilisés en combinaison avec un mode d'opération. Ce mode précise comment enchaîner les blocs pour traiter des messages plus longs et inclut l'utilisation d'un *Initialisation Vector* (IV) aléatoire, ce qui permet de renforcer la sécurité en randomisant le chiffrement. Associé à un *Message Authentication Code* (MAC), le mode d'opération garantit également l'authenticité et l'intégrité des messages, donnant ainsi naissance au *chiffrement authentifié* ou *Authenticated encryption* (AE).

Des algorithmes alternatifs existent, notamment Blowfish, Twofish et ChaCha20-Poly1305, qui sont toujours d'actualité, tandis que des failles ont été trouvées dans RC4.

D'autres outils fondamentaux de la cryptographie ont également vu le jour, comme les *fonctions de hachage cryptographique*, qui transforment des données de taille variable en une empreinte unique de longueur fixe et jouent un rôle essentiel dans la vérification de l'intégrité des données.

Leur utilisation comme identifiants uniques relativement courts garantissant automatiquement l'intégrité des données est essentielle dans les systèmes pair-à-pair, où les participants

ne se font pas toujours confiance.

L'année 1978 voit aussi la naissance du système RSA, premier algorithme de chiffrement dit *à clé publique*, c'est-à-dire ne nécessitant pas d'échange préalable d'une clé secrète partagée. RSA est basé sur un problème difficile d'arithmétique : la difficulté de factorisation des grands nombres premiers.

En plus du chiffrement à clé publique, cette nouvelle cryptographie dite *asymétrique* permet des schémas de signature numérique, offrant de nouvelles possibilités d'authentification ne nécessitant pas non plus d'échange préalable d'un secret partagé.

Les *Zero-Knowledge Proof* (ZKP) ou *preuves à divulgation nulle de connaissance* représentent également une avancée prometteuse pour renforcer la confidentialité et la sécurité de nos données personnelles. Ces méthodes permettent de démontrer qu'une affirmation est vraie sans révéler la donnée privée sous-jacente. Par exemple, elles rendent possible de prouver que vous êtes majeur sans divulguer votre âge exact ou d'autres informations personnelles.

Le *chiffrement homomorphe* est une technique permettant d'exécuter des calculs (addition et/ou multiplication) sur des données chiffrées.

Jusqu'aux années 1990, la cryptographie restait principalement réservée aux États et aux grandes industries. Cependant, en 1991, l'apparition de *Pretty Good Privacy* (PGP) démocratisa l'accès à la cryptographie, offrant à chacun un niveau de sécurité auparavant réservé aux États. Cela alimenta de nombreux débats sur le droit des citoyens à la protection de leurs données personnelles, face à des gouvernements préoccupés par des enjeux de sécurité nationale.

Aujourd'hui, la cryptographie est omniprésente dans notre vie quotidienne. La majorité de nos communications et transactions en ligne sont sécurisées grâce à des protocoles cryptographiques conçus pour garantir la confidentialité, l'intégrité et l'authenticité des données échangées. Ces protocoles, tels que *Transport Level Security* (TLS) ou *Secure SHell* (SSH), protègent les échanges sur Internet, qu'il s'agisse d'envoyer des messages, de réaliser des paiements en ligne ou d'accéder à des sites web.

À l'avenir, le développement d'ordinateurs quantiques rendra obsolètes la plupart des algorithmes de cryptographie asymétrique que nous utilisons aujourd'hui. En effet, des algorithmes quantiques comme celui de Shor permettraient de casser efficacement des systèmes basés sur des problèmes mathématiques complexes, tels que la difficulté de factorisation de grands nombres en facteurs premiers (utilisée par RSA) ou le

problème du logarithme discret dans les groupes finis (utilisé par ElGamal). Le développement d'algorithmes dits "résistants aux ordinateurs quantiques" est donc un sujet de recherche très actif, et un travail de normalisation du NIST est en cours.

## 2.2 Chiffrement à clé secrète (symétrique)

Le chiffrement à clé secrète est le cas d'usage classique de la cryptographie. Il suppose que l'expéditeur partage un secret avec le destinataire. On doit préciser *à clé secrète* pour bien le différencier des nouvelles techniques de chiffrement *à clé publique*. Comme il suppose que l'expéditeur et le destinataire partagent la même clé, on l'appelle aussi *symétrique*, tandis que le chiffrement *à clé publique* est aussi dit *asymétrique*, car les clés de chiffrement et de déchiffrement sont différentes.

Le secret partagé entre l'expéditeur et le destinataire peut être tout simplement la méthode de chiffrement, comme c'est le cas dans les méthodes de chiffrement historiques telles que la Scytale ou le chiffre de César, où l'on suppose que les espions (ou attaquants) ont des ressources très limitées. Il peut aussi être composé d'un secret (mot de passe) qu'il est nécessaire de connaître pour déchiffrer le message, comme c'est le cas dans le chiffre de Vigenère ou Enigma.

Les méthodes de chiffrement modernes, au contraire, supposent des attaquants avec des ressources importantes, quoique finies, afin de se prémunir même contre de puissants adversaires disposant d'un grand nombre de (super-)ordinateurs. On part du principe que "l'adversaire connaît le système", et donc que la sécurité repose uniquement sur la clé partagée : c'est ce qu'on appelle le *principe de Kerckhoffs*.

### 2.2.1 Scytale

La *scytale* est l'un des dispositifs de chiffrement les plus anciens connus. Son fonctionnement repose sur une méthode simple : une lanière de cuir est enroulée autour d'un bâton en bois d'un diamètre précis (la scytale). Cette opération produit une transposition des lettres du message. Pour déchiffrer le message, le destinataire doit disposer d'un bâton ayant exactement le même diamètre. Ce procédé ne modifie pas les symboles utilisés, mais réorganise simplement leur ordre, créant ainsi un anagramme du message original. Le destinataire doit ensuite rétablir l'ordre correct des lettres pour en comprendre le contenu.

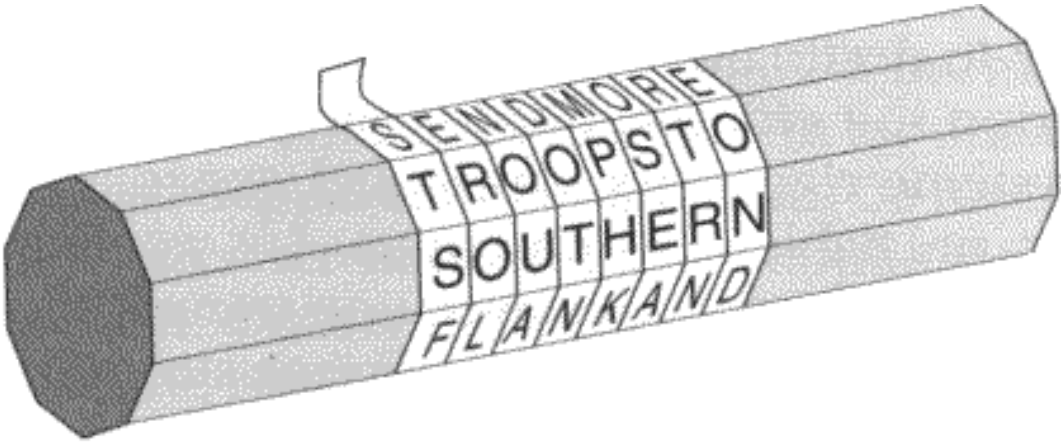


FIGURE 2.1 – Scytale

### 2.2.2 Chiffre de César

Le *chiffre de César* est un code secret très ancien, existant déjà depuis l'Antiquité et ayant notamment été utilisé par César pour certaines de ses correspondances. Il consiste à décaler chaque caractère de n positions vers l'avant dans l'alphabet. Il est aussi appelé *chiffrement par décalage*.

Avec un décalage de 3 vers la droite (comme César), A devient D, B devient E... (voir Figure 2.2).

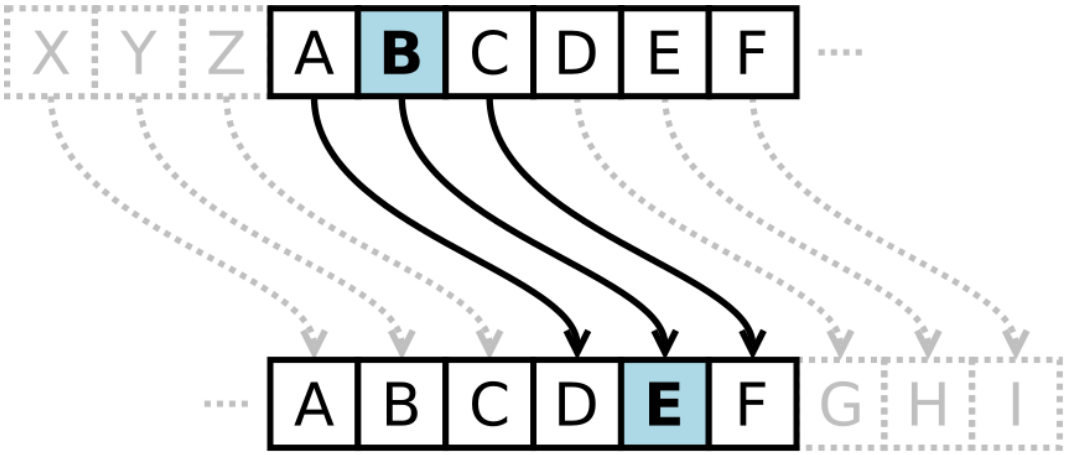


FIGURE 2.2 – Chiffrement par décalage

Le message **SALUT** deviendrait donc **VDOXW**. Pour déchiffrer, il suffit de décaler chaque lettre de 3 positions vers la gauche.

Un *chiffre de César* particulier est l'algorithme **ROT13** (décalage de 13 lettres), où la fonction de chiffrement est identique à celle de déchiffrement. Cela est dû au fait

que l'alphabet comporte 26 lettres : en décalant deux fois de 13 lettres, on revient à la position de départ (voir Figure 2.3).

