# De la confiance dans le monde OpenPGP

Damien Goutte-Gattat <dgouttegattat@incenp.org>



2016/11/27

**Résumé**

Cet article présente les notions de validité et de confiance dans l'univers OpenPGP, ainsi que les différents modèles de confiance disponibles dans les dernières versions de GnuPG.

## Table des matières



## 1. Introduction

La *confiance* est l'un des concepts à la fois les plus importants et les plus mal compris d'OpenPGP. Derrière ce terme peuvent se cacher en réalité deux notions bien distinctes, illustrées par les propositions suivantes :

- Alice est *confiante* que la clef 0xB4902A74 appartient à Bob ;
- Alice *fait confiance* à Bob quand celui-ci lui dit que la clef 0x4B493BB7 appartient à Charlie.

La première proposition fait référence à la *validité* de la clef 0xB4902A74, tandis que la seconde fait référence à la *confiance* proprement dite qu'Alice accorde à Bob.

Malheureusement, les auteurs et développeurs anglophones utilisent souvent le mot *trust* pour parler de la validité, et d'*ownertrust* pour parler de la confiance proprement dite, ce qui contribue probablement à la confusion entourant ces deux notions.

Dans cet article, nous verrons ce que recouvrent précisément ces deux notions, comment elles interagissent, et comment les manipuler avec GnuPG.

Rappelons au préalable qu'une clef publique OpenPGP est, au minimum, l'association d'une *clef brute*[1] et d'une ou plusieurs *identités* (*User ID*) représentant le *propriétaire* de la clef (c'est-à-dire, celui qui possède ou contrôle la clef privée correspondante).

# 2. La validité

On peut distinguer deux sortes de « validité » : la validite *d'une clef* et la validité *d'une identité*.

## 2.1. La validité d'une clef

Une clef OpenPGP peut être

- *révoquée*, si son propriétaire (ou un *révocateur désigné* mandaté par lui) a publié un *certificat de révocation* ;
- *expirée*, si la période de validité annoncée dans l'auto-signature la plus récente de la clef est dépassée.

Dans les deux cas, la clef dans son ensemble est *invalide*. Cette invalidité est absolue : elle est intrinsèque à la clef et ne dépend d'aucun autre facteur (notamment, elle ne dépend pas de ce que peut penser un utilisateur donné).

Si la clef n'est ni révoquée ni expirée, alors sa validité est celle de la plus valide de ses identités.

## 2.2. La validité d'une identité

La validité d'une identité d'une clef est une mesure de la certitude que l'on a que cette identité et cette clef sont bien associées, ou en d'autres termes, que la clef appartient bien au propriétaire désigné par l'identité.

Elle peut prendre plusieurs valeurs discrètes :

| | |
|---|---|
| Invalide | Je suis sûr que la clef *n*'appartient *pas* à son propriétaire proclamé. |
| Inconnue | Je n'ai aucune certitude quant à l'appartenance de la clef. |
| Marginalement valide | J'ai des raisons de penser que la clef appartient bien à qui elle prétend sans pour autant en être sûr. |
| Pleinement valide | Je suis sûr que la clef appartient bien à son propriétaire proclamé. |

Une identité n'est jamais valide ou invalide en elle-même : sa validité s'évalue toujours relativement à un utilisateur donné. Une même identité peut être pleinement valide pour

[1] Traduction libre du terme anglais *key material*, désignant au sens strict la partie purement mathématique d'une clef cryptographique (par exemple, le couple {module, exposant} pour une clef RSA). Concrètement, c'est le contenu d'un paquet OpenPGP de type Public-Key Packet [http://tools.ietf.org/html/rfc4880#section-5.5.1.1].

une personne et à validité inconnue pour une autre, si ces deux personnes ont des certitudes différentes quant à l'appartenance de la clef.

Une identité peut aussi être *révoquée* par le propriétaire de la clef. Dans ce cas, l'identité est inconditionnellement invalide.

Il ne faut pas confondre la révocation d'une *clef* avec la révocation d'une *identité* : une clef révoquée est complètement inutilisable, tandis qu'une clef dont une des identités est révoquée reste utilisable tant qu'au moins une autre de ses identités n'est pas invalide.

## 2.3. À quoi sert la validité ?

La validité répond à deux questions différentes selon que l'on veuille chiffrer un message ou vérifer une signature.

Lorsqu'on veut chiffrer un message, la validité définit si la clef du destinataire est utilisable, selon les règles suivantes :

- une clef expirée ou révoquée est inconditionnellement inutilisable[2] ;
- il est similairement impossible de chiffrer un message à destination d'une identité révoquée ou invalide ;
- une confirmation sera demandée avant de pouvoir chiffrer un message à destination d'une identité dont la validité est inconnue ;
- un avertissement sera affiché lors du chiffrement d'un message à destination d'une identité qui n'est que marginalement valide ;
- seule une identité pleinement valide associée à une clef non-expirée et non-révoquée est utilisable sans condition ni avertissement.

