Méthodes formelles en sécurité

avec quelques intermèdes socio-politiques

Lê Thành Dũng Nguyễn — chercheur au Laboratoire d'Informatique et des Systèmes

Journée DevSec (vendredi 28 février 2025, campus de Luminy)

- **Primitives cryptographiques :** chiffrement, signature, ...

- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »

- **Implémentations :** protocole idéalisé ⟶ vrai code (sur du vrai matériel)

Les maths pour la sécurité ?

- **Primitives cryptographiques :** chiffrement, signature, ...
- **Protocoles crypto :** « en supposant que les primitives sont solides, TLS doit garantir que la connexion est sécurisée »
- **Implémentations :** protocole idéalisé ⟶ vrai code (sur du vrai matériel)

Les maths pour la sécurité ?

- **Primitives cryptographiques :** chiffrement, signature, ...
  algèbre, par ex. : casser RSA $\iff$ trouver rapidement des facteurs premiers
- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »

- **Implémentations :** protocole idéalisé $\longrightarrow$ vrai code (sur du vrai matériel)

Les maths pour la sécurité ?

- **Primitives cryptographiques :** chiffrement, signature, ...
  algèbre, par ex. : casser RSA $\iff$ trouver rapidement des facteurs premiers
- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »
  logique formelle, modélisation par automates, ...
- **Implémentations :** protocole idéalisé $\longrightarrow$ vrai code (sur du vrai matériel)

Les maths pour la sécurité ?

- **Primitives cryptographiques :** chiffrement, signature, ...
  algèbre, par ex. : casser RSA $\iff$ trouver rapidement des facteurs premiers
- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »
  logique formelle, modélisation par automates, ...
- **Implémentations :** protocole idéalisé $\longrightarrow$ vrai code (sur du vrai matériel)
  analyse statique, preuves de programmes vérifiées sur machine, ...

Les maths pour la sécurité ? Notamment les **méthodes formelles**

- **Primitives cryptographiques :** chiffrement, signature, ...
  algèbre, par ex. : casser RSA $\iff$ trouver rapidement des facteurs premiers
- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »
  logique formelle, modélisation par automates, ...
- **Implémentations :** protocole idéalisé $\longrightarrow$ vrai code (sur du vrai matériel)
  analyse statique, preuves de programmes vérifiées sur machine, ...

Les maths pour la sécurité ? Notamment les **méthodes formelles**

- **Primitives cryptographiques :** chiffrement, signature, ...
  algèbre, par ex. : casser RSA $\iff$ trouver rapidement des facteurs premiers
- **Protocoles crypto :** « en supposant que les primitives sont solides,
  TLS doit garantir que la connexion est sécurisée »
  logique formelle, modélisation par automates, ...
- **Implémentations :** protocole idéalisé $\longrightarrow$ vrai code (sur du vrai matériel)
  analyse statique, preuves de programmes vérifiées sur machine, ...

Mais la sécurité n'est pas qu'une question technique...

Le protocole d'authentification de Needham–Schroeder

Alice veut prouver à Bob que c'est bien elle qui lui parle

1. Alice et Bob génèrent chacun un nombre aléatoire N_A / N_B
2. Alice envoie à Bob : «je suis Alice et voici N_A» *chiffré avec la clé publique de Bob*
   $\longrightarrow$ seul Bob peut déchiffrer le message

Le protocole d'authentification de Needham–Schroeder

Alice veut prouver à Bob que c'est bien elle qui lui parle

1. Alice et Bob génèrent chacun un nombre aléatoire N_A / N_B
2. Alice envoie à Bob : «je suis Alice et voici N_A» *chiffré avec la clé publique de Bob*
   $\longrightarrow$ seul Bob peut déchiffrer le message
3. Bob envoie à Alice (N_A, N_B) *chiffré avec la clé publique d'Alice*
   $\longrightarrow$ Alice déchiffre le message et vérifie qu'il commence bien par N_A

