Le pastillage dans le vote électronique

Maxime Lalisse
Université de Lille - Stage de Master

Sous la supervision de Véronique Cortier, Alexandre Debant et Lucca Hirschi
LORIA, équipe PESTO, CNRS, Inria, Université de Lorraine

Mars 2025 – Août 2025

Table des matières

1 Introduction

Le vote est un outil fondamental qui permet à un ensemble d'électeurs de prendre une décision ou d'exprimer une opinion. Il est employé dans de nombreux contextes (politique, associations, entreprises, etc.), avec des enjeux plus ou moins élevés. Les modalités de vote sont elles aussi variées : à main levée, à l'urne, par internet, par correspondance, etc. Nous nous intéressons ici au **vote par internet**, c'est-à-dire lorsque l'électeur peut voter à distance à l'aide de son propre appareil (ordinateur, smartphone, etc.). Plusieurs systèmes ont déjà été déployés ou étudiés en pratique, tels qu'Helios [1], Belenios [10] ou Civitas [7]. Le vote par internet est utilisé dans plusieurs pays, notamment en France [11], en Suisse [17] et en Estonie [15]. Il est aussi largement utilisé pour les élections non politiques, en particulier dans les associations et les entreprises. En France, les élections non politiques sont encadrées par la CNIL et l'ANSSI, qui publient des recommandations en matière de sécurité [2, 8].

Nous chercherons à satisfaire des propriétés de **confidentialité** [12] :

Secret du vote Le vote de chaque votant doit rester secret. Seul le résultat de l'élection est observable.

Secret de la participation Préserver le secret de la participation ou non d'un électeur.

Confidentialité à long terme Assurer la pérennité du secret du vote même si la cryptographie dont le chiffrement venait à être cassé dans le futur. Cette propriété est rarement satisfaite.

Nous chercherons également à satisfaire des propriétés de **vérifiabilité** [6, 9] :

Vérifiabilité individuelle Chaque électeur doit pouvoir vérifier que son vote a bien été pris en compte dans le décompte final.

Vérifiabilité universelle Tout observateur peut vérifier que le résultat publié correspond bien aux bulletins présents dans l'urne.

Vérifiabilité de l'éligibilité Il est possible de vérifier que seuls les électeurs autorisés ont pu voter.

Ce document explore les protocoles de vote avec **pastillage**. Avec le pastillage, chaque électeur possède des attributs appelés "pastilles" (par exemple l'âge, la ville ou la profession). Le pastillage permet de mener une élection principale (appelée **scrutin direct**) tout en obtenant en même temps des résultats pour certains sous-groupes de votants, selon leurs attributs (**scrutins indirects**). Seuls les résultats des scrutins indirects prévus dans la configuration peuvent être obtenus. Le pastillage est très utilisé dans les élections professionnelles, par exemple pour élire les représentants au niveau global de l'établissement (scrutin direct), tout en produisant des résultats distincts par collège électoral (ouvriers/employés, cadres, etc.) ou par site géographique.

Le pastillage amène des risques pour la **vérifiabilité**. Le votant doit pouvoir vérifier que son vote a bien été pris en compte dans chaque scrutin indirect. Les observateurs doivent pouvoir vérifier que les électeurs ne participent qu'aux scrutins indirects autorisés par leurs attributs. Le pastillage amène également des risques pour le **secret du vote**. Certaines attaques permettent de désanonymiser complètement un votant ou de réduire considérablement son ensemble d'anonymat (*anonymity set*), comme par exemple si un scrutin indirect contient un très petit nombre de participants.

L'ANSSI évoque deux mises en œuvre du pastillage [2]. La première consiste à chiffrer les pastilles avec le vote. Cela comporte des risques importants pour le secret du vote, surtout pour les participants ayant une combinaison de pastilles rare, voire unique. La vérifiabilité est aussi menacée car les votants peuvent mentir sur leurs pastilles et donc participer aux scrutins indirects qu'ils veulent. Pour ces raisons, elle est réservée aux élections à très faible enjeu. La seconde mise en œuvre consiste à envoyer les pastilles en clair à côté du bulletin et à faire autant d'urnes que de scrutins indirects. Nous appelons ces deux protocoles Protocole In et Protocole Side et les étudions section 3.

La **vérification formelle** permet de s'assurer qu'un protocole de vote respecte bien les propriétés de sécurité attendues. L'histoire a montré que des attaques pouvaient être découvertes plusieurs années après la conception d'un protocole [16, 3]. La vérification formelle permet de prévenir ces situations. Nous modélisons nos protocoles et propriétés de sécurité dans ProVerif [4], afin de trouver toutes les attaques possibles, et prouver l'absence d'attaques le cas échéant.

Contributions

Contexte Nous commençons par définir les concepts du pastillage et présentons deux premières mises en œuvre. Cela nous permet d'identifier de nouvelles attaques. Cela nous permet également de proposer une adaptation des propriétés de sécurité.

Mises en œuvre du pastillage Nous adaptons les propriétés de sécurité adaptées au pastillage, puis nous présentons quatre protocoles de vote électronique avec pastillage. **Protocole In** consiste à chiffrer les pastilles avec le vote. **Protocole Side** consiste à envoyer les pastilles en clair à côté du vote. **Protocole Urnes** consiste à envoyer son bulletin ainsi que les urnes auxquelles on participe. **Protocole ZKP** consiste à envoyer son bulletin ainsi que des preuves zero-knowledge d'éligibilité. Pour chaque protocole, on donne ses propriétés de sécurité, limitations et vulnérabilités.

Vérification formelle Nous proposons des modèles formels des quatre protocoles présentés plus haut ainsi qu'une formalisation des propriétés de sécurité adaptées au pastillage avec l'outil ProVerif. Enfin, nous présentons nos résultats.

2 Contexte

Un protocole de vote par internet comporte plusieurs participants : Les **Votants (Voters)**, le **Bureau de vote (Trustees)** garant du *secret du vote*, un **Serveur de vote (VS)** et éventuellement un **Registre (Reg)** garant du *droit de vote* et du secret de la liste électorale (et pastilles associées). On suppose également une base de données publique appelée **Bulletin Board (BB)** permettant la transparence et la vérification.

Pastillages. Soit T un ensemble fini de $n_T = |T|$ types de pastille, $\mathbb{A}_t$ l'ensemble fini des valeurs admissibles pour le type de pastille $t \in T$, et $\mathbb{P} = \prod_{t \in T} \mathbb{A}_t$ l'univers de toutes les combinaisons de pastilles possibles (les **pastillages**). Soit V l'ensemble des votants et $id \in V$ un votant. On note $\mathsf{attrs}(id) \in \mathbb{P}$ le pastillage du votant id.

Exemple. Si $T = \{\text{age, profession}\}$ avec $\mathbb{A}_{\text{age}} = \{\text{-18, 18-30, 31-65}\}$ et $\mathbb{A}_{\text{profession}} = \{\text{Docteur, Enseignant, Ingénieur}\}$, alors l'univers des pastillages est $\mathbb{P} = \{\text{-18, 18-30, 31-65}\} \times \{\text{Docteur, Enseignant, Ingénieur}\}$ et le pastillage de Sheldon est $\mathsf{attrs}(\text{Sheldon}) = (\text{-18, Docteur})$.

Scrutins indirects. Soit $\mathbf{BB_{main}}$ l'urne du scrutin direct. On considère $n_{bb} \in \mathbb{N}$ urnes $\mathbf{BB}_1, \ldots, \mathbf{BB}_{n_{bb}}$, chacune associée à un scrutin indirect et vue comme une sous-urne de $\mathbf{BB_{main}}$. À toute urne BB_u est associé un *langage d'éligibilité* $\boldsymbol{\mathcal{E}_u} \subseteq \mathbb{P}$ (les pastillages autorisés).

On dit que $\mathcal{E}_u$ est *factorisable* lorsqu'il existe, pour chaque type $t \in T$, un sous-ensemble $\mathbb{A}_{t,u} \subseteq \mathbb{A}_t$ tel que $\mathcal{E}_u = \prod_{t \in T} \mathbb{A}_{t,u}$. Dans ce cas, les contraintes peuvent être vérifiées séparément pour chaque type d'attribut.