Lorsqu'on veut vérifier une signature, la validité de la clef signante définit le crédit que l'on peut accorder à la signature :

- si la clef est expirée, la signature n'est valable que si elle est antérieure à la date d'expiration ;
- si la clef est révoquée parce qu'elle a été compromise, ou sans qu'une raison explicite ne soit donnée, la signature n'est pas valable ;
- si la clef est révoquée mais que le certificat de révocation précise explicitement que la clef a été remplacée ou simplement retirée du service (c'est-à-dire, rien qui laisse supposer que la clef a été compromise), la signature est valable si elle est antérieure à la révocation ;
- si la clef ne contient aucune identité pleinement valide, la signature reste valable, mais un message rappellera qu'il existe une incertitude sur le propriétaire de la clef et donc sur le signataire du message.

## 2.4. Afficher la validité

La commande `--list-keys` de GnuPG permet d'afficher la validité des identités d'une clef :

```
alice$ gpg --list-keys bob
```

[2] Et GnuPG ne fournit aucun moyen de passer outre — même l'option `--expert`, qui autorise certaines actions normalement déconseillées, ne permet pas d'utiliser une clef expirée ou révoquée.

```
pub   rsa2048 2015-06-05 [SC]
      F336DF59EADFF278C40DBD7A21321A16B4902A74
uid           [  full  ] Robert <bob@example.com>
uid           [ unknown] Bob du 92 <bob92@provider.example>
sub   rsa2048 2015-06-05 [E]
```

Cette clef a deux identités associées : une pleinement valide (*full*), l'autre à validité inconnue (*unknown*).

Les versions de GnuPG antérieures à la 2.1 n'affichent pas, par défaut, la validité. Il faut ajouter l'option `--list-options show-uid-validity` sur la ligne de commande ou dans le fichier de configuration de GnuPG.

La validité est également affichée dans l'éditeur de clef :

```
alice$ gpg --edit-key bob
pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: marginal      validity: full
sub  rsa2048/E32EF7E899E238AD
     created: 2015-06-05  expires: never       usage: E
[  full  ] (1). Robert <bob@example.com>
[ unknown] (2)  Bob du 92 <bob92@provider.example>
```

En plus de la validité de chaque identité, l'éditeur de clef affiche aussi la validité de la clef dans son ensemble. Ici, la clef est complètement valide (*validity: full*), puisqu'elle n'est ni révoquée, ni expirée, et que l'une des deux identités associées est complètement valide.

Il n'y a pas de commande ou d'option pour *modifier* la validité. En effet — et c'est là le point central de cet article —, la validité d'une identité n'est pas décidée manuellement par l'utilisateur, mais déterminée automatiquement[3] via un modèle de confiance.

# 3. Les modèles de confiance

Un *modèle de confiance* est un ensemble de règles permettant de déterminer la validité d'une identité. Formellement, on peut l'assimiler à une fonction associant à chaque identité, une valeur de validité parmi celles listées plus haut (inconnue, marginale, complète, ou invalide).

Une caractéristique distinctive d'OpenPGP, par rapport notamment à X.509/SMIME, est l'absence de modèle de confiance imposé ou même seulement défini par le standard. Ce n'est pas un oubli de la part des auteurs, mais bien au contraire une volonté affichée d'être le plus « générique » et de laisser les implémenteurs libres de développer les modèles de confiance de leur choix.[4]

Les modèles de confiance proposés par GnuPG (sélectionnables via l'option `--trust-model`) sont les suivants :

- la confiance systématique (`--trust-model always`);
- la confiance directe (`--trust-model direct`);
- la toile de confiance, en version « simple » (`--trust-model classic`) et en version « étendue »[5] (`--trust-model pgp`);

[3] C'est pourquoi on parle aussi parfois de *calculated trust* pour désigner la validité.
[4] Cette volonté apparaît à plusieurs reprises dans le standard OpenPGP. En fait, ce standard peut être compris comme une « PKI en kit », un ensemble de briques permettant d'élaborer une infrastructure de clef publique, davantage que comme une infrastructure de clef publique prête à l'emploi.
[5] Les qualificatifs « simple » et « étendue » sont une invention de l'auteur, ils ne figurent pas dans les documentations de GnuPG. Nous verrons plus loin la différence entre les deux modèles.

- le modèle *Trust On First Use*,[6] utilisé seul (`--trust-model tofu`) ou conjointement avec la toile de confiance (`--trust-model tofu+pgp`).

Le choix du modèle de confiance est une décision *locale* : chaque utilisateur peut utiliser le modèle qu'il préfère, sans incidence sur l'interopérabilité (deux personnes utilisant des modèles de confiance différents peuvent toujours communiquer).

Je ne m'étendrai pas sur les deux premiers modèles, qui sont les plus simples à comprendre mais qui sont aussi les moins utiles, sauf dans certaines conditions particulières.

Dans le modèle de « confiance systématique », toutes les clefs sont toujours considérées pleinement valides.[7] Ce modèle est de fait à éviter absolument dans le cas général des communications à travers l'Internet, où les fausses clefs ne sont pas rares. Il peut toutefois avoir un intérêt dans un environnement strictement contrôlé, dans lequel on peut être sûr qu'il ne circule que des clefs authentiques (mais a-t-on vraiment besoin d'OpenPGP dans un tel environnement ?).

À l'opposé, le modèle de « confiance directe » confie à l'utilisateur le soin de décider lui-même, directement, de la validité de chaque clef. C'est donc, par définition, une exception au principe énoncé plus haut selon lequel la validité est calculée automatiquement par le modèle de confiance et non assignée manuellement par l'utilisateur.