Le protocole d'authentification de Needham–Schroeder

Alice veut prouver à Bob que c'est bien elle qui lui parle

1. Alice et Bob génèrent chacun un nombre aléatoire N_A / N_B
2. Alice envoie à Bob : «je suis Alice et voici N_A» *chiffré avec la clé publique de Bob*
   $\longrightarrow$ seul Bob peut déchiffrer le message
3. Bob envoie à Alice (N_A, N_B) *chiffré avec la clé publique d'Alice*
   $\longrightarrow$ Alice déchiffre le message et vérifie qu'il commence bien par N_A
4. Alice envoie à Bob N_B chiffré avec la clé publique de Bob

Le protocole d'authentification de Needham–Schroeder

Alice veut prouver à Bob que c'est bien elle qui lui parle

1. Alice et Bob génèrent chacun un nombre aléatoire N_A / N_B
2. Alice envoie à Bob : «je suis Alice et voici N_A» *chiffré avec la clé publique de Bob*
   $\longrightarrow$ seul Bob peut déchiffrer le message
3. Bob envoie à Alice (N_A, N_B) *chiffré avec la clé publique d'Alice*
   $\longrightarrow$ Alice déchiffre le message et vérifie qu'il commence bien par N_A
4. Alice envoie à Bob N_B chiffré avec la clé publique de Bob

Hypothèse sur les primitives : le chiffrement à clé publique est sécurisé

Une attaque sur le protocole [Lowe 1995] ...

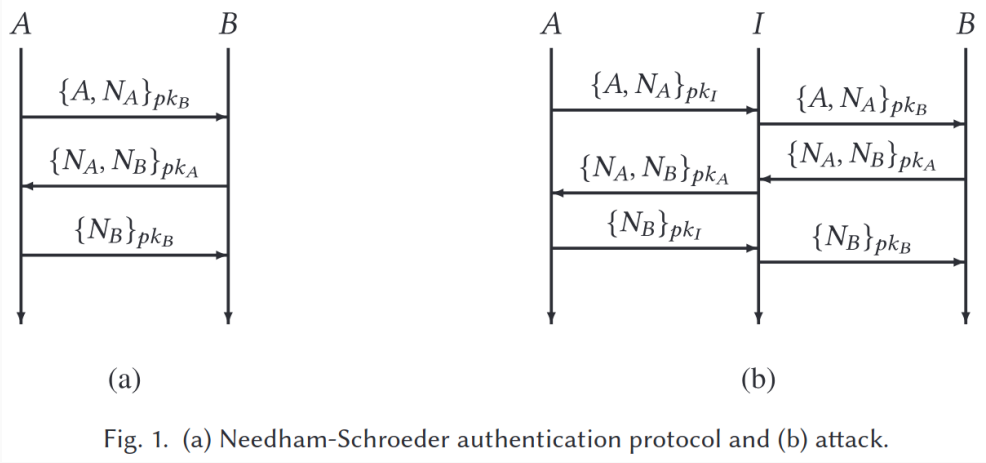


Fig. 1. (a) Needham-Schroeder authentication protocol and (b) attack.

Figure : Kulik et al. 2022, *A Survey of Practical Formal Methods for Security*

... découverte par une méthode formelle ! [Lowe 1996]

Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR

Gavin Lowe*

ABSTRACT In this paper we analyse the well known Needham-Schroeder Public-Key Protocol using FDR, a refinement checker for CSP. We use FDR to discover an attack upon the protocol, which allows an intruder to impersonate another agent. We adapt the protocol, and then use FDR to show that the new protocol is secure, at least for a small system. Finally we prove a result which tells us that if this small system is secure, then so is a system of arbitrary size.