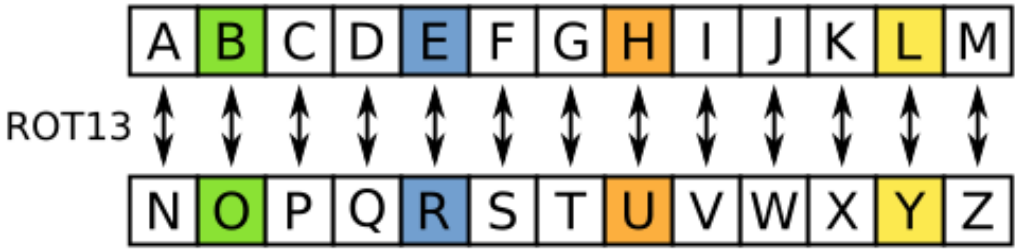


Figure 2.3 – ROT13

Le chiffrement par décalage est l'un des plus faciles à casser. Il ne modifie pas la fréquence des lettres, ce qui le rend vulnérable à l'analyse fréquentielle. De plus, il n'existe que 25 décalages possibles, ce qui le rend particulièrement vulnérable aux attaques par force brute.

### 2.2.3 Chiffre de Vigenère

C'est une méthode de chiffrement plus élaborée et plus récente. Elle a été décrite par Blaise de Vigenère dans son traité des chiffres paru en 1586, même si l'on sait qu'un système similaire était déjà utilisé dès le XIIIème siècle.

Le *chiffre de Vigenère* est semblable au chiffre de César, sauf que le décalage change à chaque lettre en suivant les lettres du mot de passe (ou clé). Pour ce faire, chaque lettre du mot de passe désigne la distance de décalage selon son indice dans l'alphabet (A : 0, B : 1, ...).

Une fois que toutes les lettres de la clé ont été utilisées, on recommence au début. Ainsi, pour une clé de longueur 1, on revient au *chiffre de César*, tandis que la sécurité augmente avec la taille de la clé. Dans le cas extrême où la clé a la même taille que le message, on se retrouve dans le cas du *chiffre de Vernam*.

Cette méthode de chiffrement a très bien fonctionné pendant un temps, notamment grâce à sa bonne résistance à l'analyse fréquentielle, mais de nouvelles méthodes de cryptanalyse permettant de le casser ont fini par être développées.

La première difficulté est de trouver la taille de la clé. On pourra ensuite utiliser les techniques classiques, telles que l'analyse fréquentielle sur les sous-segments de la

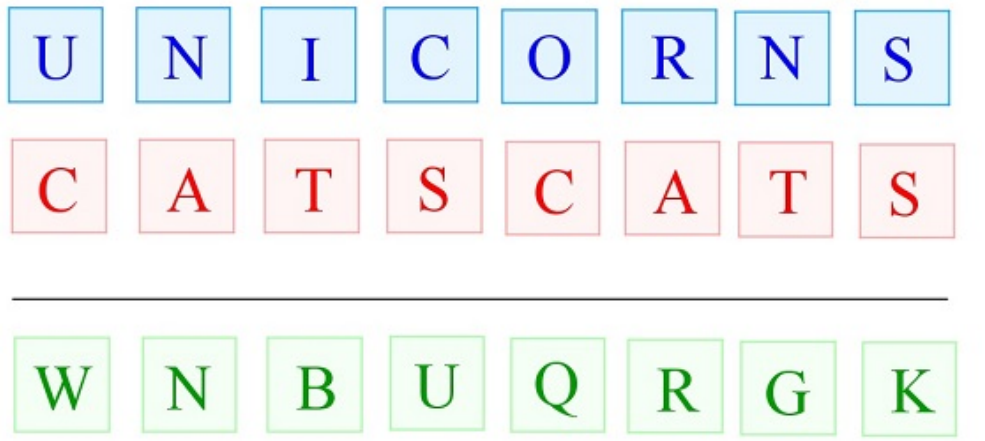


FIGURE 2.4 – Chiffre de Vigenère

taille de la clé.

Une méthode efficace pour déterminer la taille de la clé est décrite en 1863 par Friedrich Kasiski (le test de Kasiski). Des méthodes encore plus efficaces sont découvertes au XXème siècle, telles que celles basées sur l'*indice de coïncidence*.

### 2.2.4 Enigma

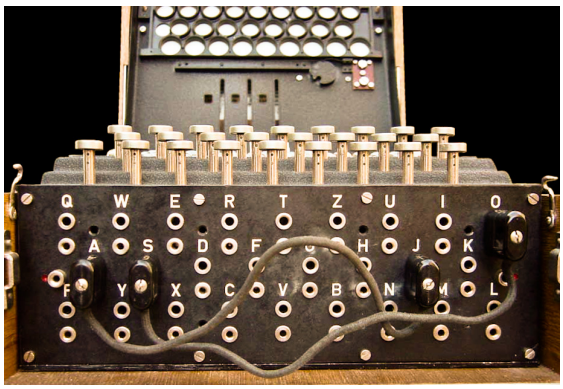

FIGURE 2.5 – La machine Enigma

Enigma est probablement l'un des systèmes de chiffrement purement mécaniques les plus sophistiqués.

Elle est composée d'une succession de rotors qui opèrent une transformation basée sur un chiffrement par substitution, dont la clé change à chaque pression d'une touche.

Heureusement pour les Alliés, cette machine n'était pas infaillible. Le décryptage de ses communications a joué un rôle crucial dans l'issue de la Seconde Guerre mondiale en contribuant à abréger le conflit.

### 2.2.5 Chiffre de Vernam

Dans le chiffre de Vernam, également connu sous le nom de *masque jetable* (*one-time pad* en anglais), la clé est exactement de la même longueur que le message.

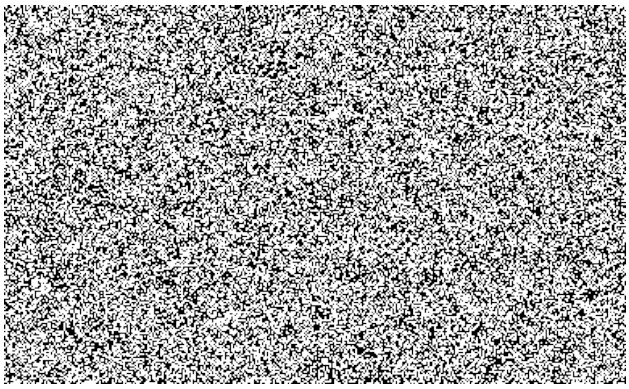

FIGURE 2.6 – Masque jetable

Chaque symbole du message est combiné avec le caractère correspondant de la clé à l'aide d'une opération logique, souvent une addition modulaire ou une opération XOR.

C'est un chiffre qui, contrairement aux précédents, offre une *sécurité parfaite*, sous les conditions cruciales que la clé soit générée de manière totalement aléatoire et qu'elle ne soit jamais réutilisée (d'où son nom de *jetable*). Une réutilisation de la clé ruinerait, au contraire, toute la sécurité du système.

### 2.2.6 Chiffrement moderne

Les algorithmes de chiffrement modernes sont beaucoup plus difficiles à casser : en effet, pour les plus récents comme AES, on ne connaît pas d'attaque significativement plus efficace que la force brute.

Ils reposent sur les notions de *confusion* et de *diffusion*, introduites par Claude Shannon [2]. La *diffusion* (aussi appelée *effet avalanche*) signifie qu'une modification minimale de l'entrée entraîne une modification drastique de la sortie, rendant toute

tentative d'analyse difficile. La *confusion*, quant à elle, garantit qu'il n'existe pas de relation exploitable entre la clé et le texte chiffré.

On distingue deux types de constructions : les *algorithmes de chiffrement de flux* et les *algorithmes de chiffrement par bloc*, même s'il est souvent possible de transformer l'un en l'autre. Par exemple, le mode d'opération CTR permet à un algorithme de chiffrement par bloc de se comporter de manière similaire à un algorithme de chiffrement de flux. Ces algorithmes étant déterministes, on ajoute un nonce aléatoire pour les randomiser. C'est l'IV.

En pratique, et pour éviter toute une série d'attaques, on cherche à garantir l'intégrité du message et l'authenticité du correspondant grâce à un MAC. On parle alors de "chiffrement authentifié" ou AE.

### 2.2.7 Chiffrement de flux

Les algorithmes de chiffrement de flux sont généralement de simples générateurs de flux pseudo-aléatoires, qui sont ensuite mélangés via un XOR avec le message en clair.

#### RC4

RC4 est un générateur de bits pseudo-aléatoires acceptant des clés de tailles variables. Il génère des nombres pseudo-aléatoires octet par octet, notamment en utilisant un état interne secret de 256 octets.

La spécification de l'algorithme est longtemps restée secrète avant d'être publiée de manière anonyme sur une liste de diffusion, probablement à la suite d'une rétro-ingénierie.

RC4 est une marque déposée par RSA Security, mais l'algorithme lui-même n'a jamais été breveté. Des implémentations non officielles existent sous d'autres noms, comme *ARC4*, et sont libres d'utilisation.

Des failles majeures ont été découvertes dans RC4 [3], le rendant aujourd'hui peu sûr. Il convient d'utiliser des algorithmes plus modernes, tels que ChaCha20.

### Salsa20 et ChaCha20

Développés par Daniel J. Bernstein et publiés dans le domaine public, Salsa20 [4] et ChaCha20 [5] sont des chiffrements de flux modernes basés sur un générateur de nombres pseudo-aléatoires ne reposant que sur trois opérations : l'addition, le XOR (OU exclusif) et le décalage. On appelle les chiffrements utilisant uniquement ces trois opérations des chiffrements ARX (Addition-Rotation-XOR), comme illustré sur la figure 2.7.