Ce modèle peut sembler attrayant, en particulier pour ceux que la complexité de la toile de confiance effraie. Néanmoins, les modèles plus récents de type TOFU, décrits plus loin, sont une meilleure alternative.

Par défaut, GnuPG utilise le modèle `pgp`, la toile de confiance étendue.

# 4. La toile de confiance

C'est *le* modèle de confiance traditionnellement associé à OpenPGP, au point d'en ignorer souvent que ce n'est qu'*un* des modèles possibles.

Il est, hélas, assez souvent mal compris et donc trop souvent mal utilisé.

La bonne compréhension de ce modèle nécessite d'introduire deux notions supplémentaires, qui étaient inutiles jusqu'à présent.

## 4.1. Notion de certification

Une *certification* est une signature sur le couple {clef brute, identité} (on parlera, très souvent, de *signature de clef*), par laquelle le signataire atteste (« certifie ») que cette clef brute et cette identité sont associées.

Formellement, l'ensemble formé d'une clef brute, d'une identité, et d'au moins une certification constitue un *certificat OpenPGP*. Ce terme est néanmoins rarement employé, l'usage ayant consacré l'expression *clef publique OpenPGP* à la place (entraînant hélas un risque de confusion avec le concept mathématique de clef publique, que je désigne dans ce document par *clef brute* justement pour éviter toute ambiguïté.).

### 4.1.1. Attributs de certification

Une certification peut être qualifiée par plusieurs attributs, dont certains peuvent affecter l'interprétation qui doit être faite de cette certification.

[6] Introduit dans GnuPG 2.1.10, sorti en décembre 2015.
[7] Hors le cas des clefs *expirées* ou *révoquées*, qui sont toujours inconditionnellement invalides quel que soit le modèle de confiance.

Une certification est toujours qualifiée par un *niveau* (*certification level*) qui traduit la rigueur avec laquelle le signataire a vérifié l'identité du propriétaire de la clef. Le niveau 0 (certification « générique ») correspond à une absence d'engagement : en certifiant à ce niveau, le signataire ne donne délibérément aucune information. Le niveau 1 correspond en principe à une absence de vérification : en certifiant à ce niveau, le signataire annonce qu'il n'a pas particulièrement vérifié l'identité du propriétaire. En certifiant au niveau 2 ou au niveau 3, le signataire annonce qu'il a procédé à des vérifications plus rigoureuses.

Malheureusement, le standard OpenPGP ne définit pas précisément ce que peuvent être les vérifications pour chaque niveau. Chaque utilisateur peut ainsi donner à chaque niveau la signification de son choix, ce qui en pratique fait perdre beaucoup d'intérêt à la notion même de niveau de certification puisque deux utilisateurs peuvent avoir une opinion différente de ce qu'est une certification « rigoureuse ».

À titre d'exemple, je définis la limite entre les certifications de niveau 2 et 3 comme suit : si j'ai rencontré le propriétaire de la clef en personne, je certifie au niveau 2 ; s'il m'a en plus présenté une pièce d'identité officielle, je certifie au niveau 3. Mais ce n'est que *ma* politique : une certification de niveau 3 émise par un autre utilisateur peut avoir une signification différente.

En pratique, la plupart des certifications sont de niveau 0. C'est le niveau de certification par défaut avec GnuPG. Compte tenu de l'absence de signification universellement reconnue des différents niveaux de confiance, il est tout-à-fait raisonnable de s'en tenir à ce niveau 0 et d'ignorer jusqu'à l'existence même des niveaux supérieurs.

Le signataire peut ajouter à sa certification une URL pointant vers un document supposé expliquer sa politique de signature (il peut notamment décrire la signification qu'il donne aux différents niveaux de certification). Cet attribut est purement informatif et à destination de l'utilisateur : GnuPG n'en fait aucun usage lui-même.

Une certification peut être marquée comme *locale*. Une telle certification n'est valable que dans le trousseau de celui qui l'a émise et sera automatiquement omise lorsque la clef sera exportée.

Enfin, une certification peut être marquée comme *irrévocable*. Normalement, toute certification peut être annulée (*révoquée*) *a posteriori* par son émetteur, simplement en publiant une signature de révocation (qui concrètement prend la même forme qu'une certification, c'est-à-dire une signature sur le couple {clef brute, identité}, mais signifie que toute certification antérieure sur le même couple, émise par la même clef, doit être considérée comme nulle). Si la certification initiale est marquée irrévocable, alors toute signature de révocation ultérieure sera ignorée.

### 4.1.2. Notion d'auto-certification

Chaque identité porte toujours au moins une *auto-certification*, c'est-à-dire une certification émise par la propre clef brute à laquelle l'identité est associée.

Si elle est sans valeur, comme on le verra, lors du calcul de la validité, elle permet surtout au propriétaire de la clef d'exprimer certaines préférences par l'intermédiaire d'attributs spécifiques aux auto-certifications :

- la durée de validité de la clef ;
- un éventuel *révocateur désigné*, sous la forme de l'empreinte d'une clef tierce habilitée à émettre des certificats de révocation pour cette clef ;
- les algorithmes de chiffrement, de condensation et de compression utilisables pour communiquer avec le propriétaire de cette clef, par ordre de préférence.

Une fois émise, une (auto-)certification n'est pas modifiable. Pour changer l'un des attributs ci-dessus (par exemple pour repousser la date d'expiration de la clef, ou pour mettre à jour les algorithmes préférés), il faut émettre une nouvelle auto-certification, qui dans les faits annulera toute auto-certification antérieure. Seule l'auto-certification la plus récente est prise en compte quand il y en a plusieurs.