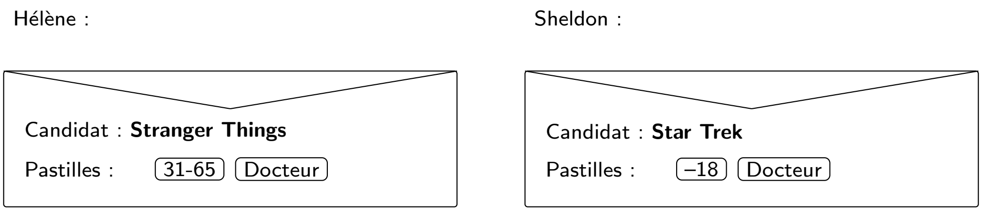
Exemple. L'urne $\mathbf{BB}_a$ est destinée aux docteurs et ingénieurs âgés de 18 à 65 ans ($\boldsymbol{\mathcal{E}_a} = \{\text{18-30, 31-65}\} \times \{\text{Docteur, Ingénieur}\}$, factorisable). Julien n'est pas éligible avec $\mathsf{attrs}(\text{Julien}) = (\text{18-30, Enseignant}) \notin \boldsymbol{\mathcal{E}_a}$, tandis qu'Hélène est éligible avec $\mathsf{attrs}(\text{Hélène}) = (\text{31-65, Docteur}) \in \boldsymbol{\mathcal{E}_a}$.

Cryptographie. Nous utiliserons des primitives cryptographiques classiques : un algorithme de chiffrement asymétrique ($\text{enc}(pk, m, r)$, $\text{dec}(sk, c)$), un schéma de signature ($\text{sign}(sk, m)$, $\text{verify}(vk, m, s)$). Pour les protocoles les plus avancés, on suppose un chiffrement rerandomisable ($\text{rerand}(pk, c, r')$), un schéma d'engagements ($\text{com}(x)$, $\text{open}(c, x, r)$) et de preuves à divulgation nulle de connaissance ($\text{prove}(R, x, w)$, $\text{verify\_zkp}(R, x, \pi)$). On utilise un chiffrement homomorphe. Nous utilisons également un chiffrement homomorphe, qui permet l'**accumulation** des chiffrés, c'est-à-dire leur combinaison pour obtenir le chiffré de la somme des messages. Enfin, à des fins d'anonymisation, nous aurons parfois recours aux **mélanges vérifiables**, qui permettent de réaliser une permutation secrète des chiffrés tout en restant vérifiable.

2.1 Premières approches

2.1.1 Chiffrer les pastilles avec le vote

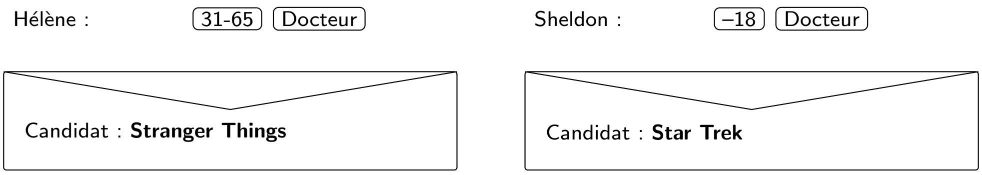
On peut envisager de simplement chiffrer les pastilles avec le vote. Cela ne demande généralement qu'une adaptation minimale du protocole, si celui-ci utilise déjà les mélanges vérifiables.



Au dépouillement, chaque pastillage associé à chaque vote est rendu public. Cela est pratique afin de pouvoir construire a posteriori les multiples scrutins indirects. Or certains électeurs ont des pastilles rares, voire uniques. Par exemple Sheldon est le seul à être docteur de moins de 18 ans. Il est donc trivial de connaître son vote (**attaque par rareté**). Le votant peut également utiliser le pastillage pour rendre son vote identifiable afin de pouvoir le vendre (**attaque à l'italienne**), ou des pastilles malicieuses peuvent être envoyées à un votant peu attentif afin de pouvoir désanonymiser son vote (**attaque par pastilles malicieuses**).

2.1.2 Envoyer les pastilles à côté du vote

Une autre approche serait d'envoyer les pastilles en clair à côté du vote chiffré. Cela permet de créer autant d'urnes qu'il y a de scrutins indirects. C'est compatible à la fois avec l'accumulation homomorphe et les mélanges vérifiables.



On procède à un dépouillement pour chaque scrutin indirect. Si l'on suppose un scrutin indirect par profession, le vote de Sheldon sera maintenant mélangé avec ceux de tous les autres docteurs. Il en va de même avec un scrutin indirect par catégorie d'âge. En revanche, si une urne est trop spécifique, comme ce serait le cas avec une urne pour les docteurs de moins de 18 ans dont Sheldon est le seul individu, alors le vote de Sheldon pourra être connu (**attaque par urne restreinte**). Parfois, c'est l'intersection des urnes qui permet de désanonymiser un vote, comme dans le cas où il existerait une urne pour les étudiants de moins de 18 ans, et une autre pour tous les étudiants et les docteurs de moins de 18 ans. On pourrait alors facilement connaître le vote de Sheldon en soustrayant les résultats (**attaque par intersection d'urnes**). Un autre cas à considérer est celui où des pastilles malicieuses sont envoyées à un votant qui ne les vérifie pas. Dans ce cas, son vote atterrira dans les urnes choisies par l'attaquant (**attaque par déplacement**). La **confidentialité à long terme** est également compromise si le pastillage permet d'identifier un votant.

2.2 Attaques

L'utilisation du pastillage ouvre la voie à de nouvelles attaques qui compromettent la confidentialité ou la vérifiabilité du scrutin.

Attaque par rareté. Dans les protocoles où les pastilles sont chiffrées avec le vote (Protocole In), le vote des électeurs possédant des pastilles rares, voire uniques, peut être facilement identifié. Cette situation est d'autant plus dangereuse que l'*anonymity set* associé à une combinaison de pastilles est petit.

Attaque par pastilles malicieuses. Dans les protocoles où les pastilles sont chiffrées avec le vote (Protocole In), un adversaire peut tenter d'attribuer à un électeur des pastilles construites de manière à rendre son bulletin identifiable. Un électeur peu attentif peut ainsi être piégé, ce qui permet à l'attaquant de désanonymiser son vote. Cela peut également servir à renverser un scrutin indirect.

Attaque à l'italienne. Dans les protocoles où les pastilles sont chiffrées avec le vote (Protocole In), un électeur peut utiliser ses pastilles de manière à rendre son bulletin reconnaissable. Il peut ainsi fournir une preuve de son vote et le monnayer, facilitant l'achat de voix ou la coercition électorale. Cela peut également servir à renverser un scrutin indirect.

Attaque par déplacement. Dans les protocoles où les pastilles ne sont pas chiffrées avec le vote, en forçant l'attribution de pastilles particulières, un attaquant peut contraindre le bulletin d'un électeur à rejoindre une urne choisie à l'avance. Cette stratégie permet soit de réduire l'*anonymity set* de la victime, soit d'influencer (voire renverser) le résultat d'un scrutin indirect en y ajoutant ou en y retirant des bulletins.

Attaque par urne restreinte. Dans tous les protocoles, si un scrutin indirect est défini de façon trop spécifique, il peut aboutir à des urnes ne contenant qu'un très faible nombre de bulletins, voire un seul. Dans ce cas, le secret du vote des électeurs concernés est compromis.

Attaque par intersection d'urnes. Dans tous les protocoles, lorsqu'un électeur appartient à plusieurs urnes indirectes, l'intersection des résultats peut réduire fortement son *anonymity set*. Dans certains cas, une simple soustraction des résultats d'urnes permet d'isoler son vote.

2.3 Adaptation des propriétés de sécurité

Avec le pastillage, de nouvelles propriétés de sécurité apparaissent, tandis que d'autres sont modifiées.

Secret du vote Un vote ne peut pas être relié à l'**identité** du votant.

Secret du vote pastillage Un vote ne peut pas être relié au **pastillage** du votant. Cette propriété est plus forte que le *secret du vote*, car même si l'identité reste masquée, un pastillage rare ou unique peut suffire à identifier le votant.

Secret du vote urnes Un vote ne peut pas être relié à la **liste des urnes** du votant. Cette propriété est plus forte que le *secret du vote pastillage*, car la liste des urnes fournit une information partielle sur le pastillage, qui peut permettre de réduire l'anonymat du votant.

Secret de participation L'**identité** des participants reste secrète.

Secret de participation pastillage Le **pastillage** des participants reste secret. Cette propriété est plus forte que le *secret de participation*, car les pastillages peuvent révéler des informations sensibles même lorsque l'identité des participants est masquée.

Secret de participation urnes La **liste des urnes** auxquelles un participant prend part reste secrète. Cette propriété est plus forte que le *secret de participation pastillage*, car la liste des urnes fournit souvent une information partielle sur les pastillages des participants.