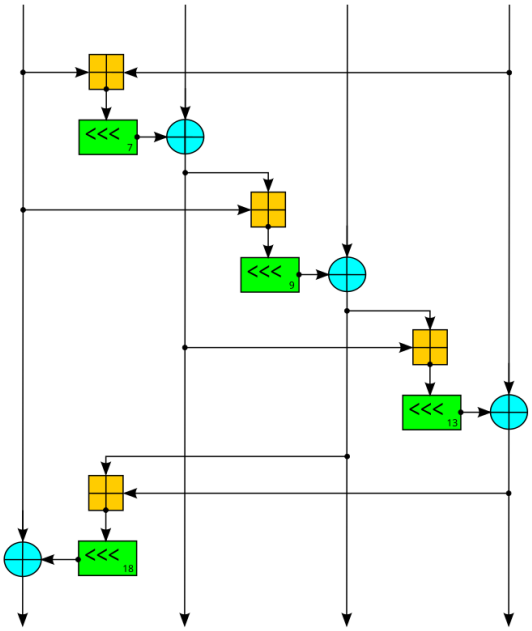


FIGURE 2.7 – Un tour de Salsa20

Ces trois opérations s'exécutent en temps constant, de manière très efficace sur les processeurs modernes, ce qui permet notamment de se prémunir contre les attaques temporelles.

Une caractéristique notable de Salsa20 et ChaCha20 par rapport aux chiffrements précédents est la possibilité de commencer le chiffrement ou le déchiffrement à un index arbitraire. Cela permet, par exemple, de déchiffrer une partie d'un fichier sans avoir à traiter son début.

### 2.2.8 Chiffrement de bloc

Un algorithme de chiffrement de bloc opérant par blocs de données de taille fixe, il est nécessaire de segmenter notre message selon la taille de ces blocs (et d'ajouter du padding si nécessaire).

L'objectif est de concevoir des permutations pseudo-aléatoires, c'est-à-dire des fonctions qui se comportent comme des fonctions pseudo-aléatoires (*Pseudo Random Function* (PRF) en anglais), tout en restant réversibles afin de permettre le déchiffrement.

Pour atteindre ces propriétés, les chiffrements modernes utilisent différentes constructions, telles que les réseaux de Feistel ou les réseaux de permutation-substitution, et intègrent des composants tels que les boîtes de substitution (S-Boxes) et les boîtes de permutation (P-Boxes), ce qui assure les propriétés de *diffusion* et de *confusion*.

Un réseau de Feistel, par exemple, est une structure utilisée dans certains algorithmes comme DES. Il a été démontré qu'un réseau de Feistel (figure 2.8) instancié avec une fonction pseudo-aléatoire devient une permutation pseudo-aléatoire indistinguable dès que sa profondeur dépasse trois tours [6]. D'autres constructions sont possibles, comme les *réseaux de permutation-substitution* (figure 2.9) utilisés par AES.

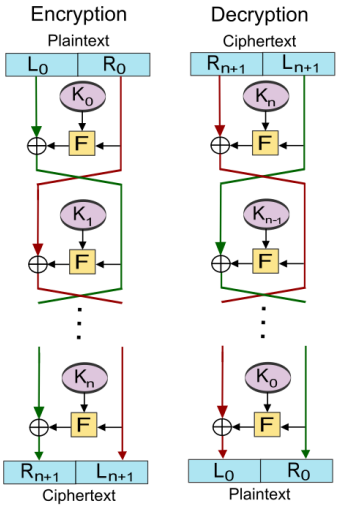


FIGURE 2.8 – Réseau de Feistel

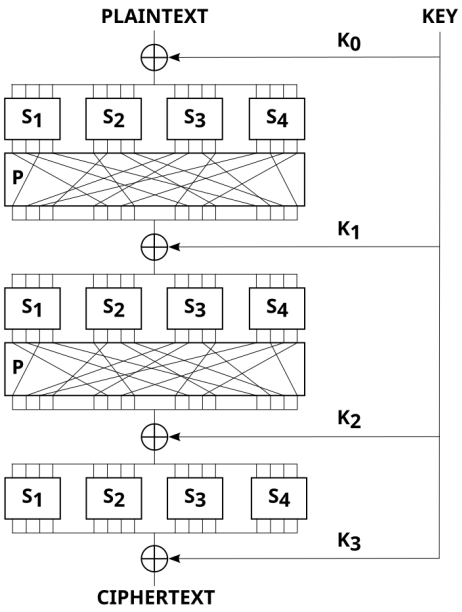


FIGURE 2.9 – Réseau de permutation-substitution

La manière d'enchaîner ces blocs, en prenant en compte l'IV et l'éventuel MAC, est appelée le *mode d'opération*. Celui-ci détermine les propriétés de sécurité du

chiffrement. Certains sont insécures, comme ECB, qui n'est pas randomisé (il chiffre donc toujours des blocs identiques de la même manière). On préférera, par exemple, CFB (figure 2.10), qui est randomisé grâce à un IV et utilise la sortie du bloc précédent pour fournir l'IV du suivant.

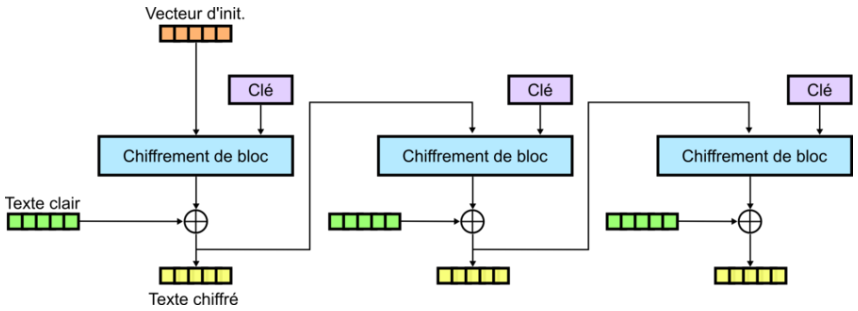


FIGURE 2.10 – Cipher Feedback Block (CFB)

Certains modes, comme CTR (figure 2.11) ou OFB, permettent de transformer un algorithme de chiffrement par bloc en chiffrement de flux.

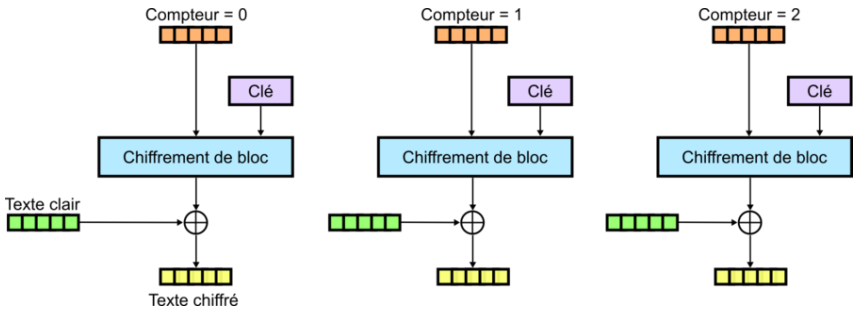


FIGURE 2.11 – CounTeR (CTR)

Aujourd'hui, on privilégie les modes garantissant l'authenticité des données, comme GCM ou CCM. C'est ce qu'on appelle le *chiffrement authentifié*.

## 2.3 Chiffrement à clé publique (asymétrique)

Les possibilités de chiffrement sans échange préalable de clé sont exposées pour la première fois par Diffie et Hellman en 1976 [7], en même temps que le concept de signatures, que l'on verra dans un chapitre dédié. Il a fallu attendre 1978 pour voir se concrétiser un tel système avec ce que l'on appelle aujourd'hui RSA [8], puis avec le cryptosystème d'ElGamal [9].

Le chiffrement à clé publique se base sur des problèmes difficiles à résoudre, comme la difficulté de factorisation de grands nombres pour RSA ou le problème du logarithme

discret pour ElGamal. Souvent, mais pas exclusivement, il repose sur l'arithmétique modulaire et la théorie des groupes finis.

En pratique, le chiffrement à clé publique est très souvent combiné avec le chiffrement à clé secrète. On utilise le premier pour négocier une clé partagée utilisable pour le second. On appelle cela le *chiffrement hybride*. Lorsque le chiffrement à clé publique sert uniquement à transmettre une clé, on parle de mécanisme d'encapsulation de clé, ou *Key encapsulation mechanism* (KEM) en anglais.

### 2.3.1 Chiffrement RSA

RSA [8], nommé d'après ses auteurs Ronald Rivest, Adi Shamir et Leonard Adleman, est le premier algorithme de chiffrement à clé publique publié. Il a été soumis à un brevet de 1983 à 2000.

Dans RSA, le destinataire doit choisir au hasard deux très grands nombres premiers (appelés $p$ et $q$). Il calcule $N = p \times q$ et choisit un exposant de chiffrement $e$ (3 est souvent un bon choix). Il publie ($N$, $e$) et garde ($p$, $q$) secrets. Comme il est le seul à connaître $p$ et $q$, il est le seul à pouvoir calculer un exposant de déchiffrement $d$ via une certaine relation mathématique.

Pour chiffrer, on élève simplement le message à l'exposant $e$ :

$$c \equiv m^e \pmod{N}$$

Pour déchiffrer, il suffit d'élever le chiffré à l'exposant $d$ :

$$m \equiv c^d \pmod{N}$$

La sécurité de RSA est donc basée sur la difficulté de trouver $d$ à partir de $N$ et $e$. On pense que c'est équivalent à la difficulté de factoriser $N$, c'est-à-dire retrouver $p$ et $q$ à partir de $N$. On dit donc que le problème RSA est basé sur la difficulté de factorisation des grands nombres.

Les nombres $p$ et $q$ doivent absolument être gardés secrets. Il est très difficile de les retrouver à partir de $N$. C'est le *problème de factorisation*, dont découle le *problème*

*RSA*. On suppose ces problèmes difficiles, voire infaisables pour des ordinateurs classiques si ces nombres sont choisis suffisamment grands. En revanche, comme une grande partie des systèmes de chiffrement à clé publique, des algorithmes quantiques existent pour résoudre ces problèmes de manière efficace [10].

RSA permet seulement de chiffrer des données de taille inférieure à $N$. Il est souvent utilisé pour négocier une clé de chiffrement symétrique qui peut ensuite être utilisée pour chiffrer efficacement des données de taille arbitraire (chiffrement hybride).