## 4.2. Notion de confiance

La *confiance* proprement dite (*ownertrust*) est une valeur associée par l'utilisateur à une clef publique qui définit le crédit à accorder aux certifications émises par cette clef.

Comme pour la validité définie plus haut, la confiance peut prendre plusieurs valeurs discrètes :

| | |
|---|---|
| aucune confiance | Ne jamais accorder aucun crédit aux certifications émises par cette clef. |
| confiance inconnue ou indéterminée | Ignorer les certifications émises par cette clef. |
| confiance marginale | Tenir compte des certifications seulement dans une certaine mesure (voir plus loin). |
| confiance complète | Accorder toute leur valeur aux certifications. |
| Confiance ultime | Valeur spéciale normalement réservée aux clefs « locales », c'est-à-dire les clefs dont la partie privée est disponible. |

Dans le modèle de la toile de confiance simple, la confiance est toujours *explicitement assignée par l'utilisateur*. Chaque clef ajoutée au trousseau se voit assignée par défaut une confiance inconnue, et ce jusqu'à ce que l'utilisateur décide de la changer explicitement.

L'assignation de la confiance se fait via la commande **trust** de l'éditeur de clefs de GnuPG.

Dans l'exemple que nous avons vu plus haut, la clef de Bob, telle qu'elle figure dans le trousseau d'Alice, a une confiance marginale (*trust: marginal*).

## 4.3. Règles de la toile de confiance

On peut maintenant poser le principe de fonctionnement de la toile de confiance, telle qu'elle est implémentée par GnuPG.

Pour déterminer la validité d'une identité, on commence par retirer toutes les certifications *inexploitables* portées par cette identité. Précisément, on retire les certifications :

- émises par des clefs inconnues, c'est-à-dire qui ne figurent pas dans le trousseau local ;
- émises par des clefs invalides ou à validité inconnue (ce qui élimine entre autres l'auto-certification émise par la clef même que l'on cherche à valider) ;
- émises par des clefs révoquées ;
- dont le niveau de certification est supérieur à zéro mais inférieur au paramètre `--min-cert-level` (qui vaut 2 par défaut, ce qui signifie que les certifications de niveau 1 sont ignorées) ;
- révoquées par leur signataire, sauf si la certification initiale était marquée *non-révocable*.

Pour chaque certification restante, on regarde ensuite la *confiance* assignée à la clef émettrice. S'il y a…

- au moins 1 certification émise par une clef à confiance ultime, l'identité est complètement valide ;
- au moins $n$ certifications émises par des clefs à confiance complète (avec $n = 1$ par défaut, modifiable avec l'option `--completes-needed`), l'identité est complètement valide ;
- au moins $m$ certifications émises par des clefs à confiance marginale (avec $m = 3$ par défaut, modifiable avec l'option `--marginals-needed`), l'identité est complètement valide ;
- entre 1 et $n - 1$ certification(s) émise(s) par des clefs à confiance complète, ou entre 1 et $m - 1$ certification(s) émises par des clefs à confiance marginale, l'identité est marginalement valide ;
- aucune certification émise par une clef à confiance au moins marginale, l'identité est à validité inconnue.

## 4.4. Un exemple

Considérons un instant la clef d'Alice :

```
alice$ gpg --list-keys alice
pub   rsa4096 2015-06-05 [SC] [expires: 2018-06-04]
      318D1F0158F237EB64797C0C5C5CE0D82EADF7D4
uid           [ultimate] Alice <alice@example.org>
```

S'agissant d'une clef dont la partie privée est disponible (puisque nous sommes sur le système d'Alice), elle est intrinsèquement ultimement valide et de confiance.

Dans toutes les captures d'écrans qui suivent, pour gagner à la fois en place et en clarté, les *sous-clefs* seront systématiquement omises.

Maintenant, imaginons qu'Alice vient juste d'obtenir la clef publique de Bob. Elle l'importe dans son trousseau et y jette un œil :

```
alice$ gpg --import bob.asc
gpg: key 21321A16B4902A74: public key "Robert <bob@example.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
alice$ gpg --list-keys --with-sig-check bob
gpg: 2 good signatures
pub   rsa2048 2015-06-05 [SC]
      F336DF59EADFF278C40DBD7A21321A16B4902A74
uid           [ unknown] Robert <bob@example.com>
sig!3        21321A16B4902A74 2015-06-05  Robert <bob@example.com>
uid           [ unknown] Bob du 92 <bob92@provider.example>
sig!3        21321A16B4902A74 2015-10-07  Robert <bob@example.com>
```

Cette clef a deux identités (`Robert <bob@example.com>` et `Bob du 92 <bob92@provider.example>`), chacune porteuse d'une auto-certification (émise par la clef 0xB4902A74, c'est-à-dire *cette* clef). En l'absence d'autres certifications, ces deux identités sont de validité inconnue.