Vérifiabilité individuelle (pastillage) Chaque électeur doit pouvoir vérifier que son vote a bien été pris en compte, dans l'ensemble des scrutins indirects auxquels il est éligible.

Vérifiabilité universelle (pastillage) La propriété traditionnelle. Tout observateur peut vérifier que le résultat publié correspond bien aux bulletins présents dans l'urne.

Vérifiabilité de l'éligibilité (pastillage) Il est possible de vérifier que chaque électeur participe uniquement aux scrutins indirects auxquels il est éligible.

3 Mises en œuvre du pastillage

Nous allons comparer quatre protocoles de vote électronique avec pastillage.

Protocole In Consiste à chiffrer les pastilles avec le vote.

Protocole Side Consiste à envoyer le vote chiffré et les pastilles (en clair).

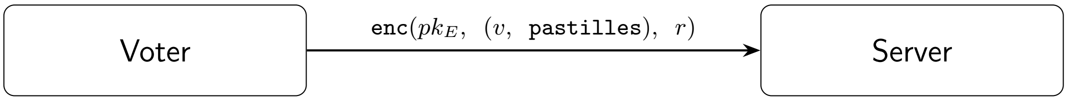
Protocole Urnes Consiste à envoyer le vote chiffré et la liste des urnes indirectes (en clair).

Protocole ZKP Consiste à envoyer le vote chiffré et des preuves zero-knowledge d'éligibilité aux scrutins indirects. Une variante consiste à envoyer un bulletin par scrutin indirect, nul si non-éligible.

Remarque. *Pour simplifier, certains détails importants sont omis. Par exemple, il est important d'ajouter une preuve de connaissance du nonce r utilisé pour chiffrer afin d'éviter la malléabilité. De plus, les votes sont signés.*

3.1 Protocole In

Les pastilles sont chiffrées avec le vote. Les bulletins sont anonymisés via des mélanges vérifiables (l'ajout des pastilles empêche la possibilité d'utiliser l'accumulation homomorphe). Après le déchiffrement, les résultats des scrutins indirects sont calculés à partir des votes ayant des pastillages éligibles.



Sécurité

Properties:	∅	**VS**	**Reg**	**VS + Reg**
Secret du vote	✓	✓	✓/✗*	✓/✗*
Secret du vote pastillage	✗	✗	✗	✗
Secret de participation	✓	✗	✗	✗
Secret de participation pastillage	✗	✗	✗	✗
Vérifiabilité individuelle	✓	✓	✓	✓
Vérifiabilité universelle	✓	✓	✓	✓
Vérifiabilité de l'éligibilité	✗	✗	✗	✗

(*) Si le votant ne vérifie pas ses pastilles, le registrar peut faire une **attaque par pastilles malicieuses**.

Attaques et limitations

Attaque par rareté. On réduit considérablement l'anonymat des votants ayant une combinaison de pastilles rare. Les votants ayant une combinaison unique sont complètement désanonymisés.

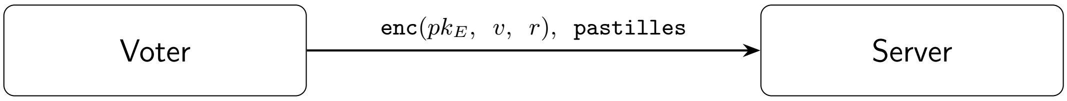
Attaque à l'italienne. Les électeurs peuvent utiliser le pastillage afin de fournir une preuve de vote et monnayer leur vote. Ce mécanisme permet également de **renverser un scrutin indirect** en votant aux urnes choisies.

Attaque par pastilles malicieuses. Si les votants ne vérifient pas bien leurs pastilles, l'administrateur d'une élection peut envoyer de mauvaises pastilles afin de désanonymiser certains votants. Ce mécanisme permet également de **renverser un scrutin indirect** en forçant à voter aux urnes choisies.

3.2 Protocole Side

Le votant envoie son bulletin ainsi que ses pastilles en clair. Pour chaque scrutin indirect, une urne est créée et remplie d'une copie des bulletins des votants éligibles. On procède ensuite normalement pour chaque urne : Les votes sont anonymisés (soit par accumulation homomorphe, soit par mélanges vérifiables) avant de procéder au dépouillement.

Remarque. *L'association entre pastilles et votant peut aussi être publiée au setup.*



Sécurité

Properties:	∅	**VS**	**Reg**	**VS + Reg**
Secret du vote	✓	✓	✓	✓/✗*
Secret du vote pastillage	✓	✓	✓	✓/✗*
Secret de participation	✓	✗	✗	✗
Secret de participation pastillage	✗	✗	✗	✗
Vérifiabilité individuelle	✓	✓	✓	✓
Vérifiabilité universelle	✓	✓	✓	✓
Vérifiabilité de l'éligibilité	✓	✓	✓	✗

(*) Si le votant ne vérifie pas ses pastilles, le registrar peut faire une **attaque par déplacement**.

Attaques et limitations

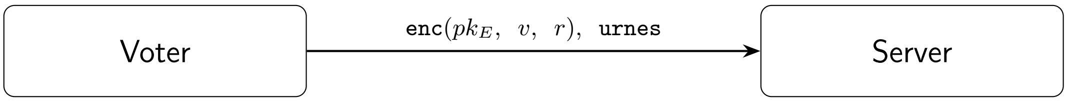
Attaque par urne restreinte. Certaines urnes peuvent ne contenir qu'un petit nombre de votants (voire un seul) et donc réduire considérablement l'*anonymity set* de cet ensemble de votants. De la même manière, l'**attaque par intersection d'urnes** est possible.

Attaque par déplacement. Si le votant ne vérifie pas ses pastilles, le registrar peut faire voter le votant dans les urnes qu'il veut en modifiant ses pastilles.

3.3 Protocole Urnes

Le votant envoie son bulletin ainsi que les urnes indirectes auxquelles il est éligible. On procède comme pour le Protocole Side, on copie chaque bulletin dans les urnes indirectes où il est éligible, on anonymise et dépouille normalement chaque urne indirecte afin d'obtenir les résultats.

Remarque. *L'association entre urnes et votant peut aussi être publiée au setup.*



Sécurité

Properties:	∅	**VS**	**Reg**	**VS + Reg**
Secret du vote	✓	✓	✓	✓/✗*
Secret du vote pastillage	✓	✓	✓	✓/✗*
Secret de participation	✓	✗	✗	✗
Secret de participation pastillage	✓	✗	✗	✗
Secret de participation urnes	✗	✗	✗	✗
Vérifiabilité individuelle	✓	✓	✓	✓
Vérifiabilité universelle	✓	✓	✓	✓
Vérifiabilité de l'éligibilité	✓	✓	✓	✗

(*) Si le votant ne vérifie pas ses pastilles, le registrar peut faire une **attaque par déplacement**.

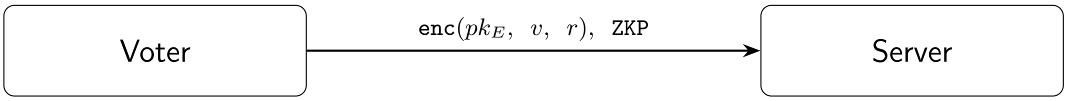
Attaques et limitations

Attaque par urne restreinte. Certaines urnes peuvent ne contenir qu'un petit nombre de votants (voire un seul) et donc réduire considérablement l'*anonymity set* de cet ensemble de votants. De la même manière, l'**attaque par intersection d'urnes** est possible.

Attaque par déplacement. Si le votant ne vérifie pas ses pastilles, le registrar peut faire voter le votant dans les urnes qu'il veut en modifiant ses pastilles.

3.4 Protocole ZKP

Les pastilles sont cachées dans des **engagements** ("commitments") publiés lors du setup. Le votant envoie son bulletin ainsi qu'une preuve zero-knowledge d'éligibilité pour chaque scrutin indirect. On procède normalement pour chaque urne afin d'obtenir les résultats.



Variantes. Selon le cas, le serveur ou le client peuvent faire les preuves :

EC (Éligibilité Client) Le votant fait les preuves zero-knowledge d'éligibilité aux urnes indirectes.
ES (Éligibilité Server) Le serveur fait les preuves zero-knowledge d'éligibilité aux urnes indirectes.

Deux constructions sont étudiées :

DN (Dispatch Normal) Le votant envoie une preuve zero-knowledge d'éligibilité par urne indirecte.
DZ (Dispatch Zero) Le votant envoie un bulletin par urne indirecte, couplé avec une preuve zero-knowledge assurant que : si éligible le bulletin capture la même intention de vote que le bulletin principal, si non éligible le bulletin est neutre (un chiffré de zéro).