Le chiffrement RSA sans padding étant déterministe, cela signifie que pour un même message clair, le texte chiffré correspondant sera toujours le même. Cela pose de nombreux problèmes de sécurité. On encapsule donc le message en le randomisant avec des algorithmes comme PKCS#1 v1.5 ou RSAES-OAEP.

### 2.3.2 Échange de clé de Diffie-Hellman

L'échange de clé de Diffie-Hellman [11] n'est pas à proprement parler une méthode de chiffrement, mais il permet de calculer une clé partagée qui peut ensuite être utilisée dans un système de chiffrement à clé secrète. Ce procédé est appelé chiffrement hybride et est largement utilisé en pratique pour des raisons de performances.

Il repose sur un groupe multiplicatif des entiers modulo un très grand nombre premier $p$. On note cet ensemble $(\mathbb{Z}/p\mathbb{Z})^\times$. On choisit également un générateur du groupe $g$. Tout nombre premier inférieur à $p$ convient, et en pratique, 3 est généralement un bon choix.

$A \to B$ : Choisit une clé privée $a$, calcule et envoie sa clé publique $A = g^a \mod p$.
$B \to A$ : Choisit une clé privée $b$, calcule et envoie sa clé publique $B = g^b \mod p$.
$A$ : Calcule la clé commune $K = B^a \mod p$.
$B$ : Calcule la clé commune $K = A^b \mod p$.

Contrairement à RSA, sa sécurité n'est pas basée sur le *problème de factorisation*, mais sur le *problème du logarithme discret*.

### 2.3.3 Chiffrement ElGamal

Comme pour l'échange de clé de Diffie-Hellman, il repose sur les groupes multiplicatifs définis sur le corps fini $\mathbb{Z}/p\mathbb{Z}$. Sa sécurité est également basée sur le *problème du logarithme discret*.

Le chiffrement ElGamal combine la création d'une clé temporaire, inspirée de l'échange de clés Diffie-Hellman, avec un mécanisme de masquage « parfaitement dissimulant » (*perfectly hiding*), que l'on peut voir comme un équivalent du *one-time pad* en arithmétique modulaire.

Le destinataire choisit aléatoirement un nombre entre 0 et $p-1$, noté $sk$, qui représente sa clé privée. Il calcule ensuite $h = g^{sk}$ et la publie : c'est la clé publique.

Le message chiffré est constitué de deux éléments :

$$(\alpha, \beta) = (h^a, h^a \times m)$$

La taille du message est limitée, car il doit être un élément du groupe, c'est-à-dire compris entre 0 et $p-1$. Le texte chiffré est composé de deux éléments du groupe, ce qui signifie qu'il est deux fois plus long que le message en clair.

Le chiffrement ElGamal est *malléable*, c'est-à-dire qu'il est possible de générer un texte chiffré valide à partir d'un autre, ce qui en fait un système de chiffrement *homomorphe*.

Par exemple, avec deux textes chiffrés $(\alpha, \beta) = (h^a, h^a \times m)$ et $(\alpha', \beta') = (h^{a'}, h^{a'} \times m')$, on peut calculer :

$$(\alpha \times \alpha', \beta \times \beta') = (h^{a+a'}, h^{a+a'} \times (m \times m'))$$

soit le texte chiffré de $m \times m'$.

Ce chiffrement présente un *homomorphisme multiplicatif*, car il est possible de multiplier les messages en clair uniquement à partir de leurs versions chiffrées.

Il existe une variante d'ElGamal permettant un *homomorphisme additif*, qui est généralement utilisée dans le vote électronique :

$$(\alpha, \beta) = (h^a, h^a \times g^m)$$

Ici, le message est "dans l'exposant". Un inconvénient est qu'il devient plus difficile de déchiffrer de grandes valeurs, car retrouver $m$ à partir de $g^m$ nécessite d'essayer toutes les valeurs possibles.

Cette variante possède un *homomorphisme additif*, car il devient possible d'additionner les messages chiffrés par une simple opération.

Par exemple, avec deux textes chiffrés $(\alpha, \beta) = (h^a, h^a \times g^m)$ et $(\alpha', \beta') = (h^{a'}, h^{a'} \times g^{m'})$, on peut calculer :

$$(\alpha \times \alpha', \beta \times \beta') = (h^{a+a'}, h^{a+a'} \times g^{m+m'})$$

soit le texte chiffré de $m + m'$.

### 2.3.4 Cryptosystème de Cramer-Shoup

Le cryptosystème de Cramer-Shoup [12] repose sur le chiffrement ElGamal et corrige sa malléabilité. Il est résistant aux attaques adaptatives à texte chiffré choisi (IND-CCA2), l'une des attaques les plus puissantes, et sa sécurité est même prouvée. En revanche, les clés et les textes chiffrés sont plus longs. Comme il n'est pas malléable, il ne peut pas être utilisé pour du chiffrement homomorphe.

### 2.3.5 Cryptographie sur les courbes elliptiques

Les courbes elliptiques sont des courbes mathématiques définis par une formule de la forme $y^2 = x^3 + ax + b$ (en forme simplifiée, dite de Weierstrass). On peut y construire un groupe en définissant l'opération du groupe (l'"addition") comme une opération géométrique où l'on trace une droite passant par ces deux points, comme

sur la figure 2.12. Cette droite coupera la courbe en un troisième point. Enfin, on prendra le négatif de ce point comme résultat de l'addition.

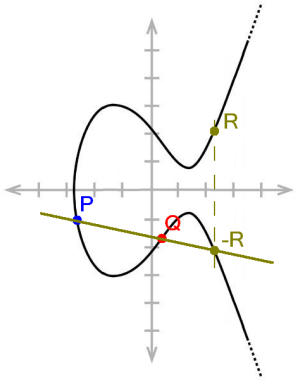


FIGURE 2.12 – Addition sur une courbe elliptique

La cryptographie sur les courbes elliptiques ou *Elliptic Curve Cryptography* (ECC) propose de remplacer les groupes finis habituels par ce groupe défini sur les courbes elliptiques.

Il existe en effet un problème analogue au *problème du logarithme discret* : le problème du *logarithme discret sur courbes elliptiques*, dont on ne connaît pas non plus d'algorithme efficace (et dont nous pensons qu'il n'en existe pas de beaucoup plus efficace que la force brute en informatique classique), contrairement à l'informatique quantique où il existe des algorithmes efficaces.

L'intérêt est que les courbes elliptiques permettent des clés plus courtes pour le même niveau de sécurité, ce qui implique moins d'opérations et donc des algorithmes plus efficaces. Ainsi, Diffie-Hellman sur les courbes elliptiques, ainsi que le chiffrement ElGamal adapté aux courbes elliptiques, sont aussi sécurisés avec des clés plus courtes.

La cryptographie sur les courbes elliptiques a aussi donné naissance à une nouvelle cryptographie : la cryptographie basée sur les pairing.

Nous allons voir rapidement quelques courbes elliptiques utilisées en pratique.

#### secp256r1

Cette courbe elliptique est la plus commune, notamment pour les signatures avec ECDSA. C'est celle recommandée par le NIST, est utilisée dans TLS et SSH.

Secp256r1 est définie sur le corps fini $\mathbb{F}_p$ avec

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

L'équation de la courbe est donnée sous la forme de Weierstrass :

$$y^2 = x^3 + ax + b$$

avec les paramètres :

$$a = -3$$

$b$ = 0x5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B

Le générateur g est donné par :

$x$ = 0x6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296

$y$ = 0x4FE342E2FE1A7F9B8EE7EB4A7C0F9E162BCE33576B315ECECBB6406837BF51F5

Le générateur est choisi comme sortie de la fonction SHA-1 avec la seed $C49D360886E704936A6678E1139$
SHA-1 produisant une sortie proche de l'aléatoire, il est peu probable qu'une backdoor ait pu être insérée de cette manière, mais si l'on veut utiliser une courbe avec des paramètres mieux justifiés, on peut utiliser *Curve25519*.

**Curve25519**

Elle est conçue par Daniel J. Bernstein. Les paramètres sont choisis de manière méticuleuse, de façon à rendre négligable la probabilité qu'ils comportent une porte dérobés.

Curve25519 est définie sur le corps fini $\mathbb{F}_p$, où $p$ est donné par :

$$p = 2^{255} - 19$$

L'équation de la courbe est exprimée sous la forme de Montgomery :

$$y^2 = x^3 + Ax^2 + x$$

avec le coefficient :

$$A = 486662$$

Le générateur g est donné par :

$$x = 9$$

On obtiendra y en résolvant l'équation de la courbe pour $x = 9$.

## 2.4 Signatures

Les schémas de signatures sont un champ de la cryptographie asymétrique et permettent à une entité ayant généré une paire de clés asymétriques de *signer* des messages.

Ils ont la propriété d'être non-répudiables : une fois faites, il est impossible de nier les avoir réalisées.

L'entité, identifiée par sa clé publique, peut utiliser sa clé privée pour signer les données qu'elle souhaite (*completeness*). On peut s'assurer que seul le possesseur de la clé a pu générer cette signature (*soundness*).

Comme on est généralement limité dans la taille du message signé (par exemple 256 bits), on signe généralement le hachage de la donnée, permettant ainsi de signer des documents arbitrairement longs.

Depuis les premiers schémas de signature (schéma de signature sur RSA, signatures de Schnorr), de nouveaux types de signatures comme BLS ou BBS+ apportent de nouvelles propriétés comme l'agrégation de signatures.

On peut généraliser les schémas de signature en attestant en Zero-Knowledge la connaissance d'une clé secrète, ce qui mène aux Signature-of-Knowledge [13], plus générales, où l'on atteste la connaissance d'une information quelconque.

