Partie 4 :
La protection des données

Formateur : Azer Zairi
(azer.zairi@gnet.tn)

Partie 4 – Sections et objectifs

4.1 Confidentialité

Identifier l'algorithme de chiffrement à utiliser en fonction des besoins

4.2 Dissimulation des données

Utiliser une technique pour masquer les données

4.3 Intégrité et authenticité

Expliquer le rôle de la cryptographie pour garantir l'intégrité et l'authenticité des données.

4.4 Utilisation des hachages

Utilisez des outils de hachage pour hacher un fichier texte et vérifier l'intégrité d'un autre fichier.

4.5 Cryptographie à clé publique

Utiliser une signature numérique

4.1 Confidentialité

Confidentialité

Confidentialité des données

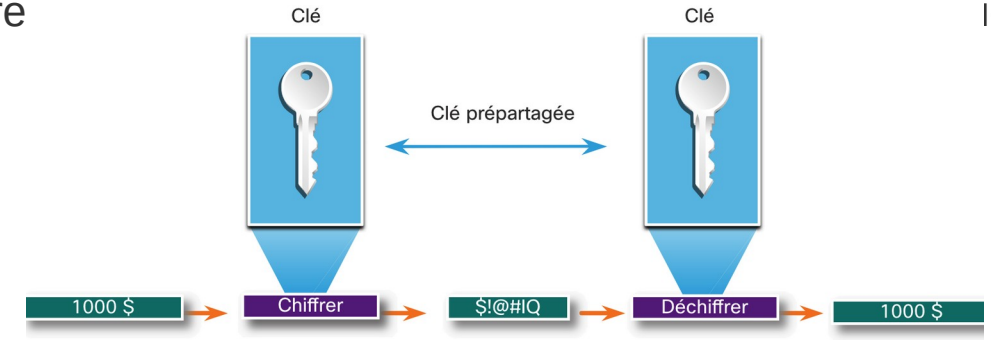
- **Deux méthodes** de **chiffrement** permettent de **garantir** la **confidentialité** des **données :** symétrique et asymétrique.
- **Chaque méthode** de chiffrement utilise un **algorithme** spécial, **appelé code**, pour chiffrer et déchiffrer les messages.
- Cet **algorithme** se compose d'une **série d'étapes bien définies** qui permettent de **chiffrer** et de **déchiffrer** des messages.
- Les **méthodes** permettant de créer du texte chiffré sont les suivantes :
  - **Transposition** : Au lieu de remplacer chaque lettre, cette méthode consiste à réorganiser les lettres du texte clair selon une clé de transposition
  - **Substitution** : Chaque lettres ou groupe de lettre du texte clair est remplacé par une autre lettre ou groupe de lettres selon certaines règles prédéfinies.
    - Ces règles sont généralement déterminées par une clé de substitution.
  - **Masque jetable** : technique de chiffrement qui implique l'utilisation d'une clé secrète générée aléatoirement pour masquer le texte clair, de sorte que le texte chiffré soit difficile à décrypter sans cette clé.

Confidentialité

Chiffrement symétrique

- Les algorithmes de chiffrement symétrique utilisent la **même clé prépartagée** pour **chiffrer** et **déchiffrer** les données.
- La clé prépartagée (ou « clé secrète ») est connue par l'expéditeur et le récepteur en amont de toute communication chiffrée.
- Lorsque vous utilisez un algorithme de chiffrement symétrique, et comme avec les autres types de chiffrement, **plus la clé est longue, plus le temps nécessaire pour la découvrir est important**.
- La plupart des clés de chiffrement sont **entre 112 et 256 bits**, mais pour s'assurer que le chiffrement est sûr, une **longueur** de clé **minimale de 128 bits** devrait être utilisée.

Remarque : Les **algorithmes** de **cryptage symétrique** sont couramment **utilisés** avec le **trafic VPN**, car ils utilisent **moins** de **ressources CPU** que les algorithmes de cryptage asymétrique, permettant le cryptage et le déchiffre ion d'un VPN.