Securité

Properties:	∅	**VS**	**Reg**	**VS + Reg**
Secret du vote	✓	✓	✓	✓/✗*
Secret du vote pastillage	✓	✓	✓	✓/✗*
Secret de participation	✓	✗	✗	✗
Secret de participation pastillage	✓	✓$^{\text{EC}}$/✗$^{\text{ES}}$	✗	✗
Secret de participation urnes	✓$^{\text{DZ}}$/✗$^{\text{DN}}$	✗	✗	✗
Vérifiabilité individuelle	✓	✓	✓	✓
Vérifiabilité universelle	✓	✓	✓	✓
Vérifiabilité de l'éligibilité	✓	✓	✓	✗

(*) Si le votant ne vérifie pas ses pastilles, le registrar peut faire une **attaque par déplacement**.

Attaques et limitations

Attaque par urne restreinte. Certaines urnes peuvent ne contenir qu'un petit nombre de votants (voire un seul) et donc réduire considérablement l'*anonymity set* de cet ensemble de votants. De la même manière, l'**attaque par intersection d'urnes** est possible.

Attaque par déplacement. Si le votant ne vérifie pas ses pastilles, le registrar peut faire voter le votant dans les urnes qu'il veut en modifiant ses pastilles.

4 Implémentations

Nos protocoles utilisent le squelette suivant :

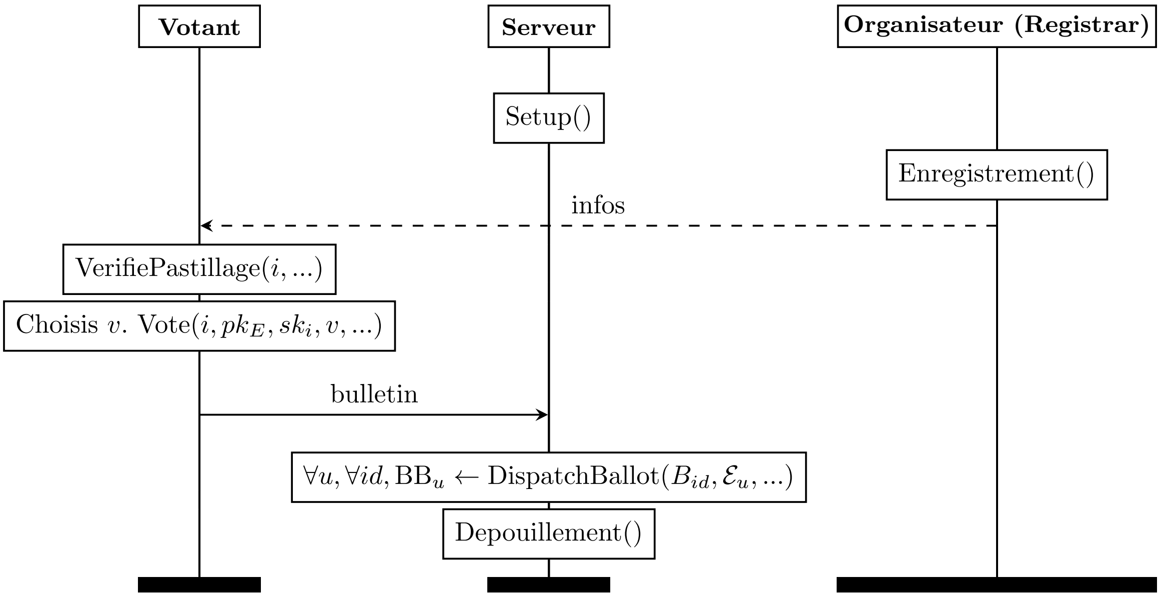


Figure 1: Les flèches pleines représentent les canaux publics et les flèches en pointillé les canaux privés.

4.1 Protocole In

Setup() Génère les paramètres de l'élection (clé de chiffrement pk_E / déchiffrement sk_E, pastilles possibles $(\mathbb{A}_t)_{t\in T}$, pastillages autorisés pour chaque urne $(\mathcal{E}_u)_{u\in U}$, etc.). Publie $pk_E, (\mathbb{A}_t)_{t\in T}, (\mathcal{E}_u)_{u\in U}$.

Enregistrement() Génère une clé de signature (sk_{id}, vk_{id}) pour chaque votant. Chaque votant est associé à une combinaison d'attributs a_{id}. Publie dans un ordre aléatoire $[vk_{id}]_{id\in V}$.

infos sk_i, vk_i, a_i.

VerifiePastillage(i, a_i**)** Le votant vérifie que ses attributs sont correct.

Vote(i, pk_E, sk_i, v, a_i**)** Génère un bulletin contenant le vote et les attributs $B_i = \mathrm{enc}(pk_E, (v, a_i), r \xleftarrow{\$} \mathbb{Z}_p)$, π_{pok} une "proof of knowledge" de r et $\sigma = \mathrm{sign}(sk_V, B)$.

bulletin B_i, π_{pok}, σ.

DispatchBallot($B_{id}, \mathcal{E}_u$**)** Tous les bulletins vont dans l'urne principale $\mathrm{BB}_{\mathrm{main}}$.

Depouillement() Dépouillement de $\mathrm{BB}_{\mathrm{main}}$ en utilisant la clé secrète de l'élection sk_E.

4.2 Protocole Side

Setup() Génère les paramètres de l'élection (clé de chiffrement pk_E / déchiffrement sk_E, pastilles possibles $(\mathbb{A}_t)_{t\in T}$, pastillages autorisés pour chaque urne $(\mathcal{E}_u)_{u\in U}$, etc.). Publie $pk_E, (\mathbb{A}_t)_{t\in T}, (\mathcal{E}_u)_{u\in U}$.

Enregistrement() Génère une clé de signature (sk_{id}, vk_{id}) pour chaque votant. Chaque votant est associé à une combinaison d'attributs a_{id}. Publie dans un ordre aléatoire $[vk_{id}, a_{id}]_{id\in V}$.

infos sk_i, vk_i.

VerifiePastillage(i, vk_i**)** Le votant vérifie sur le Bulletin Board que les attributs associés à sa clé de signature sont corrects.

Vote(i, pk_E, sk_i, v, a_i) Génère un bulletin contenant le vote $B_i = \mathrm{enc}(pk_E, v, r \xleftarrow{\$} \mathbb{Z}_p)$, π_{pok} une “proof of knowledge” de r et une signature $\sigma = \mathrm{sign}(sk_V, B)$.

bulletin B_i, π_{pok}, σ.

DispatchBallot($B_{id}, \mathcal{E}_u, a_{id}$) On sélectionne uniquement les bulletins tels que $a_{id} \in \mathcal{E}_u$.

Depouillement() On dépouille toutes les urnes indirectes $(\mathrm{BB}_u)_{u\in U}$ en utilisant la clé secrète de l'élection sk_E.

4.3 Protocole Urnes

Setup() Génère les paramètres de l'élection (clé de chiffrement pk_E / déchiffrement sk_E, pastilles possibles $(\mathbb{A}_t)_{t\in T}$, pastillages autorisés pour chaque urne $(\mathcal{E}_u)_{u\in U}$, etc.). Publie $pk_E, (\mathbb{A}_t)_{t\in T}, (\mathcal{E}_u)_{u\in U}$.

Enregistrement() Génère une clé de signature (sk_{id}, vk_{id}) pour chaque votant. Chaque votant est associé à une combinaison d'attributs a_{id}. Génère la liste des urnes où participe chaque votant u_{id}. Publie dans un ordre aléatoire $[vk_{id}, u_{id}]_{id\in V}$.

infos sk_i, vk_i.

VerifiePastillage(i, vk_i) Le votant vérifie sur le Bulletin Board que les urnes associés à sa clé de signature correspondent bien à ses attributs.

Vote(i, pk_E, sk_i, v, a_i) Génère un bulletin contenant le vote $B_i = \mathrm{enc}(pk_E, v, r \xleftarrow{\$} \mathbb{Z}_p)$, π_{pok} une “proof of knowledge” de r et une signature $\sigma = \mathrm{sign}(sk_V, B)$.

bulletin B_i, π_{pok}, σ.

DispatchBallot($B_{id}, \mathcal{E}_u, u, u_{id}$) On sélectionne uniquement les bulletins tels que $u \in u_{id}$.

Depouillement() On dépouille toutes les urnes indirectes $(\mathrm{BB}_u)_{u\in U}$ en utilisant la clé secrète de l'élection sk_E.