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire
- La spécification « Bob accepte seulement lorsqu'il parle bien à Alice »

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire
- La spécification « Bob accepte seulement lorsqu'il parle bien à Alice »

- Traduction : processus CSP $\mapsto$ sorte d'automate $\mathscr{A}$
  langage $L(\mathscr{A})$ = comportements possibles du processus (*traces*)

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire
- La spécification « Bob accepte seulement lorsqu'il parle bien à Alice »

- Traduction : processus CSP $\mapsto$ sorte d'automate $\mathscr{A}$
  langage $L(\mathscr{A})$ = comportements possibles du processus (*traces*)
- $L(\mathscr{A}_{\text{système}}) \subseteq L(\mathscr{A}_{\text{spécification}})$?

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire
- La spécification « Bob accepte seulement lorsqu'il parle bien à Alice »

- Traduction : processus CSP $\mapsto$ sorte d'automate $\mathscr{A}$
  langage $L(\mathscr{A})$ = comportements possibles du processus (*traces*)
- $L(\mathscr{A}_{\text{système}}) \subseteq L(\mathscr{A}_{\text{spécification}})$? Non, le logiciel FDR trouve un contre-exemple
  = trace (suite d'événements) violant la spéc
- Ce contre-exemple indique une attaque

CSP = Communicating Sequential Processes = une algèbre de processus

Gavin Lowe modélise en CSP :

- Le système Alice + Bob + adversaire
- La spécification « Bob accepte seulement lorsqu'il parle bien à Alice »

- Traduction : processus CSP $\mapsto$ sorte d'automate $\mathscr{A}$
  langage $L(\mathscr{A})$ = comportements possibles du processus (*traces*)
- $L(\mathscr{A}_{\text{système}}) \subseteq L(\mathscr{A}_{\text{spécification}})$? Non, le logiciel FDR trouve un contre-exemple
  = trace (suite d'événements) violant la spéc
- Ce contre-exemple indique une attaque

Pour une version corrigée du protocole, $L(\mathscr{A}_{\text{système}}) \subseteq L(\mathscr{A}_{\text{spécification}})$
$\longrightarrow$ c'est sécurisé *relativement à la modélisation de l'adversaire*

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Unlinkability = l'adversaire ne peut pas distinguer entre ces situations :

- chaque passeport intervient dans $\leqslant 1$ session du protocole
- un passeport intervient dans $\geqslant 2$ sessions du protocole

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Unlinkability = l'adversaire ne peut pas distinguer entre ces situations :

- chaque passeport intervient dans $\leqslant 1$ session du protocole
- un passeport intervient dans $\geqslant 2$ sessions du protocole

$\longrightarrow$ modélisation par une *équivalence de processus*

- équivalence de traces : ok, c'est prouvé [Hirschi, Baelde & Delaune 2016]

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Unlinkability = l'adversaire ne peut pas distinguer entre ces situations :

- chaque passeport intervient dans $\leqslant 1$ session du protocole
- un passeport intervient dans $\geqslant 2$ sessions du protocole

$\longrightarrow$ modélisation par une *équivalence de processus*

- équivalence de traces : ok, c'est prouvé [Hirschi, Baelde & Delaune 2016]
- *bisimilarité* : **non ! [Filimonov, Horne, Mauw & Smith 2019]**

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Unlinkability = l'adversaire ne peut pas distinguer entre ces situations :

- chaque passeport intervient dans $\leqslant 1$ session du protocole
- un passeport intervient dans $\geqslant 2$ sessions du protocole

$\longrightarrow$ modélisation par une *équivalence de processus*

- équivalence de traces : ok, c'est prouvé [Hirschi, Baelde & Delaune 2016]
- *bisimilarité* : **non ! [Filimonov, Horne, Mauw & Smith 2019]**
  $\rightsquigarrow$ attaque exécutable en pratique (en circonstances limitées)

Basic Access Control (passeports électroniques)

Passeports contenant une puce RFID (communication sans fil)
+ une clé de chiffrement imprimée (lue par reconnaissance optique)