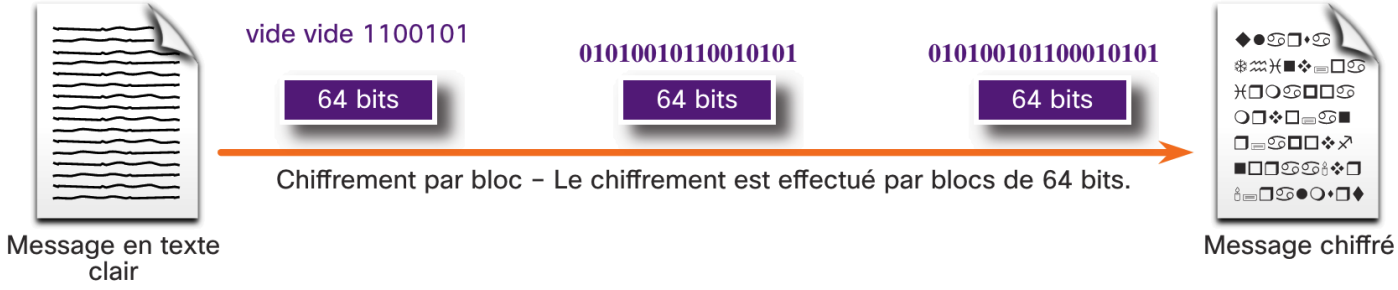


Confidentialité

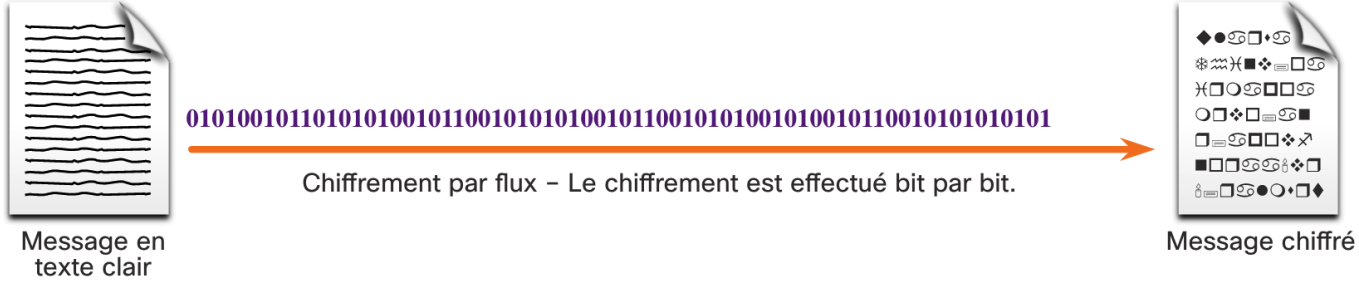
Chiffrement symétrique (suite)

Les algorithmes de chiffrement symétrique sont souvent classés dans l'une des catégories de chiffrement par bloc ou le chiffrement par flux.

- **Le chiffrement par bloc** transforme un bloc de texte en clair d'une longueur fixe en bloc de texte crypté de 64 ou 128 bits.



- **Le chiffrement de flux** est généralement plus rapide que le chiffrement par bloc, car les données sont chiffrées en continu.



Confidentialité

Chiffrement symétrique (suite)

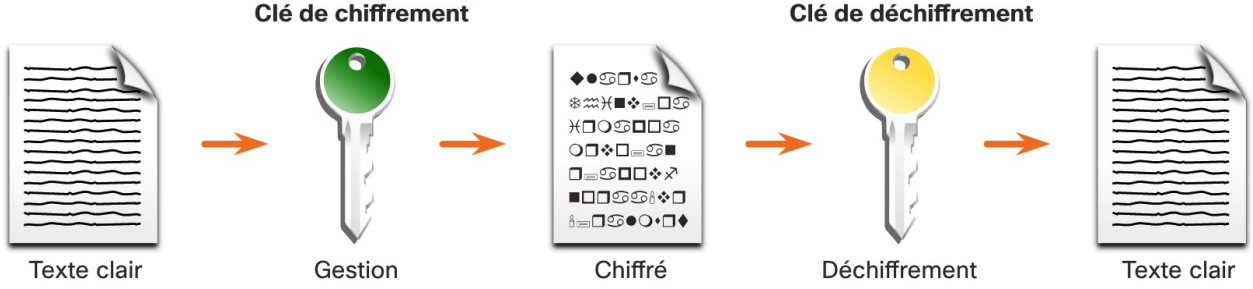
Des algorithmes de chiffrement symétrique bien connus sont décrits dans le tableau.