### 2.4.1 RSA

RSA propose aussi un schéma de signatures basé sur le problème RSA. Il fonctionne de manière très similaire (“inverse”) au chiffrement RSA. En effet, le créateur d'un groupe backdooré (qui connaît $P$ et $Q$ tels que $N = P \times Q$) choisit un $e$ quelconque du groupe et sait générer un inverse $d$ qui permet d'inverser l'exponentiation par $e$. Dans le chiffrement RSA, on choisit $e$ faible (par exemple 3), et on calcule son inverse $d$ qu'on garde secret. Dans le schéma de signature sur RSA, on choisit un $e$ quelconque (et difficile à trouver). On dévoile $d$, et on est le seul à connaître $e$ qui permet de produire un message chiffré déchiffrable par $d$, ce qui sert de signature.

### 2.4.2 *Digital Signature Algorithm* (DSA)

DSA est un schéma de signature basé sur le problème du logarithme discret, standardisé par le NIST en 1991. Il est un peu plus complexe que la signature de Schnorr, qui était protégée par brevet.

### 2.4.3 Signature de Schnorr

La signature de Schnorr est un schéma de signature basé sur le problème du logarithme discret. Elle était protégée par brevet jusqu'en 2010. C'est un premier exemple de ZKP rendu non interactif grâce à l'heuristique de Fiat-Shamir.

### 2.4.4 Signature BBS

Elle permet de générer une seule signature de taille constante pour plusieurs messages. Il sera ensuite possible de dévoiler un sous-ensemble des messages signés tout en gardant l'autre partie cachée.

## 2.5 Preuves à divulgation nulle de connaissance

### 2.5.1 Définition

Les systèmes de *preuve à divulgation nulle de connaissance* permettent de démontrer la validité d'une proposition tout en gardant secret certains paramètres impliqués.

Ces systèmes de preuve impliquent deux parties : le prouveur et le vérificateur, et sont donc des protocoles. Par exemple, pour la proposition :

$$\text{Je connais } x \text{ tel que } y = f(x)$$

le prouveur peut générer une preuve $\pi = \text{proof}(y, x)$ pour convaincre le vérificateur qu'il connaît un $x$ vérifiant $y = f(x)$, sans dévoiler $x$.

### 2.5.2 Propriétés

Tout système de preuve doit respecter les deux premières propriétés. La troisième et dernière propriété est spécifique aux systèmes de preuve à divulgation nulle de connaissance.

#### Complétude (Completeness)

Quand la proposition est vraie, il est toujours possible de générer une preuve convaincante.

#### Consistance (Soundness)

En revanche, quand la proposition est fausse, il n'est jamais possible de générer une preuve convaincante.

#### Divulgation nulle de connaissance (Zero-Knowledge)

La preuve ne dévoile aucune information sur les paramètres privés impliqués.

Cette propriété peut se décliner en deux versions :

— Version forte (*Full Zero-Knowledge*) : La confidentialité est garantie même en présence de vérificateur malicieux.
— Version affaiblie (*Honest-Verifier Zero-Knowledge*) : La confidentialité est garantie au moins quand le vérificateur respecte le protocole.

### 2.5.3 Vue d'ensemble

#### Preuve de connaissance interactive

Les systèmes de *preuve à divulgation nulle de connaissance* (Zero-Knowledge Proof, ZKP) trouvent leur origine dans les systèmes de *preuve de connaissance interactive* [14], où un prouveur peut convaincre un vérificateur qu'il possède une information, par exemple en répondant systématiquement de manière correcte à un défi auquel seule la connaissance de cette information permet de répondre.

#### Non-Interactivité

Des développements plus récents ont permis de rendre ces preuves non interactives, notamment grâce à la transformation de Fiat-Shamir [15]. On parle alors de systèmes de *preuve à divulgation nulle de connaissance non interactifs* (NIZKP, Non-Interactive Zero-Knowledge Proofs) [16].

#### Transformation de Fiat-Shamir

Cette technique très utilisée permet de transformer une preuve interactive en preuve non interactive (NIZKP). Pour ce faire, le challenge est choisi comme le hachage cryptographique de l'énoncé de la proposition. La fonction de hachage sert d'oracle aléatoire et remplace alors le vérificateur dans le choix du défi.

#### Signatures

Une application directe des NIZKP est les schémas de signature, où ils permettent de garantir l'authenticité et l'intégrité d'un message sans révéler d'informations supplémentaires sur le signataire. Nous étudierons par exemple les signatures de Schnorr.

Le concept de signature, étant à la lumière des NIZKP qu'une *preuve de connaissance d'un secret attestant une donnée*, peut être généralisé à des schémas élargis appelés Signatures-of-Knowledge [13].

#### Preuves à destination d'un vérificateur spécifique

Les *preuves à divulgation nulle de connaissance non interactives* (NIZKP) sont par défaut non-répudiables, à l'exception d'un type spécifique : les *Designated-Verifier Zero Knowledge Proofs* (DVZKP) [17]. Ce type de preuve permet de démontrer soit une proposition, soit la connaissance de la clé secrète du vérificateur. Seul le vérificateur, en tant que détenteur exclusif de sa clé secrète, peut alors valider la proposition. Par ailleurs, le prouveur peut toujours nier l'authenticité de la preuve en affirmant qu'elle a été construite par le vérificateur lui-même.

### 2.5.4 Σ-protocols

Ils se caractérisent par un échange en trois étapes faisant penser à la forme de la lettre grecque Σ.

Le prouveur transmet un engagement au vérificateur, qui répond par un défi. Le prouveur fournit alors une réponse. Si celle-ci est correcte, il convainc le vérificateur qu'il connaît le secret.

$\mathbf{P} \rightarrow \mathbf{V}$ : engagement (commitment)
$\mathbf{V} \rightarrow \mathbf{P}$ : défi (challenge)
$\mathbf{P} \rightarrow \mathbf{V}$ : réponse (response)

On peut transformer cette preuve en une version non-interactive en utilisant la *transformation de Fiat-Shamir*. Cela consiste à choisir le défi comme étant le résultat

du hachage de l'énoncé et de l'engagement. Ainsi, le prouveur génère lui-même les trois étapes du protocole, sans toutefois avoir le contrôle sur le défi.

Pour un exemple de preuves de sécurité pour les *protocoles Sigma*, voir [18].

### 2.5.5 zk-SNARK

Les **zk-SNARK** [19] sont des systèmes de preuve à divulgation nulle de connaissance dit "généraux" car ils peuvent servir à prouver n'importe quel prédicat en dévoilant seulement une partie des données utilisées pour le prouver.

#### Propriétés

**Zero-Knowledge** : Ne dévoile pas d'autre information que la validité de la proposition.

**Succinct** : Les preuves sont très courtes et de taille fixe.

**Non-interactive** : Le système ne nécessite qu'un seul échange du prouveur au vérificateur.

**ARgument of Knowledge** : En général, ces systèmes sont toujours complets. En revanche, certains systèmes ne sont pas parfaitement consistants, mais computationnellement consistants. Autrement dit, il n'est théoriquement pas impossible, mais en pratique les probabilités de réussir à forger une preuve valide sans connaissance d'un témoin valide sont *négligeables*. Cette technique est classique en cryptographie. Dans ce cas, on ne parle plus de *preuve de connaissance* (proof of knowledge), mais d'*argument de connaissance*.

#### Fonctionnement

Il faut définir sa proposition sous forme de circuit arithmétique. Ensuite, une cérémonie de confiance (trusted setup) doit avoir lieu pour créer tous les paramètres nécessaires à la preuve. Puis, il est possible de générer des preuves concises de ces propositions. Ces preuves sont généralement coûteuses à générer, mais rapides à vérifier.

### 2.5.6 Autres systèmes de preuves à divulgation nulle de connaissance

D'autres systèmes de preuves à divulgation nulle de connaissance existent, notamment STARK et Halo2 qui ne nécessitent pas de cérémonie de confiance. STARK est également résistant à l'algorithme de Shor.

## 2.6 Protocoles cryptographiques

Un protocole définit les règles de communication entre les participants afin d'atteindre un objectif. Un protocole cryptographique emploie, comme son nom l'indique, les primitives cryptographiques vues précédemment.

On peu décrire ces protocoles par les messages qui sont envoyés (c'est la notation "Alice et Bob"). Les messages étant décrits en utilisant une algèbre de termes afin de pouvoir prendre en compte les différentes variables en jeu (comme les clés et les *nonces*).

Lorsqu'on décrit un protocole, on omet généralement les vérifications que doivent faire les participants afin de rester succinct. Ces vérifications sont nécessaires mais généralement laissées implicites.

Voici un exemple de protocole trivial décrivant l'envoi par Alice à Bob d'un message $m$, et n'utilisant aucune primitive cryptographique :

$$\mathbf{A} \to \mathbf{B} : m$$

Or, l'intérêt des protocoles cryptographiques bien conçus est de garantir les propriétés de sécurité (authenticité, intégrité, confidentialité) même en présence d'un attaquant puissant. On suppose généralement un attaquant (aussi appelé adversaire, intrus) ayant un contrôle total sur le réseau, c'est-à-dire pouvant intercepter, envoyer ou modifier les messages passant sur le réseau, comme dans le modèle de Dolev-Yao [20]. On appelle également ce genre d'attaque *Machine-in-the-middle* (MITM). Elles sont, en pratique, assez courantes.

Un intrus C pourrait donc se faire passer pour Bob auprès d'Alice et pour Alice auprès de Bob afin d'intercepter et de contrôler leurs communications.

$$\mathbf{A} \rightarrow \mathbf{C(B)} : m$$

$$\mathbf{C(A)} \rightarrow \mathbf{B} : m'$$

Il est possible de se prémunir contre ce genre d'attaques. Par exemple, Alice peut signer son message, garantissant ainsi son intégrité et son authenticité.

$$\mathbf{A} \rightarrow \mathbf{B} : m, \mathrm{sign}_{K_A}(m)$$

Si Alice et Bob partagent une clé secrète $K_{AB}$, Alice peut chiffrer son message en utilisant un mode de chiffrement authentifié (AE), garantissant la confidentialité, l'intégrité et l'authenticité.

$$\mathbf{A} \rightarrow \mathbf{B} : \mathrm{enc}_{K_{AB}}(m)$$

Dans ces deux derniers cas, l'intrus ne peut plus se faire passer pour Alice.