Idée : « en gros », un adversaire interceptant les ondes ne devrait pas pouvoir s'en servir pour « pister » un passeport (forme de privacy)

Unlinkability = l'adversaire ne peut pas distinguer entre ces situations :

- chaque passeport intervient dans $\leqslant 1$ session du protocole
- un passeport intervient dans $\geqslant 2$ sessions du protocole

$\longrightarrow$ modélisation par une *équivalence de processus* $\leftrightarrow$ pouvoir de l'adversaire

- équivalence de traces : ok, c'est prouvé [Hirschi, Baelde & Delaune 2016]
- *bisimilarité* : **non ! [Filimonov, Horne, Mauw & Smith 2019]**
  $\rightsquigarrow$ attaque exécutable en pratique (en circonstances limitées)

Unlinkability = ces situations sont bisimilaires

- $\mathfrak{A}$ = chaque passeport intervient dans $\leqslant 1$ session du protocole
- $\mathfrak{B}$ = un passeport intervient dans $\geqslant 2$ sessions du protocole

Unlinkability = ces situations sont bisimilaires

- $\mathfrak{A}$ = chaque passeport intervient dans $\leqslant 1$ session du protocole
- $\mathfrak{B}$ = un passeport intervient dans $\geqslant 2$ sessions du protocole

Théorème de Hennessy–Milner (à 1000 variantes)

2 systèmes sont bisimilaires $\Leftrightarrow$ ils *satisfont* les mêmes formules de *logique modale*

- Logique classique : $\mathbb{Z}$ *satisfait* $\forall x. \exists y. x + y = 0$, mais pas $\mathbb{N}$

Unlinkability = ces situations sont bisimilaires

- $\mathfrak{A}$ = chaque passeport intervient dans $\leqslant 1$ session du protocole
- $\mathfrak{B}$ = un passeport intervient dans $\geqslant 2$ sessions du protocole

Théorème de Hennessy–Milner (à 1000 variantes)

2 systèmes sont bisimilaires $\Leftrightarrow$ ils *satisfont* les mêmes formules de *logique modale*

- Logique classique : $\mathbb{Z}$ *satisfait* $\forall x. \exists y. x + y = 0$, mais pas $\mathbb{N}$
- Logique *modale* : nécessité et possibilité en philosophie analytique
  systèmes « évoluant dans le temps » en informatique

Unlinkability = ces situations sont bisimilaires

- $\mathfrak{A}$ = chaque passeport intervient dans $\leqslant 1$ session du protocole
- $\mathfrak{B}$ = un passeport intervient dans $\geqslant 2$ sessions du protocole

Théorème de Hennessy–Milner (à 1000 variantes)

2 systèmes sont bisimilaires $\Leftrightarrow$ ils *satisfont* les mêmes formules de *logique modale*

- Logique classique : $\mathbb{Z}$ *satisfait* $\forall x. \exists y. x + y = 0$, mais pas $\mathbb{N}$
- Logique *modale* : nécessité et possibilité en philosophie analytique
  systèmes « évoluant dans le temps » en informatique

$$\mathfrak{A} \text{ satisfait } F \text{ mais pas } \mathfrak{B} \quad \Longrightarrow \quad F \text{ décrit une attaque}$$

Ici : F trouvée par un calcul « à la main » par Filimonov et al.

- Privacy : veut-on vraiment des passeports électroniques ? biométriques ?

- Privacy : veut-on vraiment des passeports électroniques ? biométriques ?
- Quand est-il raisonnable d'avoir recours au vote électronique ?

Vote électronique : le logiciel Belenios s'ouvre au grand public

21 avril 2017

La plate-forme de vote électronique Belenios conçue par les équipes Pesto et Caramba, commune à Inria et au Loria (CNRS, Inria et Université de Lorraine) est désormais ouverte au public.