Imaginons à présent qu'Alice est personnellement certaine qu'il s'agit bien de la clef de Bob (par exemple, Bob lui a confirmé l'empreinte de vive voix). Elle va donc certifier (signer) sa clef :

```
alice$ gpg --edit-key bob
```

```
pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: unknown       validity: unknown
[ unknown] (1). Robert <bob@example.com>
[ unknown] (2). Bob du 92 <bob92@provider.example>
```

Alice sélectionne la première identité, correspondant à la seule adresse de Bob dont elle soit sûre, puis la certifie :

```
gpg> 1
pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: unknown       validity: unknown
[ unknown] (1)* Robert <bob@example.com>
[ unknown] (2)  Bob du 92 <bob92@provider.example>

gpg> sign

pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: unknown       validity: unknown
 Primary key fingerprint: F336 DF59 EADF F278 C40D  BD7A 2132 1A16 B490 2A74

     Robert <bob@example.com>

Are you sure that you want to sign this key with your
key "Alice <alice@example.org>" (5C5CE0D82EADF7D4)

Really sign? (y/N) y

gpg> save
```

Ici, en utilisant la commande **sign**, Alice a opté pour une signature « normale », sans fioritures : niveau de certification 0, exportable, révocable.

Rejettons à présent un œil comme précédemment sur la clef de Bob :

```
alice$ gpg --list-keys --with-sig-check bob
gpg: 3 good signatures
pub   rsa2048 2015-06-05 [SC]
      F336DF59EADFF278C40DBD7A21321A16B4902A74
uid           [  full  ] Robert <bob@example.com>
sig!3        21321A16B4902A74 2015-06-05  Robert <bob@example.com>
sig!         5C5CE0D82EADF7D4 2016-11-26  Alice <alice@example.org>
uid           [ unknown] Bob du 92 <bob92@provider.example>
sig!3        21321A16B4902A74 2015-10-07  Robert <bob@example.com>
```

Si rien n'a changé pour l'identité « Bob du 92 » (qui n'a toujours que sa seule auto-certification et est donc toujours de validité inconnue), l'identité de « Robert », elle, porte désormais la certification d'Alice. La clef d'Alice étant de confiance ultime, cette identité est donc pleinement valide, en application des règles de la toile de confiance vues précédemment.

Remarquez qu'Alice n'a jamais directement modifié la validité de la clef de Bob. Je me permets d'insister parce que c'est la notion centrale de cet article : la validité est toujours *calculée* automatiquement par GnuPG, jamais directement assignée par l'utilisateur. *C'est en jouant sur la confiance que l'on affecte la validité*, selon des modalités qui varient selon le modèle de confiance utilisé.

Poursuivons l'exemple. Alice obtient à présent la clef de Charlie. Comme précédemment, elle l'importe et l'examine :

```
alice$ gpg --import charlie.asc
gpg: key 3800CBA74B493BB7: public key "Charlie <charlie@example.net>" imported
gpg: Total number processed: 1
gpg:               imported: 1
alice$ gpg --list-keys --with-sig-check charlie
gpg: 2 good signatures
pub   rsa2048 2015-06-05 [SC]
      3172310F9C0C11AEA1B057433800CBA74B493BB7
uid           [ unknown] Charlie <charlie@example.net>
sig!3        3800CBA74B493BB7 2015-06-05  Charlie <charlie@example.net>
sig!         21321A16B4902A74 2016-11-26  Robert <bob@example.com>
```

La clef de Charlie n'a qu'une identité, laquelle porte deux certifications : l'auto-certification de rigueur, et une certification émise par Bob. Pourquoi cette identité est-elle considérée « de validité inconnue » ?

Parce que Alice n'a jamais assigné de valeur de confiance à la clef de Bob ! GnuPG lui a donc attribué, par défaut, une confiance inconnue, qui dans le modèle de la toile de confiance ne confère aucun crédit aux certifications émises par cette clef. Allons éditer la clef de Bob pour changer ça :

```
alice$ gpg --edit-key bob
pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: unknown       validity: full
sub  rsa2048/E32EF7E899E238AD
     created: 2015-06-05  expires: never       usage: E
[  full  ] (1). Robert <bob@example.com>
[ unknown] (2)  Bob du 92 <bob92@provider.example>

gpg> trust

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 3

pub  rsa2048/21321A16B4902A74
     created: 2015-06-05  expires: never       usage: SC
     trust: marginal      validity: full
sub  rsa2048/E32EF7E899E238AD
     created: 2015-06-05  expires: never       usage: E
[  full  ] (1). Robert <bob@example.com>
[ unknown] (1)  Bob du 92 <bob92@provider.example>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg> quit
```

Alice a donc assigné une confiance marginale à la clef de Bob. Voyons ce que cela change sur la validité de la clef de Charlie :

```
alice$ gpg --list-keys --with-sig-check charlie
gpg: 2 good signatures
pub   rsa2048 2015-06-05 [SC]
      3172310F9C0C11AEA1B057433800CBA74B493BB7
uid           [marginal] Charlie <charlie@example.net>
sig!3        3800CBA74B493BB7 2015-06-05  Charlie <charlie@example.net>
sig!         21321A16B4902A74 2016-11-26  Robert <bob@example.com>
```

Avec une certification émise par une clef à confiance marginale, et avec les paramètres par défaut de la toile de confiance, la clef de Charlie est désormais marginalement valide.

### 4.5. Chaîne de certification et profondeur

Le dernier exemple permet d'illustrer le concept de *chaîne de certification*, dont nous avons besoin pour expliquer une dernière règle du modèle de la toile de confiance, celle de la *profondeur maximale* de la chaîne de certification.