4.4 Protocole ZKP

Setup() Génère les paramètres de l'élection (clé de chiffrement pk_E / déchiffrement sk_E, pastilles possibles $(\mathbb{A}_t)_{t\in T}$, pastillages autorisés pour chaque urne $(\mathcal{E}_u)_{u\in U}$, etc.). Publie $pk_E, (\mathbb{A}_t)_{t\in T}, (\mathcal{E}_u)_{u\in U}$.

Enregistrement() Génère une clé de signature (sk_{id}, vk_{id}). Chaque votant est associé à une combinaison d'attributs a_{id}, et on génère un commitment et un opening sur ses attributs (c_{id}, r_{id}) (en utilisant **CommitPastillage**). Génère la liste des urnes où participe chaque votant u_{id}. Publie dans un ordre aléatoire $[vk_{id}, c_{id}]_{id\in V}$.

infos $sk_i, vk_i, c_i, r_i, a_i$.

VerifiePastillage(i, vk_i) Le votant vérifie sur le Bulletin Board que les commitments associés à sa clé de signature correspondent bien à ses attributs.

Vote(i, pk_E, sk_i, v, a_i) Génère un bulletin contenant le vote $B_i = \mathrm{enc}(pk_E, v, r \xleftarrow{\$} \mathbb{Z}_p)$, π_{pok} une “proof of knowledge” de r et une signature $\sigma = \mathrm{sign}(sk_V, B)$.
Eligibilité Client: Pour chaque urne indirecte, génère $dispatch_{i,u} = \mathrm{DispatchBallot}(B_{id}, \mathcal{E}_u, a_{id}, c_{id}, r_{id})$.
Eligibilité Server: Ne rien faire.

bulletin $B_i, \pi_{pok}, \sigma, (dispatch_i)$.

DispatchBallot($B_{id}, \mathcal{E}_u, u, u_{id}$) Dans la variante *Eligibilité Server*, pour chaque urne indirecte, génère $dispatch_{i,u} = \mathrm{DispatchBallot}(B_{id}, \mathcal{E}_u, a_{id}, c_{id}, r_{id})$.
DispatchBallot Normal: Utilise $dispatch_{i,u}$ afin de sélectionner ou non le bulletin pour cette urne.
DispatchBallot Zero: $B_{i,u}, \pi_u = dispatch_{i,u}$. Vérifie que $B_{i,u}$ est bien formé en utilisant π_u. Ajoute $B_{i,u}$ à l'urne BB_u

Depouillement() On dépouille toutes les urnes indirectes $(\mathrm{BB}_u)_{u\in U}$ en utilisant la clé secrète de l'élection sk_E.

Algorithmes

On utilise le chiffrement ElGamal (exponentiel) et les Pedersen commitments. Soit $\mathbf{a} = (a_1, \ldots, a_n)$ les attributs du votant. On considère une urne factorisable, i.e. $\mathcal{E}_u = \prod_i A_{u,i}$. Chaque votant a un vecteur de commitments $\mathbf{c}$, un par **type** d'attribut. On utilise les zero-knowledge proofs définies dans Annexe - Zero-Knowledge Proofs.

CommitPastillage

On génère un vecteur de Pedersen commitments, un par pastille.

Algorithm 1: $\texttt{CommitPastillage}(\mathbf{a})$

for $i \in [0..n]$ **do**
  $r_i \xleftarrow{\$} \mathbb{Z}_p$
  $c_i = g^{a_i} h^{r_i}$
Return $[c_i, r_i]_{i \in [1..n]}$

DispatchBallot Normal

On génère une preuve zero-knowledge d'éligibilité pour chaque urne. Pour chaque attribut, on génère l_i, un commitment égal à 1 si on est éligible pour l'attribut i (0 sinon). On calcule l_Σ la somme de tous les l_i puis on prouve en zero-knowledge que l_Σ est égal au nombre d'attributs (si éligible) ou inférieur (si non éligible).

Algorithm 2: $\texttt{DispatchBallot}(B_i, \mathcal{E}_u, \mathbf{a}, \mathbf{c}, \mathbf{r})$

for $i \in [0..n]$ **do**
  $r'_i \xleftarrow{\$} \mathbb{Z}_p$
  if $a_i \in \mathbb{A}_i$ **then**
    $l_i = g^1 h_i^{r'_i}$
  else
    $l_i = g^0 h_i^{r'}$
  $\pi_{l_i} = \texttt{bproof}[r_i, r'_i](c_i, \mathbb{A}_i, \mathbb{A}_i^c, l_i)$
$l_\Sigma = \prod_i l_i$
$r'_\Sigma = \sum_i r'_i$
if $\boldsymbol{a} \in E_u$ **then**
  $\pi_{\sqsubseteq} = \texttt{proof}_{\mathrm{DL}}[r'_\Sigma](l_\Sigma / g^n)$
  Return $B_i, \pi_{\sqsubseteq}, [l_i, \pi_{l_i}]_{i \in \mathbb{Z}_n}$
else
  $\pi_{\not\sqsubseteq} = \texttt{iproof}[r'_\Sigma](l_\Sigma, [0..(n-1)])$
  Return $\emptyset, \pi_{\not\sqsubseteq}, [l_i, \pi_{l_i}]_{i \in \mathbb{Z}_n}$

DispatchBallot Zero

On construit un nouveau bulletin et un ensemble de preuves zero-knowledge démontrant que : soit on est éligible et B_{out} capture la même intention de vote que B_{in}, ou on est non éligible et B_{out} est un chiffré de zéro. Pour ce faire, on procède comme DispatchBallot Normal, puis on calcule m un commitment sur le booléen d'éligibilité. Enfin on construit B_{out} en fournissant une preuve de rerand conditionnelle, conditionnée par m.

Algorithm 3: `DispatchBallot`$(B_i, \mathcal{E}_u, \mathbf{a}, \mathbf{c}, \mathbf{r})$

for $i \in [0..n]$ **do**
  $r'_i \xleftarrow{\$} \mathbb{Z}_p$
  if $a_i \in \mathbb{A}_i$ **then**
    $l_i = g^1 h_i^{r'_i}$
  else
    $l_i = g^0 h_i^{r'_i}$
  $\pi_{l_i} = \texttt{bproof}[r_i, r'_i](c_i, \mathbb{A}_i, \mathbb{A}_i^c, l_i)$
$l_\Sigma = \prod_i l_i$
$r'_\Sigma = \sum_i r'_i$
$r'' \xleftarrow{\$} \mathbb{Z}_p$
if $\boldsymbol{a} \in E_u$ **then**
  $m = g^1 h^{r''}$ // Is eligible
else
  $m = g^0 h^{r''}$
$\pi_m = \texttt{bproof}[r'_\Sigma, r'](l_\Sigma, [n], [0..(n-1)], m)$
$r''' \xleftarrow{\$} \mathbb{Z}_p$
if $\boldsymbol{a} \in E_u$ **then**
  $B_{\text{out}} \leftarrow (\alpha_{\text{in}} \times g^{r'''},\ \beta_{\text{in}} \times y^{r'''})$
else
  $B_{\text{out}} \leftarrow (g^{r'''},\ y^{r'''})$
$\pi_{dispatch} = \texttt{rproof}[r, r'''](m, B_{\text{in}}, B_{\text{out}})$
Return $B_{\text{out}}, \pi_{dispatch}, [l_i, \pi_{l_i}]_{i \in [1..n]}, m, \pi_m$

5 Vérification formelle

Garantir la sécurité d'un protocole cryptographique est une tâche complexe : de nouvelles vulnérabilités sont régulièrement découvertes, parfois plusieurs années après la conception initiale [16].La vérification formelle vise à établir une preuve de sécurité d'un protocole cryptographique, donc de démontrer l'absence d'attaque. Cette approche est aujourd'hui incontournable, exigée notamment pour la conception des protocoles de vote électronique en Suisse [5]. Dans cette section, nous présentons ProVerif, comment nous l'avons utilisé pour modéliser les protocoles, et les résultats que nous avons obtenus.

5.1 ProVerif

ProVerif [4] est un outil de vérification formelle des protocoles cryptographiques dans le modèle symbolique. Le protocole est formalisé en **pi-calcul appliqué**. Les messages échangés sont des **termes**, c'est-à-dire des expressions construites à partir de constructeurs appliqués à des variables, constantes, ou d'autres termes. Les **nonces** sont des variables aléatoires que l'attaquant a une chance négligeable de deviner. Une **théorie équationnelle** décrit les opérations possibles sur les termes. Enfin, on décrit des **processus** qui spécifient le comportement des participants honnêtes. ProVerif traduit le protocole en un système de clauses logiques qu'il explore automatiquement. L'**attaquant** est omnipotant sur le réseau, il peut intercepter, bloquer, injecter, ou modifier tous message sur le réseau. Il est capable de construire de nouveau messages construits avec les termes qu'il a observé. En revanche il ne peut pas casser la cryptographie.