Chacun peut en effet organiser une élection qu'il soit ou non expert en informatique. Depuis septembre 2016, plusieurs élections de structures académiques se sont ainsi déroulées via ce logiciel et ont montré que d'autres personnes que les concepteurs pouvaient aisément mettre en place une élection grâce à cet outil.

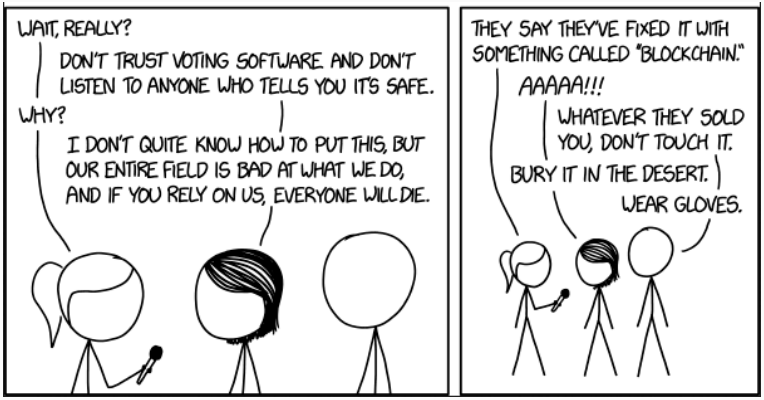
https://www.loria.fr/fr/2017/04/vote-electronique-le-logiciel-belenios-souvre-au-grand-public/

- Privacy : veut-on vraiment des passeports électroniques ? biométriques ?
- Quand est-il raisonnable d'avoir recours au vote électronique ?

> « **À quand l'utilisation du logiciel Belenios dans le milieu politique ?**
> *Le logiciel Belenios n'est pas adapté aux élections politiques car le vote électronique reste du vote par correspondance. Il n'y a pas de réel isoloir et il faut faire confiance à l'ordinateur. Or, un ordinateur peut être infecté. L'informaticien Laurent Grégoire avait d'ailleurs montré, lors des législatives de 2012 qu'il était possible de modifier le choix d'un électeur au moment de l'envoi du bulletin dans l'urne électronique pour les Français vivant à l'étranger. Le vote papier reste plus sûr que le vote électronique.* »

`https://www.loria.fr/fr/2017/04/vote-electronique-le-logiciel-belenios-souvre-au-grand-public/`

- Privacy : veut-on vraiment des passeports électroniques ? biométriques ?
- Quand est-il raisonnable d'avoir recours au vote électronique ?



https://xkcd.com/2030/

Avec la loi Narcotrafic, Signal quittera la France

Par Guillaume Belfiore, rédacteur en chef adjoint.
Publié le 27 février 2025 à 08h17

La présidente de la fondation Signal a affirmé publiquement qu'elle ne distribuerait plus la messagerie sécurisée au sein des pays ayant voté des lois visant à interdire ou amoindrir le chiffrement.

An Analysis of Signal's PQXDH

Karthikeyan Bhargavan (Cryspen), Charlie Jacomme (Inria), Franziskus Kiefer (Cryspen), Rolfe Schmidt (Signal)

October 20, 2023

Signal recently published a new, post-quantum secure, version of their X3DH protocol called PQXDH. As with any new cryptographic protocol, it is important to precisely analyse its security properties, especially for something as important as Signal.

In this blog post we give an overview of PQXDH and describe how we modeled its security using two formal verification tools (ProVerif and CryptoVerif). We detail some issues we found, and how we, a mix of Inria researchers, Cryspen researchers and Signal developers, worked together to improve the PQXDH specification.