Dans cet exemple, Alice a certifié la clef de Bob, qui a lui-même certifié la clef de Charlie. Cette dernière se trouve ainsi à l'extrémité d'une chaîne remontant jusqu'à la clef d'Alice par l'intermédiaire de celle de Bob.

Comme nous regardons tout ceci depuis le point de vue d'Alice, sa clef est le point de départ de la chaîne. Elle est associée à une *profondeur* nulle. La clef de Bob, directement certifiée Alice, a une profondeur de 1 ; celle de Charlie, certifiée par Bob, a une profondeur de 2. Si Charlie certifiait la clef de David (et en supposant qu'Alice attribue une confiance explicite à la clef de Charlie, ce qu'elle n'a pas encore fait dans notre exemple), celle-ci aurait une profondeur de 3, et ainsi de suite.

La règle de la profondeur maximale dit simplement que la profondeur associée à une clef ne peut pas dépasser 5 (valeur par défaut, modifiable par l'option `--max-cert-depth`), ou autrement dit, qu'une chaîne de certification ne peut pas compter plus de 6 maillons.

Ainsi, si David certifiait la clef de Fanny qui elle-même certifiait la clef de Gordon qui lui-même certifiait la clef de Helen, cette dernière serait quoi qu'il arrive trop éloignée de la clef d'Alice (profondeur de 6) pour être considérée valide, même si Alice faisait explicitement confiance à David, Fanny et Gordon.

Dans les faits, il est extrêmement rare qu'une clef ne soit pas validée à cause de cette profondeur maximale. Les chaînes de certifications s'arrêtent généralement avant de heurter cette limite, par défaut de confiance explicite (plus on s'éloigne d'Alice, moins il y a de chance qu'elle connaisse suffisamment bien les personnes impliquées pour pouvoir assigner un niveau de confiance à leurs clefs).

## 5. La toile de confiance étendue

Jusque là, ce que nous avons vu était la toile de confiance *simple* (si, si…). Voyons à présent ce que recouvre la toile de confiance *étendue*.

La toile de confiance étendue fonctionne comme la toile de confiance simple, mais prend en compte un type particulier de certifications qu'on appelle les *trust signatures*.

La toile de confiance étendue est le modèle de confiance par défaut de GnuPG. Toutefois, si vous n'utilisez pas (et n'avez jamais émis) de *trust signatures*, alors dans les faits vous n'utilisez que la toile de confiance simple… comme la quasi-totalité des utilisateurs de GnuPG.

### 5.1. Notion de *trust signature*

Une *trust signature* est semblable à une certification simple comme décrit plus haut (c'est-à-dire une signature sur un couple {clef brute, identité}), mais avec deux paramètres supplémentaires :

- une *profondeur* $n$, indiquant que la clef portant cette *trust signature* est habilitée à émettre elle-même des *trust signatures* de profondeur $n - 1$ (une profondeur de zéro rend la *trust signature* strictement équivalente à une certification simple) ;

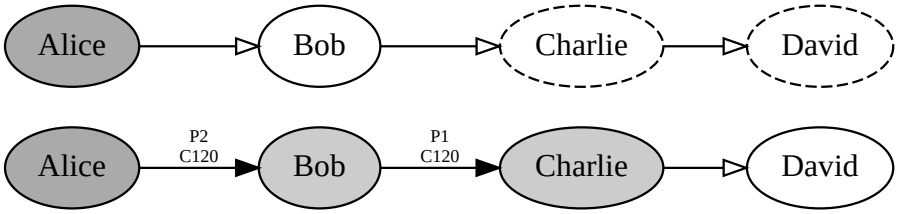
- une *valeur de confiance* à assigner à la clef portant cette *trust signature* ; en principe cette valeur peut s'étendre de zéro à 255, mais en pratique il n'y a que deux possibilités : toute valeur inférieure à 120 est interprétée comme assignant une confiance marginale, et toute valeur supérieure ou égale à 120 est interprétée comme assignant une confiance complète.

C'est dans ce second paramètre que réside la différence entre la toile de confiance simple et la toile de confiance étendue : les certifications de la toile de confiance simple ne servent qu'à déterminer la *validité* d'une identité, l'assignation de la confiance étant du seul ressort de l'utilisateur ; les *trust signatures* de la toile de confiance étendue déterminent à la fois la validité d'une identité *et* la confiance assignée à la clef associée.

## 5.2. Impact des *trust signatures*

Pour illustrer l'impact des *trust signatures* sur la toile de confiance, reprenons à nouveau l'exemple d'une chaîne de certification allant de Alice à David en passant par Bob et Charlie.

### Figure 1. Chaînes de certification sans et avec *trust signatures*




Deux exemples de chaînes de certification. Chaque cercle représente une clef ; un tracé plein dénote une clef valide tandis qu'un tracé en pointillé dénote une clef à validité inconnue ; un fond grisé dénote une clef de confiance tandis qu'un fond blanc dénote une clef à confiance inconnue. Les flèches à pointe vide représentent des certifications simples, et les flèches à pointe pleine représentent des *trust signatures*, dont la profondeur (P) et la valeur de confiance (C) sont indiqués.

En absence de *trust signatures*, du point de vue d'Alice seule la clef de Bob est valide (puisqu'elle est certifiée par une clef à confiance ultime, la sienne). Même si Bob a certifié la clef de Charlie, cette dernière restera à validité inconnue tant que Alice n'aura pas explicitement exprimé sa confiance envers Bob. (Il en va de même, *a fortiori*, pour la clef de David.)