En revanche, s'il est en position de MITM, il pourra rejouer le message. Il peut également intercepter et ne pas renvoyer les messages.

Nous allons maintenant voir un protocole un peu plus complexe : le protocole de Needham-Schroeder [21]. Ce protocole permet à Alice et Bob de se mettre d'accord sur une clé de session même si le réseau n'est pas sûr.

$$\mathbf{A} \rightarrow \mathbf{S} : A, B, N_A$$

$$\mathbf{S} \rightarrow \mathbf{A} : \mathrm{enc}_{K_A}(N_A, B, K_{AB}, \mathrm{enc}_{K_B}(K_{AB}, A))$$

$$\mathbf{A} \rightarrow \mathbf{B} : \mathrm{enc}_{K_B}(K_{AB}, A)$$

$$\mathbf{B} \rightarrow \mathbf{A} : \mathrm{enc}_{K_{AB}}(N_B)$$

$$\mathbf{A} \rightarrow \mathbf{B} : \mathrm{enc}_{K_{AB}}(N_B - 1)$$

Même si ce protocole semble simple, il est difficile de prouver qu'il est sûr. En l'occurrence, on l'a cru sûr pendant plus de 15 ans avant qu'une faille ne soit découverte par Gavin Lowe [1].

Beaucoup de protocoles que nous utilisons tous les jours sont aussi, voire plus, complexes (comme TLS ou SSH). Afin d'assurer leur sécurité, il est important de les analyser rigoureusement et, lorsque c'est possible, de les prouver formellement.

## 2.7 Preuves de protocole cryptographique

Les preuves de protocoles cryptographiques permettent de démontrer certaines propriétés de sécurité, telles que la confidentialité de certaines informations.

Moins connues et plus récentes que les preuves de programme, elles sont plus complexes, et le problème n'est même pas forcément décidable (surtout si l'on prend en compte un nombre non borné de sessions). En pratique, on s'en sort quand même avec certains outils comme *ProVerif*, au prix du fait qu'il ne termine pas nécessairement.

De nombreux scénarios d'attaques (parfois des centaines) sont généralement à considérer. De plus, les protocoles impliquant plusieurs acteurs, il faut examiner tous les cas possibles (notamment lorsque tel acteur ou une collusion d'acteurs est corrompue). Les preuves sont longues et complexes. Elles sont donc largement automatisées.

Elles se sont généralement basées sur le modèle symbolique : les primitives cryptographiques sont idéalisées, et les protocoles sont modélisés à l'aide d'une algèbre de termes. On suppose un adversaire puissant, ayant un contrôle total sur le réseau, que l'on appelle adversaire de Dolev-Yao [20].

Avec les protocoles de vote électronique, on cherche également à prouver de nouvelles propriétés de vérifiabilité.

Les deux outils principaux pour faire des preuves de protocole dans le modèle symbolique sont ProVerif et Tamarin.

Il est aussi possible de faire des preuves dans un autre modèle dit “computationnel”, avec des outils comme EasyCrypt ou CryptoVerif.

# Chapitre 3

# Protocoles de vote par internet

## 3.1 Introduction

Le vote par internet doit assurer la confidentialité (le **secret du vote**), pour lequel on va pouvoir appliquer des techniques classiques de la cryptographie.

Il doit également répondre à un besoin de transparence. La réponse des cryptographe a été de développer la notion de **vérifiabilité de bout-en-bout** (*End-to-end Verifiability* (E2E-V)), comparable à la notion de chiffrement de bout-en-bout (*End-to-end Encryption* (E2E-E)) dans les protocoles de messagerie électronique.

Le besoin de concilier confidentialité et vérifiabilité, deux aspects qui peuvent sembler contradictoire, sont au coeur des protocoles de vote par internet. Lorsqu'on ne veut pas qu'une seule entité soit garante du secret du vote (ou de la vérifiabilité), on essaye de partager la responsabilité entre plusieurs entités, il suffit alors qu'une petite partie des acteurs soient honnête pour les garantir.

En raison de la nature à distance de ces protocoles, de nouveaux dangers apparaissent également : le risque de **vote sous influence** et le risque de permettre la **vente**

**de vote**. Ces risques sont d'autant plus importants que l'enjeu du scrutin est grand. Des méchanismes dits de **resistance à l'influce** sont parfois nécessaires, tels que permettre de pouvoir mentir sur son vote, ou de pouvoir revoter secrètement. En revanche ils sont difficiles à mettre en place, alourdissent souvent beaucoup le protocole. Aussi, il n'existent que dans certains protocoles.

## 3.2 Secret du vote

Dans la majorité des cas, le vote doit rester secret, afin de permettre au votant de s'exprimer comme il le désire. C'est inscrit dans le code électoral pour les élections politiques, dans le code du travail pour les élections professionnelles. C'est aussi généralement le cas dans les associations, selon leur statuts.

Voici les principaux mécanismes permettant le *secret du vote* :

### 3.2.1 Chiffrement du vote

Dans les systèmes de vote électronique qui garantissent le secret du vote, les votes sont très souvent chiffrés par le votant vers le comité électoral via un système de chiffrement à clé publique. Souvent, on choisit ElGamal dont les propriétés (comme la malléabilité) va permettre de pouvoir anonymiser les bulletins de vote avant de les déchiffrer, soit via l'agrégation homomorphe, soit les mixnets, présentés plus loin.

### 3.2.2 Partage du secret par un comité électoral

Le comité électoral est composé de plusieurs personnes qui doivent se réunir pour déchiffrer le résultat de l'élection. Il est important qu'ils le fassent après que les votes aient étés anonymisés.

Le comité électoral est composé de plusieurs membres appelés *trustees* ou *gardiens*, qui détiennent chacun une partie de la clé de l'élection, via un système de partage de secret (voir [**?**] par exemple).

Il est possible de définir un quorum de *trustess* qui doivent être présent pour déchiffrer le résultat (3 parmi 5, 5 parmi 7, ...).

Les différentes parts peuvent soit êtres crées de manière centralisé par une entité qui est chargée de les distribuer au différents *trustees* (puis doit les oublier), soit, encore mieux, de manière décentralisés, via un protocole permettant de construire la clé de l'élection sans que tous les secrets n'ai jamais besoin d'être tous réunis [22, 23].

### 3.2.3 Agrégation homomorphe

Les votes chiffrées sont agrégés en un résultat chiffré (voir section sur ElGamal).

Cette méthode est vérifiable car chacun peut la reproduire à l'aide des bulletins de vote chiffrés et vérifier le résultat.

C'est une des deux méthodes d'anonymisation des données (avec l'agrégation homomorphe).

### 3.2.4 Mixnets

Les mélanges vérifiables (vérifiable mixnets) [24] permettent de faire un mélange de données chiffrés (en l'occurrence via ElGamal), en prouvant, à l'aide de ZKP, que les résultat correspond bien à un ordonnancement re-randomisé des données d'origine, sans en dévoiler l'ordre, et sans avoir jamais besoin de les déchiffrer.

Cette étape est vérifiable car elle est accompagnée de ZKP.

C'est une des deux méthodes d'anonymisation des données (avec l'agrégation homomorphe).

### 3.2.5 Déchiffrement vérifiable

Le comité électoral intervient lors du dépouillement de l'élection afin de déchiffrer le résultat (**après anonymisation**). Ils n'ont pas besoin d'intervenir simultanément, chaque *trustee* peut émettre à son tour (ou simultanément) un déchiffrement partiel. En assemblant le nombre requis de déchiffrement partiels, on retrouve le résultat.

## 3.3 Vérifiabilité

La notion de vérifiabilité de bout-en-bout (E2E-V) a été développée spécifiquement pour le vote, et est comparable à la notion de chiffrement de bout-en-bout (E2E-E) dans les protocoles de messagerie électronique.

**End-to-End Verifiability (E2E-V)** : Le résultat correspond :

— Aux votes des votants qui ont vérifié.
— À un sous-ensemble des votes des votants qui n'ont pas vérifié (leur vote a pu être droppé).
— Un votant (malhonnête) ne peut pas voter plus d'une fois.

Comme cette notion est complexe, on peut la diviser en sous-propriétés et démontrer les sous-propriétés individuellements.

**Vérifiabilité individuelle** : Le votant peut vérifier que son intention de vote est bien prise en compte. Elle se divise encore en :

— **cast-as-intented** : Je peux m'assurer mon bulletin de vote contient bien mon intention de vote.
— **recorded-as-cast** : Je peux m'assurer que mon bulletin de vote est enregistré correctement. On peut par exemple utiliser un *Bulletin Board* public.

**Vérifiabilité universelle** (**tallied-as-recorded**) : L'urne est vérifiable : Je peux m'assurer que le résultat proclamé correspond bien à l'expression des bulletins présents dans l'urne. On utilise une technique cryptographique pour anonymiser les suffrages, soit les mélanges vérifiables, soit l'agrégation homomorphe. On procède enfin à un déchiffrement vérifiable.

*Remarque.* Le terme **universel** peut porter à confusion car il n'est pas toujours possible de faire cette vérification uniquement à partir des donnée publique (seul les auditeurs le peuvent), notamment en raison de risque pour la confidentialité long-terme (everlasting privacy). On peut parler de **vérifiabilité de l'urne** pour rester général.

**Verifiabilité de l'éligibilité** : L'urne ne continer que les bulletins émis par des votants légitimes, et au plus un par votant. L'élibilité est définie par la *liste électorale* qui n'est pas publique mais est parfois être accessible sur demande.

La combinaison des trois propriétés de **cast-as-intended**, **recorded-as-cast** et **tallied-as-recorded** donne **tallied-as-cast** (dépouillé comme voté), ce qui, combiné à la vérifiabilité de l'éligibilité correspond à la vérifiabilité de bout en bout (E2E-V).