Listing 1: Syntaxe de ProVerif

```
(* Nonce *)
new k;
(* Termes *)
fun enc(bitstring,bitstring):bitstring.
let c = enc(k, m).
```

```
(* Theorie equationnelle *)
fun enc(bitstring,bitstring):bitstring. (* encryption *)
reduc forall m:bitstring, k:bitstring;  (* decryption *)
  dec(enc(k,m), k) = m.
(* Channel *)
free c:channel;
(* Processus *)
let Alice() =
  out(c, 23).
let Bob() =
  in(c, n); out(c, 42).
(* Main process *)
process
  Alice() | Bob()
```

5.2 Modélisation

Nous présentons ici comme exemple la modélisation du Protocole In.

Listing 2: Protocole In

```
(* Le votant reçoit sa clé privée depuis le registrar, chiffre son vote et ses
pastilles et envoie le résultat au serveur. *)
let Voter(id:id_t, v:bitstring) =
  in(c_reg(id), (scred:sskey, a:attr_t));
  let cred = spk(scred) in
  (* generate ballot *)
  new r:rand;
  new s:rand;
  let c = aenc((v, a), pk_e, r) in
  let sig = sign(c, scred, s) in
  let pi = zkp2(r, cred, a) in
  let b = (c, sig, pi) in
  (* cast ballot *)
  event Voted(id, v);
  out(public, (cred, b))
  | out(c_server(id), (cred, b)) (* Publish ballot *)
  (* verification *)
  | get BULLETIN_BOARD(=BBmain, =cred, =b) in
  event Verified(BBmain, id, v)

(* Le serveur reçoit un bulletin du votant, le vérifie puis le publie sur le
bulletin board. *)
let Server(id:id_t) =
  in(c_server(id), (cred:spkey, b:bitstring));
  get CREDENTIALS(_, =cred, _) in (* credential exist *)
  if (verify_all(cred, b))
  then (
    event GoingToTally(BBmain, id, cred, b);
    insert BULLETIN_BOARD(BBmain, cred, b);
  ).

(* Main process *)
process
  in(public, (a1:attr_t, a2:attr_t));
  Registrar(Alice, a1, scred_Alice) | Registrar(Bob, a2, scred_Bob)
  | Client(Alice, choice[A,B])   | Client(Bob, choice[B,A])
  | Server(Alice) | Server(Bob)
  | Tally()
```

5.3 Propriétés de sécurité

Le secret du vote est généralement défini comme une propriétés d'équivalence observationnelle.

Secret du vote. On utilise la définition habituelle [12]. On donne les même pastilles à Alice et Bob.

```
equivalence
  Voter(Alice, CandidateA, attr) | Voter(Bob, CandidateB, attr)
  Voter(Alice, CandidateB, attr) | Voter(Bob, CandidateA, attr)
```

Secret du vote pastillage. Cette propriété est vraie si on peut connaître le pastillage d'un vote.

Remarque. *On doit s'assurer qu'Alice et Bob votent aux mêmes urnes.*

```
equivalence
  Voter(Alice, CandidateA, attr1) | Voter(Bob, CandidateB, attr2)
  Voter(Alice, CandidateB, attr2) | Voter(Bob, CandidateA, attr1)
```

Vérifiabilité

Vérifiabilité individuelle. Si l'électeur vérifie, son bulletin correspond bien à son intention de vote (**cast-as-intended**) et les bulletins sont enregistrés tels qu'envoyés par l'électeur (**recorded-as-cast**).

```
(* cast-as-intended *)
query bb: bb_t, b:bitstring, id,id':id_t, cred,cred':spkey, v:bitstring, a:attr_t;
event(Verified(BBmain, id, v)) && event(IsEligible(id, bb)) && event(DispatchEnd(bb, id, cred)) ==>
event(Voter(id, cred, honest)) && event(GoingToTally(bb, id', cred, (b,a))) && open_eq(v, b).
(* recorded-as-cast *)
query bb: bb_t, b:bitstring, id,id':id_t, cred,cred':spkey, v:bitstring, a:attr_t;
event(GoingToTally(bb, id, cred, (b,a))) && event(Voter(id, cred', honest)) ==>
event(Voted(id, v)) && open_eq(v, b).
```

Vérifiabilité universelle / tallied-as-recorded. Cette propriété est assurée par la cryptographie.

Verifiabilité de l'éligibilité. Les bulletins enregistrés proviennent bien d'électeurs éligibles.

```
query bb: bb_t, b:bitstring, id:id_t, cred,cred':spkey, v:bitstring, l:status_t, a:attr_t;
event(GoingToTally(bb, id, cred, (b,a))) ==> event(Voter(id, cred', l)) && event(IsEligible(id, bb)).
```

5.4 Résultats

Nous avons exécuté nos modèles avec ProVerif et obtenu les résultats suivants :

	Confidentialité							
	Secret du vote				**Secret du vote pastillage**			
Scénario de corruption	HH	CH	HC	CC	HH	CH	HC	CC
Protocole In	✓ (484s)	✓ (6s)	✓oot	✓oot	✗	✗	✗	✗
Protocole Side	✓ (30s)	✓ (71s)	✓oot	✓oot	✓ (107s)	✓ (138s)	✓oot	✓oot
Protocole Urnes	✓ (2.5s)	✓ (60s)	✓oot	✓oot	✓ (2.9s)	✓ (61s)	✓oot	✓oot
Protocole ZKP	✓	✓	✓	✓	✓	✓	✓	✓

	Vérifiabilité											
	cast-as-intended				**recorded-as-cast**				**éligibilité**			
Scénario de corruption	HH	CH	HC	CC	HH	CH	HC	CC	HH	CH	HC	CC
Protocole In	✓	✓	✓cbp	✓cbp	✓	✓	✓cbp	✓cbp	✗	✗	✗	✗
Protocole Side	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Protocole Urnes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Protocole ZKP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗

Légende.

- **Scénarios de corruption** :

  HH Serveur honnête + Registrar honnête

  CH Serveur malhonnête + Registrar honnête

  HC Serveur honnête + Registrar malhonnête

  CC Serveur malhonnête + Registrar malhonnête

- **Symboles** :

  ✓ Propriété démontrée

  ✗ Propriété réfutée

  ✓oot Analyse non concluante (dépassement de temps (*out of time*))

  ✓cbp Analyse non concluante (impossibilité de conclure (*cannot be proved*))

  ✓ Analyse non concluante (autre cas)

6 Conclusion

Cette étude s'intéresse au mécanisme de pastillage dans le contexte du vote électronique. Le pastillage consiste à associer à chaque électeur un ensemble de pastilles (telles que l'âge ou la profession) afin de déterminer sa participation à différentes urnes indirectes.

L'analyse met en évidence plusieurs risques pour le **secret du vote**. En particulier, le pastillage peut réduire de manière significative l'*anonymity set* associé à un suffrage (c'est-à-dire l'ensemble des électeurs auxquels ce vote pourrait appartenir). Dans certains cas, des attaques permettent même de désanonymiser intégralement le vote d'électeurs. Cela se produit notamment avec le protocole In si des pastilles malicieuses sont attribuées à un électeur, ou encore lorsque des scrutins indirects trop restreints sont définis. Des vulnérabilités concernant la **vérifiabilité** ont également été observées, en particulier pour le protocole In.

Nous proposons également une nouvelle construction reposant sur l'utilisation de preuves à divulgation nulle de connaissance (ZKP), afin de garder les pastilles secrètes tout en permettant au protocole d'être universellement vérifiable.

Enfin, afin d'évaluer rigoureusement ces protocoles, chacun a été modélisé avec l'outil **ProVerif**, ce qui nous a permis de démontrer formellement certaines propriétés de sécurité et de vérifier l'absence d'attaques. Des écarts notables en termes de sécurité apparaissent entre les quatre implémentations de pastillage que nous avons étudiées.

Pistes de recherche

Approfondir les preuves à divulgation nulle de connaissance Les protocoles reposant sur les preuves à divulgation nulle de connaissance (Zero-Knowledge Proof, ZKP) sont prometteurs, mais pourraient être mieux définis, analysés et explorés. Il serait pertinent d'explorer d'autres cadres technologiques de ZKP (SNARKs, STARKs, ...) afin de comparer leurs performances, garanties de sécurité et applicabilité au contexte du pastillage.