Notez que chez lui, Bob fait peut-être confiance à Charlie, auquel cas la clef de David serait valide à ses yeux. Mais c'est sans intérêt pour Alice, qui n'a de toute façon aucun moyen de savoir quelle confiance Bob accorde à Charlie. Bob est le seul à savoir ça (même Charlie l'ignore) : dans la toile de confiance simple, la confiance est toujours une valeur *locale*, jamais partagée avec les autres membres de la toile.

Imaginons à présent que Alice a certifié la clef de Bob, non avec une certification simple, mais avec une *trust signature* de profondeur 2 et de confiance 120 : la clef de Bob est alors non seulement valide, mais se voit aussi *automatiquement* attribué une confiance complète. Si, de son côté, Bob a certifié la clef de Charlie avec une *trust signature* de profondeur 1 et de confiance 120, alors pour Alice, la clef de Charlie est valide et se voit aussi assigné *automatiquement* une confiance complète. Du coup, la clef de David, certifiée par Charlie, devient valide aux yeux d'Alice, même si elle ne s'est jamais prononcée elle-même sur la confiance à accorder à Charlie.

Notez que si Alice avait certifié la clef de Bob avec une certification simple, ou avec une *trust signature* de profondeur 1, la *trust signature* de Bob sur la clef de Charlie aurait été

ignorée, ou plus exactement, traitée comme une certification simple, et aucune confiance n'aurait été automatiquement assignée à la clef de Charlie.

Cela signifie qu'il n'y a aucun risque d'utiliser la toile de confiance étendue par erreur. Même si tous les correspondants d'Alice émettaient des *trust signatures* à tout-va, celles-ci ne seront prises en compte par Alice que si elle décide elle-même d'entrer à son tour dans le jeu de la toile de confiance étendue, en émettant des *trust signatures* sur les clefs de ses correspondants. Si elle souhaite au contraire continuer à décider elle-même, seule, de la confiance à accorder à chacun, il lui suffit de ne jamais émettre que des certifications simples… ce que fait déjà GnuPG par défaut.

En pratique, *personne* n'émet de *trust signatures*. Sérieusement. En plus de dix ans d'utilisation de GnuPG, je n'en ai jamais vu une seule dans la nature. C'est pourquoi, bien que la toile de confiance étendue soit le modèle de confiance par défaut de GnuPG, dans les faits tout le monde utilise la toile de confiance *simple* (où les *trust signatures* peuvent exister mais sont traitées comme des certifications simples : leur valeur de confiance est ignorée, et elles ne servent qu'à calculer la *validité*).

### 5.3. Limitation du champ des *trust signatures*

Outre la profondeur et la valeur de confiance, une *trust signature* peut se voir adjoindre un troisième paramètre : une expression rationnelle qui limite les identités pour lesquelles la clef portant cette *trust signature* est autorisée à émettre elle-même des *trust signatures*.

Par exemple, imaginons que Alice certifie la clef de Bob avec une *trust signature* associée à l'expression rationnelle `<[^>]+@example.net>$`. Dans ce cas, les *trust signatures* émises par Bob ne seront considérées comme légitimes que si elles sont appliquées sur des identités dont l'adresse e-mail est dans le domaine `example.net`. Si Bob certifie une identité `Charlie <charlie@nimportequoi.example>`, sa certification sera ignorée.

Ce mécanisme est analogue à l'extension *Name Constraints* du monde X.509, qui limite les domaines pour lesquels une autorité de certification est habilitée à émettre des certificats.

L'implémentation de cette fonctionnalité dans GnuPG est plus restrictive que ne le permet le RFC 4880. GnuPG ne permet que d'exprimer une contrainte sur le domaine de l'adresse e-mail (comme dans l'exemple ci-dessus) ; il n'est pas possible de spécifier librement une expression rationnelle arbitraire.

## 6. Le modèle *Trust On First Use*

### 6.1. Pourquoi un nouveau modèle de confiance ?

Le modèle de confiance *Trust On First Use* (TOFU) a été introduit dans GnuPG 2.1.10, en décembre 2015. C'est la première introduction d'un nouveau modèle de confiance depuis les débuts de GnuPG.

La raison d'être de ce modèle part d'un constat qui ne surprendra guère les utilisateurs de GnuPG (ou de toute autre implémentation d'OpenPGP) : la toile de confiance est trop complexe à appréhender. Beaucoup d'utilisateurs (y compris parfois des utilisateurs de longue date) ne la comprennent pas réellement, ou ne comprennent pas toutes ses subtilités, et en conséquence ne l'utilisent pas correctement. Et même pour les connaisseurs, elle est souvent trop pénible à utiliser.

Le modèle de la « confiance à la première utilisation » est, sur le papier, moins « puissant » que la toile de confiance ; il est vulnérable, par définition, à une tentative d'usurpation

lors du premier contact. *Mais* il est plus facile à appréhender et à utiliser. Il offre ce que certains auteurs anglophones appellent de la *better-than-nothing security*.