## 3.4 Résistance à l'influence

Un influenceur (aussi appelé coerciteur) ne peux forcer le votant à voter d'une certaine manière. Comme cette propriété est difficile à formaliser, il existe la propriété affaiblie suivante :

**receipt-freeness** : il est impossibilité pour le votant de générer un reçu permettant de prouver son vote.

La notion plus forte de **résistance à l'influence** typiquement requiert le revote, permettant à un votant qui fut sous le contrôle de l'influenceur au moment du vote de changer son vote.

## 3.5 Protocoles

### 3.5.1 Helios

Helios [25] propose un protocole assurant à la fois le secret du vote et la vérifiabilité de bout en bout (E2E-V). Il est inspiré des travaux de Cramer, Gennaro, Schoenmakers [26] et Benaloh [27, 28].

Dans Helios, les votants chiffrent leur vote, garantissant le secret du vote. Les votes sont ensuite anonymisés puis déchiffrés. Helios est pleinement vérifiable, car quiconque le désire peut vérifier chaque étape. C'est un modèle dit d'*audit ouvert* (open-audit).

Helios n'offre pas de mécanisme de résistance à l'influence : une personne avertie peut extraire les nombres aléatoires utilisés pour chiffrer les votes et ainsi fournir un reçu permettant de prouver son vote. Il existe d'ailleurs un bouton permettant de le faire automatiquement afin de sensibiliser les votants au problème.

#### Mise en place

Une clé de chiffrement (ElGamal) est générée par l'administrateur et partagée aux votants. Il est suggéré que cette clé soit partagée entre plusieurs "trustees". Pour des raisons de simplicité, Helios est présenté avec un seul trustee, le serveur.

#### Phase de vote

Les votants chiffrent leur vote via le code JavaScript fourni par Helios.

Avant d'envoyer leur vote, ils peuvent, s'ils le désirent, mettre au défi la plate-forme de prouver qu'elle a bien chiffré le bon choix, en lui demandant de révéler les nombres aléatoires utilisés pour chiffrer le vote, d'une manière analogue au Benaloh Challenge [28].

Lorsqu'ils ont confiance, ils peuvent valider leur vote en s'authentifiant auprès de la plate-forme.

#### Dépouillement

Les "trustees" mélangent les votes en utilisant des mixnets vérifiables (appelés Sako-Kilian mixnets [24]). Ils re-randomisent les votes tout en les réordonnant.

Les votes anonymisés sont enfin déchiffrés. Une preuve de déchiffrement prouve qu'il est correct. Les votes déchiffrés sont ensuite comptabilisés librement pour donner le résultat.

### 3.5.2 Belenios

Belenios [29] est une version formellement vérifiée d'Helios (Helios-C). Belenios apporte et formalise également la vérifiabilité de l'éligibilité.

Belenios offre également une plate-forme de vote depuis 2015 et est utilisé dans divers contextes, notamment pour des élections au sein d'universités ou d'associations. Plus de 1 000 élections y ont été organisées en 2020 et 2021, pour un total de plus de 100 000 bulletins reçus [30].

La spécification du protocole est détaillée dans un autre document [31]. Elle requiert d'ailleurs que tous les événements soient ajoutés à une base de données (Bulletin Board) de type "append-only log", c'est-à-dire qu'il est possible d'ajouter des événements, mais jamais d'en enlever, ce qui permet à des auditeurs de surveiller en temps réel son évolution et doter le protocole de capacités théoriques de décentralisation.

Comme Helios, il est bien indiqué dans les situations où l'on veut garantir le secret du vote et la vérifiabilité, mais où l'on ne considère pas la coercition comme un grand danger. Le revote est possible, ce qui offre un mécanisme basique de résistance à la coercition, mais ce mécanisme reste limité. Si l'on s'inquiète de risques de coercition, il vaudra mieux utiliser BeleniosRF, BeleniosVS ou Civitas.

### 3.5.3 I-voting

I-voting est un système de vote par internet mis en place à l'échelle nationale en Estonie depuis 2005. En 2011, 140 000 électeurs, soit 24,3 % des votants, l'ont utilisé pour l'élection du parlement estonien.

Son fonctionnement est inspiré d'Helios. Les citoyens sont en possession d'une carte d'identité numérique qui fait office de *Public key infrastructure* (PKI). L'existence de cette PKI évite de devoir envoyer des codes de vote tout en permettant la vérifiabilité de l'éligibilité. Les citoyens estonien peuvent voter depuis chez eux en utilisant leurs appareils personnels. I-voting ne remplace pas le vote aux urnes (qu'ils appellent p-voting), mais une manière alternative de participer.

Il est possible de vérifier son vote avec un second appareil (vérifiabilité individuelle).

Seul les condensats des ballots sont publiés (comme ça même si le chiffrement est cassé un jour, on n'aura pas l'historique des votes de chaque votant). Les observateurs (parti politiques, ...) peuvent vérifier la vérifiabilité universelle lors de la phase de dépouillement (tally).

### 3.5.4 CHVote

CHVote [32] est un protocole de vote électronique suisse développé par le canton de Genève. Le projet, complètement open-source (AGPL 3.0), est composé de plusieurs

briques logicielles bien documentées. Un généreux programme de bug bounty existe afin de traquer les éventuels bugs.

CHVote est vérifiable, comme l'exigent les exigences de sécurité fixées par la Confédération suisse. La vérifiabilité individuelle est assurée à l'aide de codes de vérification.

### 3.5.5 Swiss Post

Ce protocole a été initialement développé par Scytl en collaboration avec la Poste suisse (Swiss Post) en 2015. Cependant, plusieurs vulnérabilités ont été découvertes en 2019 [**?**, 33], ce qui a conduit à son interruption afin de corriger ces failles. Après modifications, il a été réintroduit en 2023.

Swiss Post est vérifiable. La vérifiabilité individuelle est également assurée par un système de codes de vérification.

### 3.5.6 Protocole de Neuchâtel

Le protocole de Neuchâtel est un autre système de vote électronique suisse, développé dans le canton de Neuchâtel. Il a été utilisé pour la première fois en 2013. Comme les deux autres protocoles suisses, il est vérifiable, notamment grâce aux codes de vérification.

Une analyse formelle du protocole de vote électronique de Neuchâtel [34] démontre que le secret du vote est préservé même si le serveur est compromis, et que la vérifiabilité individuelle est assurée même si l'appareil de vote est compromis (mais pas le serveur).

### 3.5.7 BeleniosRF

BeleniosRF [35] est une variante de Belenios apportant un mécanisme de *receipt-freeness*, c'est-à-dire que le votant sera incapable de prouver son vote à un influenceur, ce qui permet une très bonne résistance à l'influence.

BeleniosRF est basé sur un service de randomisation (assuré par exemple par le serveur de vote). Cela permet que les nombres aléatoires utilisés lors du chiffrement

du vote ne soient pas connus du votant. En revanche, le votant peut s'assurer que son vote a seulement été re-randomisé et non modifié grâce à une primitive récente appelée *Signature on randomizable ciphertexts* (SRC) [36] (une implémentation corrigée de cette primitive est donnée). Les SRC permettent au service de randomisation de régénérer une signature valide sur un chiffré re-randomisé (mais pas sur autre chose), ce qui permet d'assurer à la fois la vérifiabilité (via la signature) et la *receipt-freeness*.

Au passage, BeleniosRF en profite pour formaliser, dans la théorie des jeux, une propriété dite *strong receipt-freeness (sRF)*, où le votant ne peut pas prouver son vote même s'il est malicieux et contrôle l'appareil de vote.

### 3.5.8 BeleniosVS

BeleniosVS [37] se base sur BeleniosRF, mais utilise des feuilles de vote (voir figure 3.1) afin de garantir la vérifiabilité et la confidentialité, même si l'appareil de vote est malhonnête. Dans ce protocole, l'appareil de vote ne génère plus le vote, mais se contente de le re-randomiser, grâce à une SRC. Il n'a donc jamais connaissance du vote. Le serveur re-randomise également le vote, ce qui offre une garantie supplémentaire en cas d'appareil de vote malicieux.

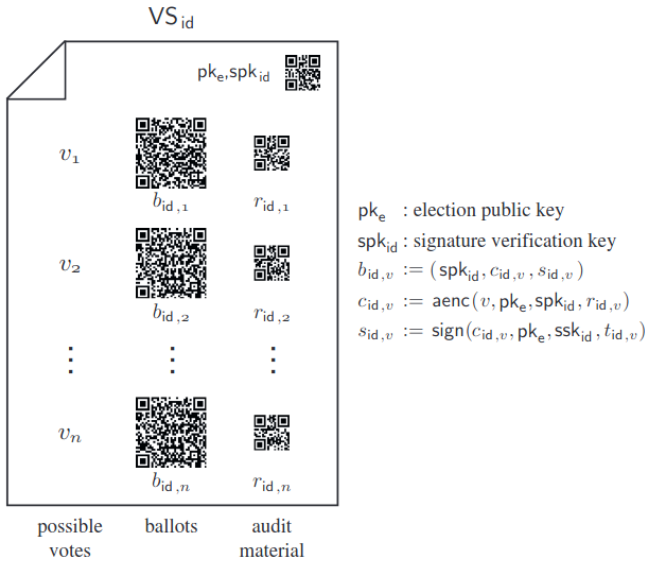


FIGURE 3.1 – A Voting Sheet

La vérifiabilité est déléguée à un autre appareil, qui va vérifier que la feuille de vote encode bien les votes pour les bons candidats.

### 3.5.9 BeleniosCaI

BeleniosCaI [38] permet d'assurer la vérifiabilité individuelle (CaI signifie *cast-as-intended*) lorsqu'on ne contrôle pas (complètement) l'appareil de vote, comme lorsque l'on utilise des logiciels propriétaires ou des machines propriétaires telles que la plupart des machines à voter actuelles.

BeleniosCaI est basé sur un code de retour fourni par l'appareil de vote, qui est lié à la valeur du vote. Il est ensuite possible de vérifier son vote en utilisant à la fois le code de retour et les données enregistrées sur le serveur.

Ce mécanisme, comme le Benaloh Challenge auquel il est une alternative argumentativement plus praticable, fournit seulement une preuve probabiliste et non une preuve certaine que le vote n'a pas été manipulé (ici avec une probabilité d'une fois sur deux).