Algorithmes de chiffrement symétriques	Description
Algorithme DES (Data Encryption Standard)	Il s'agit d'un algorithme de chiffrement symétrique hérité. Il utilise une courte longueur de clé qui le rend non sécurisé pour la plupart des utilisations actuelles.
3DES (Triple DES)	Ceci est le remplacement de DES et répète le processus d'algorithme DES trois fois. Elle devrait être évitée si possible, car elle devrait être retirée en 2023. Si elle est mise en œuvre, utiliser des clés à durée de vie très courte.
AES	Il s'agit d'un algorithme de chiffrement symétrique populaire et recommandé. Il propose des combinaisons de clés 128, 192 ou 256 bits pour chiffrer des blocs de données de 128, 192 ou 256 bits.
Algorithmes SEAL (Software-Optimized Encryption Algorithm)	Il s'agit d'un algorithme de chiffrement symétrique rapide qui constitue une alternative à DES et 3AES. Il s'agit d'un chiffrement de flux qui utilise une clé de cryptage de 160 bits et qui a un impact moindre sur l'unité centrale par rapport à d'autres algorithmes basés sur des logiciels.
Algorithmes de la série Rivest Ciphers(RC)	Bien que plusieurs variantes aient été développées, celle de l'algorithme RC4 était la plus répandue. RC4 est un algorithme de chiffrement de flux utilisé pour sécuriser le trafic web avec les protocoles SSL et TLS. Il a été constaté qu'il avait plusieurs vulnérabilités qui l'ont rendu non sécurisé. RC4 ne doit pas être utilisé.

Confidentialité

Chiffrement asymétrique

- Les algorithmes **asymétriques** (ou «algorithmes à **clé publique**») sont conçus de sorte que la **clé** utilisée pour le **chiffrement diffère** de la **clé** utilisée pour le **chiffrement**.
- La clé de déchiffrement ne peut être déduite de la clé de chiffrement dans un délai raisonnable, et inversement.



- Les algorithmes asymétriques utilisent une clé publique et une clé privée, et les **deux clés** sont **capables** du processus de **cryptage**, mais la **clé** appariée **complémentaire** est nécessaire pour le déchiffrement.
- Le **processus** est également **réversible** puisqu'une clé privée doit être utilisée pour déchiffrer les données chiffrées avec la clé publique.
- Les **algorithmes asymétriques** assurent la **confidentialité** et **l'authentification** en utilisant ce processus.

Confidentialité

Chiffrement asymétrique (suite)

- Le **chiffrement asymétrique** peut utiliser des **longueurs** de **clé entre 512** à **4096** bits.
- Les longueurs de **clé supérieures ou égales à 2 048 bits** sont **fiables**, tandis que les longueurs de **clé de 1 024 ou moins** sont considérées comme **insuffisantes**.
- **Exemples** de protocoles qui utilisent des algorithmes à clé asymétrique :
  - Internet Key Exchange (IKE) **: il s'agit d'une composante fondamentale du VPN IPSec.**
  - **SSL (Secure Socket Layer) -** Ceci est maintenant implémenté en tant que **TLS** (Transport Layer Security) standard de l'IETF.
  - **SSH (Secure Shell) -** protocole qui assure une connexion à distance sécurisée aux appareils réseau.
  - **Pretty Good Privacy (PGP) :** il s'agit d'un programme informatique qui assure la confidentialité et l'authentification du chiffrement.
- Les **algorithmes asymétriques** sont **beaucoup plus lents** que les algorithmes symétriques, de sorte qu’ils sont généralement utilisés dans les **mécanismes cryptographiques** à **faible volume**, comme les **signatures numériques** et **l’échange** de **clés**.
- Toutefois, la gestion des clés des algorithmes asymétriques a tendance à être plus simple que celle des algorithmes symétriques, car l'une des deux clés de chiffrement ou de déchiffrement peut généralement être rendue publique.