L'introduction de ce modèle de confiance (qui n'est pas encore le modèle par défaut — c'est toujours la toile de confiance étendue, pour l'instant — mais qui est appelé à le devenir) s'inscrit dans le cadre d'une ré-orientation des objectifs de GnuPG dans l'ère post-Snowden. Depuis ses débuts, GnuPG a été conçu pour répondre aux besoins d'utilisateurs faisant face à un niveau de menace élevé — le genre d'utilisateurs pour qui la difficulté d'accès du logiciel n'était pas forcément rédhibitoire, ou était un prix à payer acceptable. Désormais, à l'ère de la surveillance de masse, les développeurs de GnuPG considèrent que GnuPG doit être facile d'accès par défaut, pour les utilisateurs « ordinaires » souhaitant échapper à cette surveillance.[8]

L'idée n'est absolument pas de « castrer » GnuPG, de le rendre inutile à ceux qui font face à un niveau de menace élevé — notamment, ceux qui font face à des attaques ciblées et non pas à la simple surveillance de masse. C'est juste que, *par défaut*, GnuPG ne sera pas configuré pour eux.

## 6.2. Principe

GnuPG conserve une trace de toutes les associations {adresse e-mail, clef} qu'il rencontre (une telle association est appelée un *binding* dans la description du modèle), et associe à chacune d'elles une *politique TOFU*, parmi les cinq suivantes : `unknown`, `auto`, `good`, `bad`, et `ask`.

Chaque politique (à part la politique `ask`) correspond à une valeur possible de validité : `unknown` correspond à une validité inconnue, `auto` à une validité marginale, `good` à une validité complète, et `bad` correspond à une invalidité. Lorsque le modèle est interrogé pour déterminer la validité d'une identité, il renvoie la valeur de validité correspondant à la politique associé au *binding* concerné. (Si la politique est `ask`, GnuPG demande en direct à l'utilisateur ce qu'il doit faire avec cette clef.)

À la réception d'un premier message signé par `bob@example.com`, le *binding* {`bob@example.com`, 0xB4902A74} est ajouté à la base de données TOFU, associé à la politique par défaut qui est `auto`. Cela confère automatiquement une validité marginale à cette clef.

L'utilisateur peut à tout moment changer la politique associée à un *binding*. Par exemple, si Alice est sûre qu'il s'agit bien de la clef de Bob, elle peut lui assigner la politique `good`, conférant à sa clef une validité complète.[9]

À la réception d'un nouveau message de `bob@example.com`, GnuPG vérifie si la clef utilisée est la même que celle figurant dans le *binding* précédemment enregistré. Si c'est bien le cas, il n'y a pas de conflit, et GnuPG affiche que le message provient d'une clef marginalement valide. Si la clef est différente, l'utilisateur est alerté de l'existence d'un conflit.

## 6.3. Choix de la politique TOFU par défaut

La politique TOFU par défaut est celle appliquée implicitement lorsqu'un *binding* est ajouté à la base de données TOFU. Par défaut, cette politique par défaut est `auto` (comme on l'a vu ci-dessus), mais elle est modifiable avec l'option `--tofu-default-policy`. Et le choix de cette politique par défaut change profondément le comportement du modèle.

[8] Werner Koch, développeur principal de GnuPG, a exprimé cette idée lors de la rencontre annuelle des développeurs Debian de 2015 (DebConf 2015). L'enregistrement de son intervention est disponible [http://meetings-archive.debian.net/pub/debian-meetings/2015/debconf15/GnuPG_Past_Present_and_Future.webm], de même que ses diapositives [https://gnupg.org/ftp/blurbs/debconf15_gnupg-past-present-future.pdf].

[9] Néanmoins, dans le modèle TOFU, la distinction entre la validité complète et la validité marginale est moins pertinente que dans le modèle de la toile de confiance, et Alice pourrait simplement laisser la politique par défaut.

Trois politiques peuvent être définies comme la politique TOFU par défaut : `good`, `unknown`, et `auto`.

Avec `good` comme politique par défaut, on choisit un modèle « optimiste », dans lequel toute clef nouvellement rencontrée est implicitement considérée comme complètement valide. C'est en quelque sorte, le « vrai TOFU », le TOFU proprement dit.

À l'inverse, avec `unknown` comme politique par défaut, on choisit un modèle « pessimiste » voire « paranoïaque ». Il n'y a aucune validité implicite, toute clef nouvellement rencontrée est à validité inconnue et le reste jusqu'à ce que l'utilisateur assigne explicitement une politique `good` ou `auto`. C'est ce qui se rapproche le plus du modèle de confiance « directe » que j'avais rapidement abordé en présentant les différents modèles de confiance, avec en plus la détection des conflits.

Enfin, avec `auto` comme politique par défaut, on choisit un modèle intermédiaire, dans lequel toute clef nouvellement rencontrée est considérée comme marginalement valide.

### 6.4. Le modèle TOFU+PGP

Le modèle TOFU+PGP, comme son nom l'indique, est une combinaison du modèle de la toile de confiance étendue et du modèle TOFU.

Pour déterminer la validité d'une identité, les deux modèles sont interrogés successivement ; l'identité est valide si ① elle est valide dans au moins un des deux modèles, et ② elle n'est invalide dans aucun des deux modèles.

Ce modèle est particulièrement intéressant lorsque la politique TOFU par défaut est `unknown`. Dans ce cas, seule la toile de confiance assigne des valeurs de validité positives (validité marginale ou complète), et le modèle TOFU ne sert alors qu'à détecter les conflits.

## A. À propos de ce document