TD réseau – Sécurité/crypto - octobre 2005

Cryptographie, contrôle d'authenticité et vérification d'intégrité

Dans le domaine des réseaux informatiques, deux démarches complémentaires permettent d'assurer la confidentialité des échanges entre deux acteurs. D'une part la sécurité propre des transmissions vise à empêcher un tiers d'exploiter des failles dans le logiciel ou le matériel et de s'approprier des données confidentielles. D'autre part, la cryptographie et la vérification d'intégrité permettent dans le cas ou une interception est rendue possible par une faille du réseau, d'assurer une sécurité des données par encryptage du message.
Nous utiliserons le cryptosystème GnuPG qui à l'avantage d'être libre d'utilisation et de ne reposer sur aucun algorithme breveté (RSA, le plus connu était breveté jusqu'il y à peu). Md5sum va nous permettre de signer des ficher et de nous assurer qu'une transmission de donnée s'effectue sans faille ni compromission.
Nous travaillerons sous linux, en utilisant principalement la console.

Introduction : problème des clés symétriques.

Avant 1976, les cryptosystèmes reposaient tous sur des chiffrement à clé symétrique. Le principe en est très simple et intuitif : deux personnes qui veulent s'envoyer des messages codés utilisent une même méthode pour chiffrer et déchiffrer les messages. Par exemple une méthode peut être : *ramplecar tous las "e" per das "a"*. Les deux utilisateur qui connaissent cette règle peuvent alors s'échanger des messages cryptés. Conséquence de l'utilisation d'une clé symétrique : un utilisateur tiers, dès lors qu'il connaît la clé peut chiffrer et déchiffrer tout message.
Deux problèmes se posent :

- Avant d'échanger des messages cryptés, les deux utilisateurs doivent se transmettre la clé.
- Il est nécessaire d'établir un canal de transmission "sécurisé" pour l'échange de la clé (chuchotement à l'oreille, lettre cachetée ...)

Ces deux préalables sont particulièrement difficiles à établir. En particulier les transmissions sur les réseaux informatique sont connus pour leurs failles lors de transmission de données.
La solution proposée par les algorithmes à clé asymétrique est un système à deux clés ou seule la clé permettant de chiffrer le message (clé publique) est envoyée à travers le réseau. Il est possible à tous de chiffrer un message mais seul le possesseur de la clé privée va pouvoir déchiffrer le message.

1) créer des clés publiques et privées avec GnuPG

GnuPG est un système de cryptographie à clé asymétrique, c'est a dire qu'il fonctionne avec deux clés. La clé publique sera diffusée à toute personne désirant de vous envoyer un message crypté. La clé privée sera connue uniquement par la personne qui veut déchiffrer le message (vous en l'occurrence).
La première étape consiste à créer chacune de ces deux clés.

Opération 1.1 :

- Lancez la création d'un nouveau jeu de clé avec `gpg --gen-key`.
- Choisissez comme algorithme "DSA and Elgamal"
- Longueur de clé ELG-E : 2048
- donnez une expiration à votre clé (2-3 jours, pas plus long)
- complétez toutes les informations personnelles demandées, donnez des informations exactes et évitez les caractères accentués.

- Le "pathphrase" correspond au mot de passe qu'il vous sera demandé pour déchiffrer le message. La case est sensible, il est conseillé de mêler chiffres et lettres. **! Retenez ce mot de passe !**

GnuPG passe dans sa phase de création de clé qui peut durer un certain temps. Il se peut aussi que vous ayez à générer des évènements aléatoires pendant cette procédure (frappe au clavier, mouvement de souris).
Pendant ce temps, recherchez sur Internet des informations concernant l'algorithme de GnuPG

Question 1.1 :

- Que signifie l'acronyme DSA ?
- Quelles sont les trois étapes de l'algorithme DSA ?

Les systèmes de cryptographie asymétriques sont considérés comme les plus sûrs actuellement. Toutefois, on pourrait critiquer au moins un point

Question 5.2 :

- Dans les systèmes asymétriques, quelle est la partie qui peut intéresser un pirate ? Quel est donc le point faible de la cryptographie asymétrique ?
- Comment GnuPG apporte une (petite) sécurité supplémentaire pour adresser ce problème ?

Les clés doivent être générées ... nous vérifions rapidement leur présence ainsi que le contenu de la clé publique

Opération 1.2 :

- Vérifiez la présence de la clé secrète avec `gpg --list-keys`.
- Vérifiez la présence de la clé privée avec `gpg -list-secret-keys`.
- La sortie obtenue doit être de la forme :

```
sec  1024D/8AEEFA6F 2005-12-08 [expires: 2005-12-11]
uid  Mathieu Petit (Cle de Mathieu) <petit@ecole-navale.fr>
```

  Les chiffres soulignés sont l'identifiant de la paire de clé. Notez l'identifiant correspondant à vos clés.
- Affichez la clé publique avec `gpg --armor --export [identifiant]`.

2) Chiffrer et déchiffrer en local.