Confidentialité

Chiffrement asymétrique (suite)

Algorithmes de chiffrement asymétrique	Longueur de clé (Key Length)	Description
Diffie-Hellman (DH)	512, 1 024, 2 048, 3 072, 4 096	Cela permet à deux parties de s'entendre sur une clé qu'elles peuvent utiliser pour chiffrer les messages qu'elles s'échangent. La sécurité de cet algorithme repose sur l'hypothèse qu'il est facile d'élever un nombre à une certaine puissance, mais qu'il est difficile de calculer quelle puissance a été utilisée étant donné le nombre et les résultats.
DSS (Digital Signature Standard (DSS) et DSA (Digital Signature Algorithm)	De 512 à 1 024	DSS spécifie DSA comme algorithme pour les signatures numériques. DSA est un algorithme à clé publique basé sur le schéma de signature ElGamal. La vitesse de création de signature est similaire à RSA mais est 10 à 40 fois plus lente pour la vérification.
Algorithmes de chiffrement Rivest, Shamir et Adleman (RSA)	De 512 à 2048	Il est destiné à la cryptographie à clé publique basée sur la difficulté actuelle de factoriser de très grands nombres. Il est considéré comme sûr étant donné les clés suffisamment longues et l'utilisation de mises en œuvre à jour.
ElGamal	De 512 à 1 024	Utilisé pour la cryptographie à clé publique basée sur l'accord de clé DH. Toutefois, l'inconvénient du système ElGamal est qu'il rend le message chiffré très volumineux, environ deux fois la taille du message d'origine. C'est pourquoi il n'est utilisé que pour les petits messages tels que les clés secrètes.
Techniques de courbe elliptique	224 ou plus	Elle permet d'adapter de nombreux algorithmes de chiffrement, tels que Diffie-Hellman ou ElGamal.

Confidentialité

Chiffrement asymétrique – Confidentialité

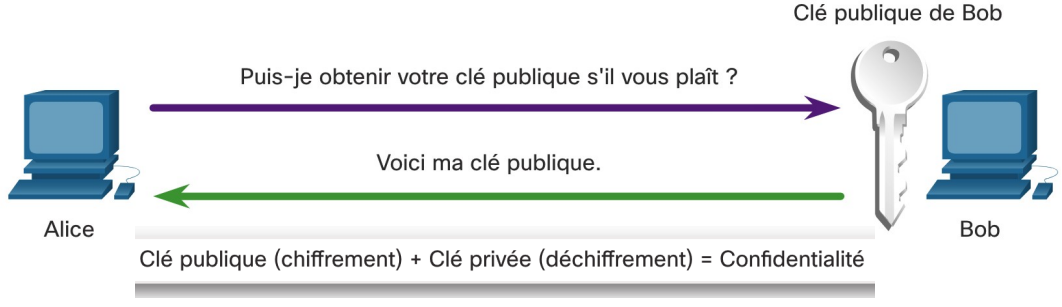
- Les algorithmes **asymétriques** assurent la **confidentialité sans partager** de **mot de passe** au préalable.
- La confidentialité des algorithmes asymétriques est garantie quand vous lancez le processus de chiffrement avec la clé publique.
- Le processus peut être résumé à l'aide de la formule :

Clé publique (chiffrement) + Clé privée (déchiffrement) = Confidentialité

- La clé publique sert à chiffrer les données, la clé privée doit être utilisée pour les déchiffrer.
- Un seul hôte possède la clé privée ; par conséquent, la confidentialité est assurée.
- Si la clé privée est compromise, une autre paire de clés doit être générée pour remplacer la clé compromise.

Exemple : Échange de données de Bob et Alice – Confidentialité

- Alice demande et obtient la clé publique de Bob.