Explorer les "anonymous credentials" Les "anonymous credentials", et en particulier les signatures BBS+, permettraient au votant de révéler sélectivement certaines pastilles tout en conservant le secret des autres. Une piste intéressante consisterait à concevoir un protocole dans lequel une autorité de confiance envoie à chaque votant ses pastilles signées via une signature BBS+. Cette approche pourrait s'intégrer à certaines implémentations d'EUID (European Unified ID).

Développer et enrichir les modèles formels Les modèles ProVerif pourraient être étendus afin de vérifier un plus grand nombre de propriétés, notamment en matière de robustesse et de confidentialité. L'utilisation d'autres outils d'analyse formelle, tels que Tamarin, permettrait également de diversifier les approches et de valider les résultats obtenus.

Mettre en œuvre une preuve de concept Une étape importante serait la réalisation d'un prototype fonctionnel intégrant le mécanisme de pastillage dans un système de vote électronique existant, tel que Belenios. Cette expérimentation offrirait l'opportunité d'évaluer concrètement les performances, la faisabilité technique et l'expérience utilisateur.

7 Annexe - Zero-Knowledge Proofs

We define here the Zero-Knowledge Proofs used in sous-section 3.4. They are inspired by those used in Belenios [14, 13]. The main types of proofs are:

- $\texttt{proof}_{\mathbf{DL}}$: Proof of knowledge of a discrete logarithm.
- $\texttt{proof}_{\mathbf{CP}}$: Chaum-Pedersen proof: knowledge of r such that $\alpha = g^r$ and $\beta = y^r$.
- $\texttt{proof}_{\mathbf{OR}}$: OR-proof.
- $\texttt{proof}_{\mathbf{MIX}}$: Mixed OR-proof: Either $C = g^r$ or $(\alpha, \beta) = (g^r, y^r)$.
- `iproof`: Proof of set membership.
- `bproof`: Blinded proof of set membership.
- `rproof`: Rerandomization proof.

Public context: Group $G = \langle g, h \rangle$, with $\#G = q$, where g and h are independent generators. A public encryption key y is known. The special variable S holds all the context for a given statement.

$\texttt{proof}_{\mathbf{DL}}$: Knowledge of a Discrete Logarithm

Statement

$$\text{I know } r \text{ such that } C = g^r$$

Remark. *We can use this to prove knowledge of an opening of a Pedersen Commitment ($C = g^a h^r$) to a specific value (a) by running the protocol on C/g^a.*

Algorithms

Algorithm 4: $\texttt{proof}_{\mathrm{DL}}[r](C)$

$w \xleftarrow{\$} \mathbb{Z}_p; A = g^w$
$\mathsf{challenge} = \mathsf{SHA256}(S||A)$
$\mathsf{response} = w - r \times \mathsf{challenge}$
Return $\mathsf{challenge}, \mathsf{response}$

Algorithm 5: $\texttt{verifyProof}_{\mathrm{DL}}(C, \mathsf{challenge}, \mathsf{response})$

$A = g^{\mathsf{response}} \times C^{\mathsf{challenge}}$
Check that $\mathsf{challenge} = \mathsf{SHA256}(S||A)$

Security proofs

See "Knowledge a discrete logarithm" in [13]

$\texttt{proof}_{\mathbf{CP}}$: Knowledge of equality of two DL (Chaum-Pedersen)

Statement

$$\text{I know } r \text{ such that } \alpha = g^r \text{ and } \beta = y^r$$

Remark. *We can use this to prove that an ElGamal ciphertext is an encryption of 0.*

Remark. *We can use this to prove that* (α_2, β_2) *is a rerand of* (α_1, β_1) *by proving that* $(\alpha_2/\alpha_1, \beta_2/\beta_1)$ *is an encryption of 0.*

Algorithms

Algorithm 6: $\texttt{proof}_{\mathrm{CP}}[\mathrm{r}](\alpha, \beta)$

$w \xleftarrow{\$} \mathbb{Z}_p; A = g^w; B = y^w$
$\mathsf{challenge} = \mathsf{SHA256}(S||A||B)$
$\mathsf{response} = w - r \times \mathsf{challenge}$
Return $\mathsf{challenge}, \mathsf{response}$

Algorithm 7: $\texttt{verifyProof}_{\mathrm{CP}}(\alpha, \beta, \mathsf{challenge}, \mathsf{response})$

Compute $A = g^{\mathsf{response}} \times \alpha^{\mathsf{challenge}}; B = y^{\mathsf{response}} \times \beta^{\mathsf{challenge}}$
Check that $\mathsf{challenge} = \mathsf{SHA256}(S||A||B)$

Security proofs

See "Proof of set membership" in [13] (in the special case where the set of possibles messages is $\{0\}$).

$\texttt{proof}_{\mathbf{OR}}$: OR-proof of knowledge of a DL

Statement

$$\text{I know } r \text{ such that } C_j = g^r \in \{C_0, \dots, C_n\}$$

Algorithms

Algorithm 8: $\texttt{proof}_{\mathrm{OR}}[\mathrm{r}](C_0, \dots, C_n)$

for $i \neq j$ **do**
    $\mathsf{challenge}_i, \mathsf{response}_i \xleftarrow{\$} \mathbb{Z}_p$
    $A_i = g^{\mathsf{response}_i} \times C_i^{\mathsf{challenge}_i}$
$w \xleftarrow{\$} \mathbb{Z}_p; A_j = g^w$
$\mathsf{challenge}_j = \mathsf{SHA256}(S||A_0|| \dots ||A_n) - \sum_{j \neq i} \mathsf{challenge}_i$
$\mathsf{response}_j = w - r \times \mathsf{challenge}_j$
Return $\mathsf{challenge}_0, \mathsf{response}_0, \dots, \mathsf{challenge}_n, \mathsf{response}_n$

Algorithm 9: $\texttt{verifyProof}_{\mathrm{OR}}(C_0, \dots, C_n, \mathsf{challenge}_0, \mathsf{response}_0, \dots, \mathsf{challenge}_n, \mathsf{response}_n)$

for i **do**
    $A_i = g^{\mathsf{response}_i} \times C_i^{\mathsf{challenge}_i}$
Check that $\sum_i \mathsf{challenge}_i = \mathsf{SHA256}(S||A_0|| \dots ||A_n)$

Security proofs

Completeness: By construction (we can follow the algorithm for all inputs).

Zero-Knowledge: Anyone can create a valid transcript as follows: pick a random e, pick random pairs $(\mathsf{challenge}_j, \mathsf{response}_j)$, for all $j \in [0, n]$, except for $\mathsf{challenge}_0$ that is computed as $\mathsf{challenge}_0 = e - \sum_{j \in [1,k]} \mathsf{challenge}_j$. Then the A_j are just computed from the formula used for the verification. This valid transcript has the same probability distribution than a genuine transcript and is therefore indistinguishable.

Soundness: Facilement adaptée de la preuve de soundness de "Proof of set membership" dans [13].

`proof`$_{\text{MIX}}$: OR-proof mixing the two

Statement

$$\text{I know } r \text{ such that } C = g^r \text{ or } (\alpha, \beta) = (g^r, y^r).$$

Remark. *We can use this to construct* *`rproof`*

Algorithms

Algorithm 10: `proof`$_{\text{MIX}}[\text{r}](C, \alpha, \beta)$

if $C = g^r$ **then**
- $\mathsf{challenge}_1, \mathsf{response}_1 \xleftarrow{\$} \mathbb{Z}_p$
- $A_1 = g^{\mathsf{response}_1} \times \alpha^{\mathsf{challenge}_1}$
- $B_1 = y^{\mathsf{response}_1} \times \beta^{\mathsf{challenge}_1}$
- $w \xleftarrow{\$} \mathbb{Z}_p$
- $A_0 = g^w$
- $\mathsf{challenge}_0 = \mathsf{SHA256}(S||A_0||A_1||B_1)$
- $\mathsf{response}_0 = w - r \times \mathsf{challenge}_0$

else
- // When $(\alpha, \beta) = (g^r, y^r)$
- $\mathsf{challenge}_0, \mathsf{response}_0 \xleftarrow{\$} \mathbb{Z}_p$
- $A_0 = g^{\mathsf{response}_0} \times C^{\mathsf{challenge}_0}$
- $w \xleftarrow{\$} \mathbb{Z}_p$
- $A_1 = g^w$
- $\mathsf{challenge}_1 = \mathsf{SHA256}(S||A_0||A_1||B_1)$
- $\mathsf{response}_1 = w - r \times \mathsf{challenge}_1$

Return $\mathsf{challenge}_0, \mathsf{response}_0, \mathsf{challenge}_1, \mathsf{response}_1$

Algorithm 11: `verifyProof`$_{\text{MIX}}(C, \alpha, \beta, \mathsf{challenge}_0, \mathsf{response}_0, \mathsf{challenge}_1, \mathsf{response}_1)$

$A_0 = g^{\mathsf{response}_0} \times C_i^{\mathsf{challenge}_0}$

$A_1 = g^{\mathsf{response}_1} \times \alpha^{\mathsf{challenge}_1}$

$B_1 = y^{\mathsf{response}_1} \times \beta^{\mathsf{challenge}_1}$

Check that $(\mathsf{challenge}_0 + \mathsf{challenge}_1) = \mathsf{SHA256}(S||A_0||A_1||B_1)$

Security proofs

Completeness: By construction (we can follow the algorithm for all inputs).