Nous avons tout les outils en main pour pouvoir chiffrer et déchiffrer des messages. Si votre ordinateur contient des données sensibles (fichier de mots de passes par exemple), il peut être utile de les crypter, même s'il n'est pas question de transmission réseau içi.

Opération 2.1 :

- Créer et enregistrez un fichier contenant un message (nommez le "message.txt").
- Encryptez le fichier avec

```
gpg --encrypt --armor -r [identifiant] message.txt
```

- Le fichier message.txt.asc est crée. Vérifiez que son contenu est bien encrypté.
- Supprimez l'original : `rm message.txt`

Passons à l'étape de décryptage du message.

Opération 2.2 :

- décrypter le message avec `gpg -d message.txt.asc`. GnuPG vérifie l'utilisation de la clé privée en vous demandant votre pathphrase puis le contenu du fichier s'affiche en clair
- sauver le résultat dans message.txt : `gpg -d message.txt.asc > message.txt`

3) Mettre sa clé publique à disposition

Le moyen le plus simple de permettre à une autre personne de vous envoyer des messages chiffrés est de rendre votre clé publique disponible sur Internet. Il existe à cet effet des serveurs de clé qui répertorient les clés publiques de millions d'utilisateurs. Une autre solution consiste à placer votre clé publique sur votre site internet ...

Opération 3.1 :

- Sauvegardez votre clé publique dans un fichier sur disque dur :
  `gpg --armor --export [identifiant] > public.key`
- Rendez vous sur http://keyserver.veridis.com:11371/ et choisissez "submit your key". Selectionner le fichier public.key et uploadez le sur le serveur de clé.
- Depuis la page principale du serveur, effectuez une recherche sur le critère "ecole-navale". Toutes les clés publiques dont la description contient un mail de l'école doivent ressortir. Vous devriez y trouver la votre, mais aussi celles de vos camarades.

maintenant que plusieurs clés sont disponibles, nous pouvons encrypter des messages à destination d'autres utilisateurs. il faut pour cela installer leur clé publique dans GnuPG.

Opération 3.2 :

- De façon informelle, constituez des groupes de deux personnes. L'une aura le rôle A et l'autre le rôle B.
- Récupérez via le serveur la clé publique de votre interlocuteur par le lien dans la colonne KeyID. Sauvez le fichier pubkey.asc sur votre disque dur.
- Importez cette clé dans GnuPG : gpg --import pubkey.asc
- Listez les clés publiques. Vous devez avoir dans votre trousseaux deux clés (la votre et celle de votre interlocuteur).

4) Conversation secrète

On imagine que vous voulez établir une conversation privée avec votre interlocuteur. Nous allons jouer un dialogue entre A et B (enfin juste une réplique)

Si vous etes la personne de rôle A, effectuez "Opération 4.1.A", si vous avez le rôle B, effectuez "Opération 4.1.B"

Opération 4.1.A :

- Préparez un fichier contenant une question (n'importe quoi, dans la mesure ou B sait y répondre)
- Encryptez ce fichier en utilisant cette fois ci l'identifiant de votre interlocuteur de rôle B.
- Envoyez lui le message par mail et attendez sa réponse (consultez votre boite mail régulièrement)
- Une fois le message réponse de B reçu, décryptez le.

Opération 4.1.B :

- Vérifiez régulièrement votre mail. Une fois que vous aurez reçu le message de A, décryptez le et préparez un fichier contenant votre votre réponse.
- Encryptez ce fichier en utilisant cette fois ci l'identifiant de votre interlocuteur de rôle A.
- Envoyez votre réponse encryptée par mail à A

Question 4.1 :

- L'échange s'est sans doute bien déroulé sans qu'aucune des données du message ne transite en clair. A priori, vous êtes sûrs d'avoir reçu un message chiffré de la part de l'interlocuteur que vous avez choisi. Trouvez vous des éléments qui permettent de certifier l'authenticité de la provenance de vos messages ? Ces éléments ont-ils un sens en sécurité réseau ? Autrement dit, qu'est-ce qui permet à A d'être certain que c'est bien B qui envoie les messages et inversement, comment pour B être sûr de A?

5) S'assurer de la provenance d'un message

Signer un courrier papier (ou tout autre document) permet de laisser sur cette lettre une empreinte qui doit nous assurer de l'authenticité de l'origine de la lettre. Toutefois, ces signatures sont falsifiables et le contrôle de l'authenticité du courrier donne beaucoup de travail aux graphologues.

Lors de transmissions numérique, le principe reste valable : il faut signer son message avec une signature personnelle pour garantir son authenticité.

Une signature numérique permet de faire la correspondance entre un message et le rédacteur de ce message. Elle est formée en deux étapes :

1. Génération d'une emprunte à partir du message que l'on veut signer. Nous étudierons l'emprunte d'un message un peu plus loin ...
2. Encryptage de cette emprunte avec notre clé privée.

Le résultat de cette d'encryptage va constituer la signature du message. En général, on l'ajoute a la suite du message. La caractéristique importante de cette signature est qu'elle peut être décodée uniquement en utilisant notre clé publique. Tout utilisateur possédant notre clé publique va pouvoir en déchiffrant la signature s'assurer que la provenance du message est bien la bonne car nous sommes les seuls a pouvoir chiffrer une signature lisible en utilisant notre clé publique.