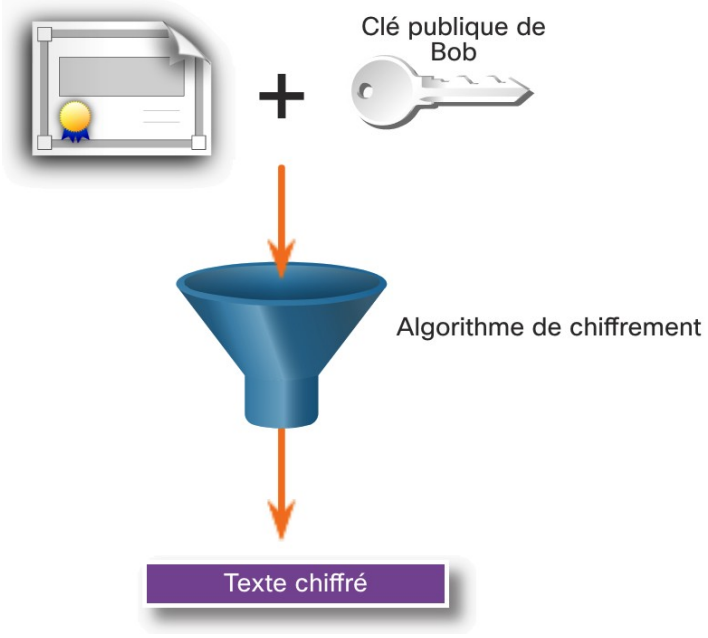


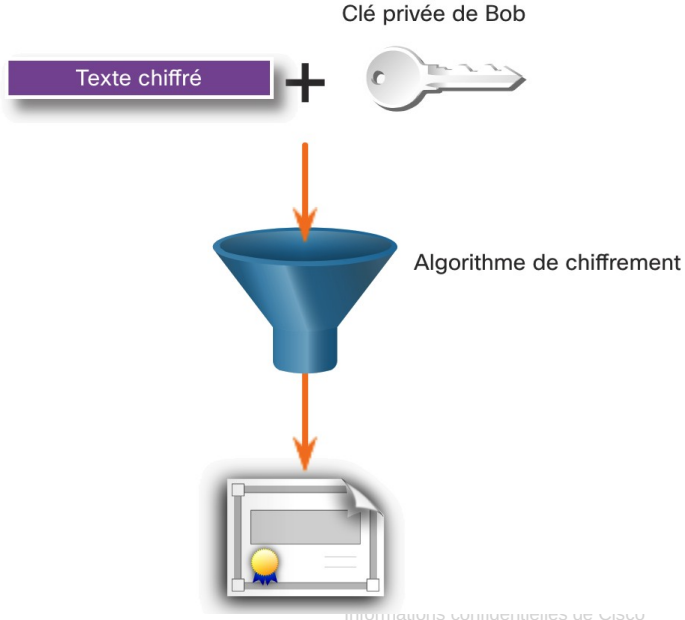
Confidentialité

Chiffrement asymétrique – Confidentialité (suite)

Exemple : Échange de données de Bob et Alice – Confidentialité

- Alice utilise la clé publique de Bob pour chiffrer un message à l'aide d'un algorithme.
- Alice envoie le message chiffré à Bob.
- Puis, Bob utilise sa clé privée pour déchiffrer le message.
- Puisque Bob est le seul avec la clé privée, le message d'Alice ne peut être déchiffré que par Bob et donc la confidentialité est atteinte.





Confidentialité

Chiffrement asymétrique – Authentification

- **L'authentification** des algorithmes asymétriques est réalisée quand vous lancez le processus de **chiffrement** avec la **clé privée**.
- Le processus peut être résumé à l'aide de la formule :

Clé privée (chiffrer) + clé publique (déchiffrer) = Authentification

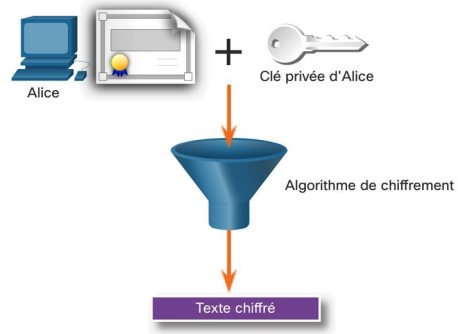
- La clé privée sert à chiffrer les données, la clé publique correspondante doit être utilisée pour les déchiffrer.
- Un seul hôte possède la clé privée ; par conséquent, seul cet hôte peut avoir chiffré le message, authentifiant ainsi l'expéditeur.
- Si un hôte déchiffre avec succès un message à l'aide d'une clé publique, c'est que celui-ci a bien été chiffré avec la clé privée ; l'identité de l'expéditeur est vérifiée.