ProVerif & CryptoVerif : prouveurs automatiques de sécurité des protocoles
(différence : modèle d'adversaire)

ProVerif & CryptoVerif : prouveurs automatiques de sécurité des protocoles
(différence : modèle d'adversaire)

Attention aux problématiques de fiabilité logicielle « générale »

« A claim was reported [...] that unlinkability does hold for ePassports that conservatively implement the BAC protocol. [...] A twist is that the original claim which we discovered to be flawed was based on a proof in ProVerif, that went through due to a bug, now resolved in Proverif. When the bug was corrected the old proof didn't go through, but no proof or counter-example was reported until [la faille dont on a parlé] »

— Horne & Mauw 2021

Implémentations vérifiées en F* (langage de programmation fonctionnel incorporant des preuves vérifiées par ordinateur ≠ preuves automatiques)

- primitives crypto (bibliothèque HACL*)
- protocole Signal (alternative à la lib officielle)
- ...

Implémentations vérifiées en F* (langage de programmation fonctionnel incorporant des preuves vérifiées par ordinateur ≠ preuves automatiques)

- primitives crypto (bibliothèque HACL*)
- protocole Signal (alternative à la lib officielle)
- ...
- protocole TLS : assure la sécurité des connexions HTTPS
  - composant logiciel particulièrement critique

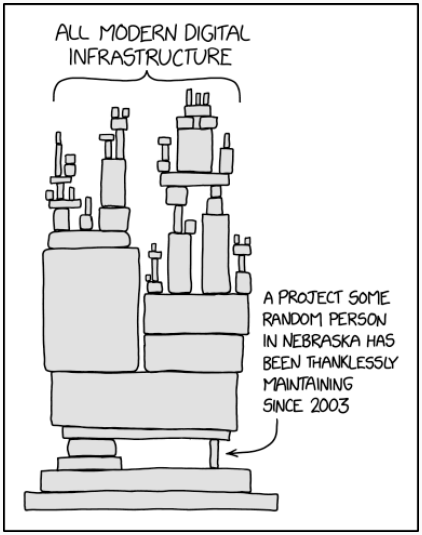
Implémentations vérifiées en F* (langage de programmation fonctionnel incorporant des preuves vérifiées par ordinateur ≠ preuves automatiques)

- primitives crypto (bibliothèque HACL*)
- protocole Signal (alternative à la lib officielle)
- ...
- protocole TLS : assure la sécurité des connexions HTTPS
  - composant logiciel particulièrement critique
  - vulnérabilités régulièrement trouvées dans les implémentations populaires, par ex. OpenSSL

Implémentations vérifiées en F* (langage de programmation fonctionnel incorporant des preuves vérifiées par ordinateur ≠ preuves automatiques)

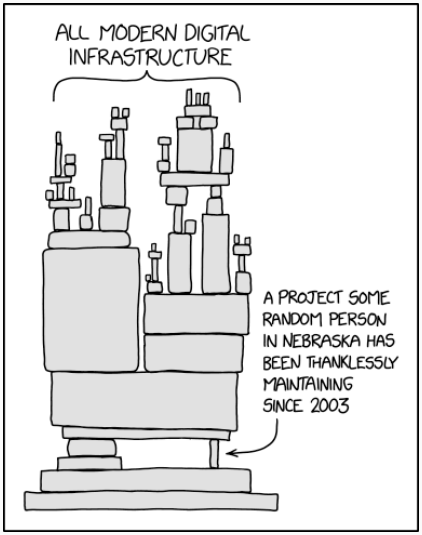
- primitives crypto (bibliothèque HACL*)
- protocole Signal (alternative à la lib officielle)
- …
- protocole TLS : assure la sécurité des connexions HTTPS
  - composant logiciel particulièrement critique
  - vulnérabilités régulièrement trouvées dans les implémentations populaires, par ex. OpenSSL particulièrement sévère : *Heartbleed* (2014)

- Vrai problème : projet maintenu par des bénévoles manquant de temps
  - « The total labor pool for OpenSSL maybe adds up to two fulltime developers. Think about it, OpenSSL only has two people to write, maintain, test, and review 500,000 lines of business critical code. »