Opération 5.1 :

- Créer un message "signature.txt" contenant du texte.
- Signer ce message en utilisant votre clé privée :
  `gpg --local-user [identifiant] --clearsign signature.txt`
- afficher le contenu du message signé :
  `cat signature.txt.asc`
- Verifiez la provenance du message signé :
  `gpg --verify signature.txt.asc`

Question 5.1 :

- comment est structuré un message signé
- Le message est-il chiffré ? Quelles étapes supplémentaires doit on effectuer pour obtenir un message chiffré signé ?
- modifiez le message dans le fichier signature.txt.asc et controlez à nouveau la provenance. Que ce passe t'il ?

Signons et encryptons un message en même temps à destination de votre interlocuteur.

Opération 5.2 :

- Créer un nouveau message "courrierA.txt" (si vous etes A) ou "courrierB.txt" (si vous etes B) contenant du texte.
- Encryptez et signez ce message simultanément :
  `gpg --recipient [id. interlocuteur] --armor --sign --encrypt courrier[A/B].txt`
- supprimez l'original "courrier*[A/B]*.txt", vérifiez que le fichier "courrier*[A/B]*.txt.asc"est bien crypté
- envoyez le fichier "courrier*[A/B]*.txt.asc" à votre interlocuteur.
- Après réception du message de la part de votre interlocuteur, décryptez le.

Question 5.2 :

- Lors de la dernière opération, comment GnuPG vous à t'il garanti la provenance du message ?

6) S'assurer de l'intégrité du message

Entre l'émission et la réception, il y a un bon moment de transport. Le message sera peut être altéré, intentionnellement ou non, des paquets peuvent se perdre, se dégrader ... Les algorithmes de hachages permettent de réduire un fichier à une combinaison binaire courte appelé somme de contrôle ("checksum")

Opération 6.1 :

- Créez un nouveau message nommez le "checksum.txt", encryptez et signez le
- Utilisez le programme md5sum pour générer la somme de contrôle de message de ce message :
  `md5sum checksum.txt.asc > checksum.txt.md5`

- Ouvrez le fichier checksum.txt.md5, il contient la somme de contrôle associée au nom du fichier.

Le message doit être transmis accompagné de sa somme de contrôle (le fichier .md5). Charge au récepteur de verifier l'intégrité des paquets recus en contrôlant à nouveau le fichier

Opération 6.2 :

- Vérifiez l'intégrité du fichier "checksum.txt.asc" (on imagine que vous venez de le recevoir) avec la somme de contrôle associée "checksum.txt.md5" :

```
md5sum --check checksum.txt.md5
```

- Alterez les données de "checksum.txt.asc" (supprimez un caractère) et refaites le contrôle d'intégrité.

Question 6.1 :

- Que ce passe t'il lors de ce dernier contrôle de données ? Générez à nouveau la somme de contrôle de "checksum.txt.asc" dans le fichier "checksum.txt2.md5". Comparez les deux sommes de contrôle (dans les fichier .md5), sont elles identiques ?
- Pour un message transitant par le réseau, nous pouvons maintenant assurer son contenu, son origine et son intégrité. C'est presque parfait. Toutefois, la somme de contrôle, qui emprunte les mêmes réseaux, n'est elle pas un point faible ? Dans quelle mesure ?

Conclusion

La plupart des systèmes de cryptographie modernes, dont SSL pour les échanges sur Internet fonctionnent par clé asymétrique. Nous avons vu que ces algorithmes augmentent de façon significative la sécurité des échanges par rapport aux mécanismes de sécurité à clé symétrique. La seule méthode de craquages de ce type d'algorithme reste l'usage de la force brute de calcul des ordinateurs. Vous avez encodés des clés de 2048 bits. Pour un attaquant, cela veux dire qu'il existe 2^{2048} combinaisons de clé à tester avant d'être certain de craquer le message. Avec la puissances des ordinateurs, il faut régulièrement augmenter la longueur des clés pour être sûr qu'aucune machine ne puisse tester toutes les combinaisons en un temps raisonnable.

Les systèmes à clé asymétriques, utilisent des grands nombres premiers pour générer les clés. Toute la solidité de ces algorithmes provient du fait que l'on ne connaît pas d'algorithme pour obtenir les diviseurs premiers de ces grands nombres. Si une telle méthode existait, il serait possible de déduire la clé privée a partir de la clé publique.

En 20 ans de recherche mathématique, la division en facteurs premiers est devenue une sorte de Graal dont la quête pourrait bientôt aboutir avec la constructions d'ordinateurs utilisant les principes de la mécanique atomique. On sait depuis 1994 (Peter Shor) qu'un ordinateur quantique pourrait résoudre le problème de la décomposition en facteur premiers en un temps polynomial. L'arrivée de l'ordinateur quantique signifiera-t-il la fin de près de 30 ans de cryptographie ?