Zero-Knowledge: Anyone can create a valid transcript as follows: pick a random e, pick random $\mathsf{response}_0, \mathsf{challenge}_1, \mathsf{response}_1$ and compute $\mathsf{challenge}_0 = e - \mathsf{challenge}_1$. Then the A_0, A_1, B_1 are computed from the formula used for the verification. This valid transcript has the same probability distribution than a genuine transcript and is therefore indistinguishable.

Special-Soundness: Let $((A_0, A_1, B_1), e, (\mathsf{challenge}_0, \mathsf{response}_0, \mathsf{challenge}_1, \mathsf{response}_1))$ and $((A_0, A_1, B_1), e', (\mathsf{challenge}'_0, \mathsf{response}'_0, \mathsf{challenge}'_1, \mathsf{response}'_1))$ be two distinct valid transcripts with the same commitments. Assume that we have $\mathsf{challenge}_0 = \mathsf{challenge}'_0$ and $\mathsf{challenge}_1 = \mathsf{challenge}'_1$, it follows that $\mathsf{response}_0 = \mathsf{response}'_0$ and $\mathsf{response}_1 = \mathsf{response}'_1$ and $e = \mathsf{challenge}_0 + \mathsf{challenge}_1$ which contradicts that the transcripts are distincts. Therefore, $\mathsf{challenge}_0 \neq \mathsf{challenge}'_0$ or $\mathsf{challenge}_1 \neq \mathsf{challenge}'_1$. Suppose that $\mathsf{challenge}_0 \neq \mathsf{challenge}'_0$ (the proof is similar if $\mathsf{challenge}_1 \neq \mathsf{challenge}'_1$). We have $A_0 = g^{\mathsf{response}_0} \times c^{\mathsf{challenge}_0} = g^{\mathsf{response}'_0} \times c^{\mathsf{challenge}'_0}$ so $g^{\mathsf{response}_0 - \mathsf{response}'_0} = c^{\mathsf{challenge}_0 - \mathsf{challenge}'_0}$. It follows that we can compute the value $r = (\mathsf{response}_0 - \mathsf{response}'_0)/(\mathsf{challenge}_0 - \mathsf{challenge}'_0) \bmod p$ where the division is well defined since $\mathsf{challenge}_0 \neq \mathsf{challenge}'_0$. Therefore, we have constructed a value r such that the statement is true.

`iproof`: Proof of set membership

Statement

I know an opening of c to an $a \in \mathbb{A}$.

Definition

$$\texttt{iproof}[r](c, \mathbb{A}) : \mathrm{open}(c, a, r) \text{ with } a \in \mathbb{A}$$

Algorithms

This complex statement is built using the following proofs:

$$\text{Let } \mathbb{A} = \{a_0, \ldots, a_n\}$$

$$\pi : \texttt{proof}_{\mathrm{OR}}[r](c/g^{a_0}, \ldots, c/g^{a_n})$$

`bproof`: Blinded proof of set membership

Statement

Either c opens to $a \in \mathbb{A}$ and l opens to 1, or c opens to $b \in \mathbb{B}$ and l opens to 0.

Definition

$$\texttt{bproof}[r, r'](C, \mathbb{A}, \mathbb{B}, l) : l = \begin{cases} \mathrm{com}(1) & \text{si } c \in \mathrm{com}(\mathbb{A}) \\ \mathrm{com}(0) & \text{si } c \in \mathrm{com}(\mathbb{B}) \end{cases}$$

Algorithms

This complex statement is built using the following proofs:

$$\text{Let } \mathbb{A} = \{a_0, \ldots, a_{|\mathbb{A}|}\} \text{ and } \mathbb{B} = \{b_0, \ldots, b_{|\mathbb{B}|}\}$$

$$\pi_0 : \texttt{proof}_{\mathrm{OR}}[r](l, c/g^{a_1}, \ldots, c/g^{a_{|A|}}) \tag{1}$$
$$\pi_1 : \texttt{proof}_{\mathrm{OR}}[r'](l/g^1, c/g^{b_1}, \ldots, c/g^{b_{|B|}}) \tag{2}$$
$$\pi_2 : \texttt{proof}_{\mathrm{OR}}[r'](l, l/g^1) \tag{3}$$

Remark. *Interchange r and r' in π_0 and π_1 when the second case is true.*

`rproof`: Rerandomization proof

Statement

Either l opens to 1 and $(\alpha_{\text{out}}, \beta_{\text{out}})$ is a rerandomization of $(\alpha_{\text{in}}, \beta_{\text{in}})$,
or l opens to 0 and $(\alpha_{\text{out}}, \beta_{\text{out}})$ is an encryption of 0.

Definition

$$\texttt{rproof}[r, r'](l, \alpha_{in}, \beta_{in}, \alpha_{out}, \beta_{out}) : (\alpha_{out}, \beta_{out}) = \begin{cases} \text{rerand}((\alpha_{in}, \beta_{in}), r') & \text{si open}(l, 1, r) \\ \text{encrypt}(pk, 0, r') & \text{sinon} \end{cases}$$

Algorithms

This complex statement is built using the following proofs:

$$\pi_0 : \texttt{proof}_{\text{MIX}}[r](l, \alpha_{out}, \beta_{out}) \tag{4}$$
$$\pi_1 : \texttt{proof}_{\text{MIX}}[r'](l/g^1, \alpha_{out}/\alpha_{in}, \beta_{out}/\beta_{in}) \tag{5}$$
$$\pi_2 : \texttt{proof}_{\text{OR}}[r'](l, l/g^1) \tag{6}$$

Remark. *Interchange r and r' in π_0 and π_1 when the second case is true.*

Références

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[2] ANSSI. Recommandations pour la mise en œuvre du vote par internet pour les élections non politiques – version pour consultation publique. `https://cyber.gouv.fr/sites/default/files/document/GuideVPI_consultation_publique.pdf`, 2025. Consultation publique, version 0.1.

[3] Steven M Bellovin and Michael Merritt. Limitations of the kerberos authentication system. *ACM SIGCOMM Computer Communication Review*, 20(5):119–132, 1990.

[4] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. *Version from*, 16:05–16, 2018.

[5] Swiss Federal Chancellery. Federal chancellery ordinance on electronic voting (oev). `https://www.bk.admin.ch/dam/bk/en/dokumente/pore/OEV_draftforconsultation2021.pdf.download.pdf/OEV_draftforconsultation2021.pdf`, 2021. Draft of 28 April 2021.

[6] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

[7] Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368. IEEE, 2008.

[8] CNIL. Sécurité des systèmes de vote par correspondance électronique (sve) – projet de recommandation. `https://www.cnil.fr/sites/cnil/files/2025-01/projet_de_recommandation_securite_des_systemes_de_vote_par_correspondace_electronique.pdf`, 2025. Consultation publique jusqu'au 28 février 2025.

[9] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798. IEEE, 2016.

[10] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: a simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning: Essays Dedicated to Catherine A. Meadows*, pages 214–238. Springer, 2019.

[11] Alexandre Debant and Lucca Hirschi. Reversing, breaking, and fixing the french legislative election {E-Voting} protocol. In *32nd usenix security symposium (usenix security 23)*, pages 6737–6752, 2023.

[12] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[13] Pierrick Gaudry. Some zk security proofs for belenios. 2017.

[14] Stéphane Glondu. Belenios specification. *Version 0.1. http://www. belenios. org/specification. pdf*, 2013.

[15] Sven Heiberg and Jan Willemson. Verifiable internet voting in estonia. In *2014 6th international conference on electronic voting: Verifying the vote (evote)*, pages 1–8. IEEE, 2014.

[16] Gavin 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*, pages 147–166. Springer, 1996.

[17] Swiss Post. Swiss post voting system: system specification. `https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/`, 2025.