Il semble que pour mentir sur son vote auprès d'un influenceur, il faut mentir sur son code de retour.

### 3.5.10 Civitas et JCJ

Civitas [39], une continuation du protocole JCJ [40], est un système de vote par Internet conçu pour résister à la coercition.

JCJ et Civitas utilisent un système de "fake credentials" : le votant peut générer de faux credentials qui seront supprimés au comptage. Le votant peut donc mentir sur son credential et soit le donner à l'influenceur, qui aura l'impression de voter comme il le veut, soit faire semblant de voter pour un autre candidat ou de s'abstenir. Pour l'influenceur, il n'est pas possible de distinguer un vrai credential d'un faux.

Le votant pourra voter comme il le souhaite avec son vrai credential lorsqu'il n'est plus sous l'influence du coerciteur.

Civitas/JCJ offre également le secret du vote (les votes sont chiffrés), ainsi que les deux vérifiabilités (individuelle et universelle).

En revanche, Civitas est assez coûteux, la complexité est en $O(n^2)$ en fonction du nombre de votants, notamment pour l'élimination des faux credentials.

### 3.5.11 Selene

Dans Selene [41], le votant pourra vérifier son vote en clair dans le résultat proclamé via un numéro de suivi, donnant une excellente vérifiabilité individuelle. Le numéro de suivi n'est donné qu'au dépouillement, ce qui rend très difficile la vente de vote, le votant pouvant mentir sur son numéro de suivi. Cela donne une bonne résistance à la coercition.

### 3.5.12 sElect

sElect [42] est un système minimaliste qui offre une excellente vérifiabilité (y compris vis à vis d'un appareil de vote malveillant, un problème difficile pour les dérivés d'Helios). En revanche, le secret du vote ne tient que si le serveur est honnête.

## Chapitre 4

# Conclusion

Le vote par internet répond à un nouveau besoin. Sa plus grande praticité en fait une alternative au vote papier dans certaines occasions où l'enjeu reste modéré (associations, élections professionnelles, etc.).

Néanmoins, les protocoles de vote par internet sont relativement récents et se heurtent à des défis importants : la vérifiabilité ET la confidentialité, et les risques de vente de vote à grande échelle "maintenant avec preuve".

Le vote étant un méchanisme critique, indispensable à nos systèmes politiques basés sur la démocratie, l'adoption d'un système de vote électronique est très risqué. Il est n'est généralement pas autorisé pour les élections politiques en France. L'Union européenne, par exemple, émet des recommandations défavorables à son utilisation. En revanche, certains pays, comme l'Estonie, l'ont pleinement adopté, avec plus de 50 % des électeurs y ayant recours.

En revanche, il est pressenti comme une bonne alternative au vote par correspondance qui offre actuellement peu de sécurité et aucune vérifiabilité. En France, l'élection la plus importante dans laquelle il fut utilisable fut les législatives de 2022 pour les Français résidant à l'étranger, en complément des autres modes de vote (à l'urne ou

par correspondence).

Pour les élections non-politiques, Le vote électronique est de mieux en mieux défini et encadré, notamment avec les recommendations de la CNIL et le guide de l'ANSSI. Il est adopté par un nombre croissant de petites structures (associations, entreprises, etc.)

Une fois cette technologie mise en place, elle pourrait favoriser de nouvelles manières de voter, notamment avec de méthodes de comptage plus sophistiquées, telles que le jugement majoritaire, le vote par approbation ou la méthode de Condorcet. Le recours au vote serait également facilité, permettant de pouvoir y avoir recours à plus d'occasions.

Une grande variété de protocoles existent, certains mettant l'accent sur la simplicité et la vérifiabilité [27, 28, 25, 42, 38], tandis que d'autres se concentrent plutôt sur la résistance à la vente de vote [40, 41].

C'est donc un domaine ayant de beaux jours devant lui, à condition que des garanties de sécurité solides soient mises en place, à l'image de ce qui a été fait pour la messagerie instantanée avec des protocoles comme celui de Signal. Notamment, il va pouvoir profiter des développement dans les preuves formelles des protocoles pour lui éviter les nombreuses attaques que subissent les protocoles cryptographique avant le développement de cette dernière discipline.

# Références


[1] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using fdr," in *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 147–166, Springer, 1996. [Cité en pages 2 et 29]

[2] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949. [Cité en page 10]

[3] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *Selected Areas in Cryptography : 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*, pp. 1–24, Springer, 2001. [Cité en page 11]

[4] D. J. Bernstein, "The salsa20 family of stream ciphers," in *New stream cipher designs : the eSTREAM finalists*, pp. 84–97, Springer, 2008. [Cité en page 12]

[5] D. J. Bernstein *et al.*, "Chacha, a variant of salsa20," in *Workshop record of SASC*, vol. 8, pp. 3–5, Citeseer, 2008. [Cité en page 12]

[6] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 373–386, 1988. [Cité en page 13]

[7] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography : The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022. [Cité en page 14]

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978. [Cité en pages 14 et 15]

[9] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985. [Cité en page 14]

[10] P. W. Shor, "Algorithms for quantum computation : discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994. [Cité en page 16]

[11] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978. [Cité en page 16]

[12] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Advances in Cryptology—CRYPTO'98 : 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, pp. 13–25, Springer, 1998. [Cité en page 18]

[13] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *Advances in Cryptology-CRYPTO 2006 : 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*, pp. 78–96, Springer, 2006. [Cité en pages 21 et 25]

[14] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Providing sound foundations for cryptography : On the work of shafi goldwasser and silvio micali*, pp. 203–225, 2019. [Cité en page 24]

[15] A. Fiat and A. Shamir, "How to prove yourself : Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*, pp. 186–194, Springer, 1986. [Cité en page 24]

[16] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Advances in Cryptology—CRYPTO'87 : Proceedings 7*, pp. 52–72, Springer, 1988. [Cité en page 24]

[17] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 143–154, Springer, 1996. [Cité en page 25]

[18] P. Gaudry, "Some zk security proofs for belenios," 2017. [Cité en page 26]

[19] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology–EUROCRYPT 2016 : 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pp. 305–326, Springer, 2016. [Cité en page 26]

[20] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983. [Cité en pages 27 et 29]

[21] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978. [Cité en page 28]

[22] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Distributed elgamal á la pedersen : application to helios," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, pp. 131–142, 2013. [Cité en page 33]

[23] J. Benaloh, M. Naehrig, and O. Pereira, "Reactive : Rethinking effective approaches concerning trustees in verifiable elections," *Cryptology ePrint Archive*, 2024. [Cité en page 33]

[24] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme : A practical solution to the implementation of a voting booth," in *Advances in Cryptology—EUROCRYPT'95 : International Conference on the Theory and Application of Cryptographic Techniques Saint-Malo, France, May 21–25, 1995 Proceedings 14*, pp. 393–403, Springer, 1995. [Cité en pages 33 et 36]

[25] B. Adida, "Helios : Web-based open-audit voting.," in *USENIX security symposium*, vol. 17, pp. 335–348, 2008. [Cité en pages 35 et 44]

[26] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997. [Cité en page 35]

[27] J. D. C. Benaloh, *Verifiable secret-ballot elections*. Yale University, 1987. [Cité en pages 35 et 44]

[28] J. Benaloh, "Simple verifiable elections.," *EVT*, vol. 6, pp. 5–5, 2006. [Cité en pages 35, 36 et 44]

[29] V. Cortier, P. Gaudry, and S. Glondu, "Belenios : a simple private and verifiable electronic voting system," *Foundations of Security, Protocols, and Equational Reasoning : Essays Dedicated to Catherine A. Meadows*, pp. 214–238, 2019. [Cité en page 36]

[30] V. Cortier, P. Gaudry, and S. Glondu, "Features and usage of belenios in 2022," in *The International Conference for Electronic Voting (E-Vote-ID 2022)*, University of Tartu Press, 2022. [Cité en page 36]

[31] S. Glondu, "Belenios specification," *Version 0.1. http ://www. belenios. org/specification. pdf*, 2013. [Cité en page 37]

[32] R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis, "Chvote protocol specification," *Cryptology ePrint Archive*, 2017. [Cité en page 37]

[33] V. Cortier, A. Debant, and P. Gaudry, "A privacy attack on the swiss post e-voting system," in *RWC 2022-Real World Crypto Symposium*, 2022. [Cité en page 38]

[34] V. Cortier, D. Galindo, and M. Turuani, "A formal analysis of the neuchâtel e-voting protocol," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 430–442, IEEE, 2018. [Cité en page 38]

[35] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf : A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1614–1625, 2016. [Cité en page 38]

[36] O. Blazy, G. Fuchsbauer, D. Pointcheval, and D. Vergnaud, "Signatures on randomizable ciphertexts," in *Public Key Cryptography–PKC 2011 : 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings 14*, pp. 403–422, Springer, 2011. [Cité en page 39]

[37] V. Cortier, A. Filipiak, and J. Lallemand, "Beleniosvs : Secrecy and verifiability against a corrupted voting device," in *2019 IEEE 32nd computer security foundations symposium (CSF)*, pp. 367–36714, IEEE, 2019. [Cité en page 39]

[38] V. Cortier, A. Debant, P. Gaudry, and S. Glondu, "Belenios with cast as intended," in *International Conference on Financial Cryptography and Data Security*, pp. 3–18, Springer, 2023. [Cité en pages 40 et 44]

[39] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas : Toward a secure voting system," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 354–368, IEEE, 2008. [Cité en page 40]

[40] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pp. 61–70, 2005. [Cité en pages 40 et 44]

[41] P. Y. Ryan, P. B. Rønne, and V. Iovino, "Selene : Voting with transparent verifiability and coercion-mitigation," in *Financial Cryptography and Data Security : FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pp. 176–192, Springer, 2016. [Cité en pages 41 et 44]

[42] R. Küsters, J. Müller, E. Scapin, and T. Truderung, "select : a lightweight verifiable remote voting system," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pp. 341–354, IEEE, 2016. [Cité en pages 41 et 44]