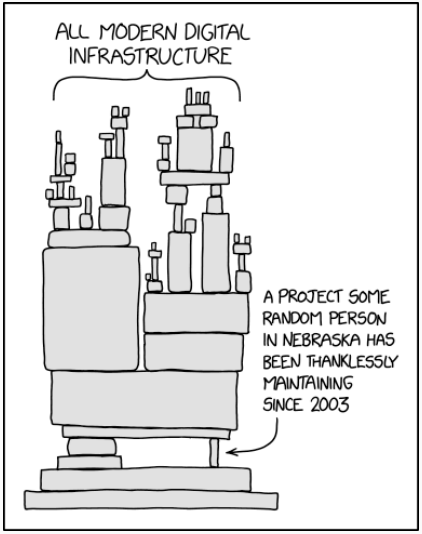
https://xkcd.com/2347/

- Vrai problème : projet maintenu par des bénévoles manquant de temps
  - « The total labor pool for OpenSSL maybe adds up to two fulltime developers. Think about it, OpenSSL only has two people to write, maintain, test, and review 500,000 lines of business critical code. »



https://xkcd.com/2347/

- Vrai problème : projet maintenu par des bénévoles manquant de temps
  - « The total labor pool for OpenSSL maybe adds up to two fulltime developers. Think about it, OpenSSL only has two people to write, maintain, test, and review 500,000 lines of business critical code. »
  - Plus récemment (2024) :
    burnout + *social engineering*
    $\Longrightarrow$ backdoor `xz`



https://xkcd.com/2347/

- Vrai problème : projet maintenu par des bénévoles manquant de temps
  - « The total labor pool for OpenSSL maybe adds up to two fulltime developers. Think about it, OpenSSL only has two people to write, maintain, test, and review 500,000 lines of business critical code. »
  - Plus récemment (2024) : burnout + *social engineering* $\Longrightarrow$ backdoor `xz`
- Heartbleed est un « bête » problème d'accès mémoire (*buffer over-read*)

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$
- invention des pointeurs `NULL` : « my billion dollar mistake » (Hoare)

(À titre de comparaison, 21/02/2025 : la Corée du Nord vole 1,5B$ en cryptomonnaies)

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$
- invention des pointeurs `NULL` : « my billion dollar mistake » (Hoare)

(À titre de comparaison, 21/02/2025 : la Corée du Nord vole 1,5B$ en cryptomonnaies)

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$
- invention des pointeurs `NULL` : « my billion dollar mistake » (Hoare)

Solution : **arrêter d'écrire des nouveaux programmes en C ou C++ !**

- Langages haut niveau : parfois inadaptés à la programmation système

(À titre de comparaison, 21/02/2025 : la Corée du Nord vole 1,5B$ en cryptomonnaies)

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$
- invention des pointeurs `NULL` : « my billion dollar mistake » (Hoare)

Solution : **arrêter d'écrire des nouveaux programmes en C ou C++ !**

- Langages haut niveau : parfois inadaptés à la programmation système
- **Rust :** langage « hype » conciliant sûreté et manipulation de la mémoire

Fortement inspiré de la recherche académique ; indirectement de la *logique linéaire* de J.-Y. Girard (chercheur émérite I2M) via la *correspondance preuves-programmes*

(À titre de comparaison, 21/02/2025 : la Corée du Nord vole 1,5B$ en cryptomonnaies)

Le coût des bugs de memory safety (à prendre avec des pincettes)

- Heartbleed : estimation initiale à 500M$
- invention des pointeurs `NULL` : « my billion dollar mistake » (Hoare)

Solution : **arrêter d'écrire des nouveaux programmes en C ou C++ !**

- Langages haut niveau : parfois inadaptés à la programmation système
- **Rust :** langage « hype » conciliant sûreté et manipulation de la mémoire

Fortement inspiré de la recherche académique ; indirectement de la *logique linéaire* de J.-Y. Girard (chercheur émérite I2M) via la *correspondance preuves-programmes*

systèmes de typage avancés = méthodes formelles « légères »