Confidentialité

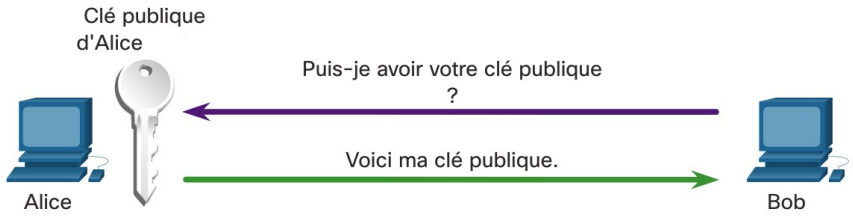
Chiffrement asymétrique – Authentification (suite)

Exemple : Échange de données de Bob et Alice – Authentification

- Alice chiffre le message à l'aide de sa clé privée
- Alice envoie le message chiffré à Bob.
- Bob doit authentifier que le message vient véritablement d'Alice.

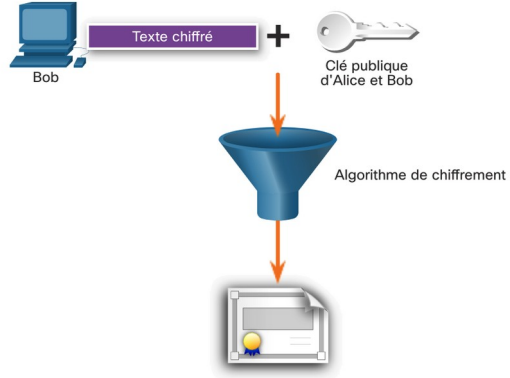


Clé privée (chiffrer) + clé publique (déchiffrer) = Authentification



- Afin d'authentifier le message, Bob demande la clé publique d'Alice.

Bob doit vérifier que le message provient effectivement d'Alice. Il demande et obtient la clé publique d'Alice.



- Bob utilise la clé publique d'Alice pour déchiffrer le message.

Bob utilise la clé publique pour déchiffrer le message et vérifier qu'il provient effectivement d'Alice.

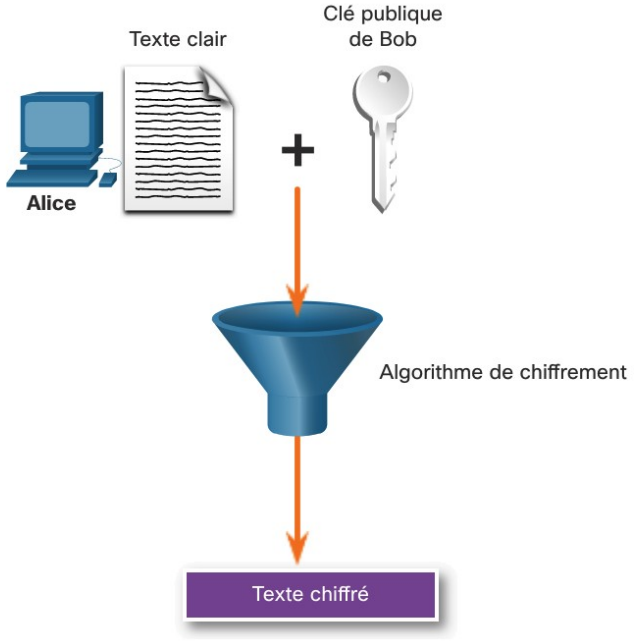
Confidentialité

Chiffrement asymétrique – Intégrité

- En **combinant** les **deux processus** de **chiffrement asymétrique**, vous assurez **l'intégrité**, **l'authentification** et la **confidentialité** de vos messages.
- ***Dans cet exemple***, un message est chiffré avec la clé publique de Bob et un hash chiffré est réalisé avec la clé privée d'Alice afin d'en garantir la confidentialité, l'authenticité et l'intégrité.

Exemple : Bob et Alice échangent des données – Intégrité

- Alice veut envoyer un message à Bob en s'assurant que seul celui-ci peut lire le document.
- Alice veut garantir la confidentialité de son message.
- Alice utilise la clé publique de Bob pour chiffrer le message.
- Seul Bob peut le déchiffrer à l'aide de sa clé privée.

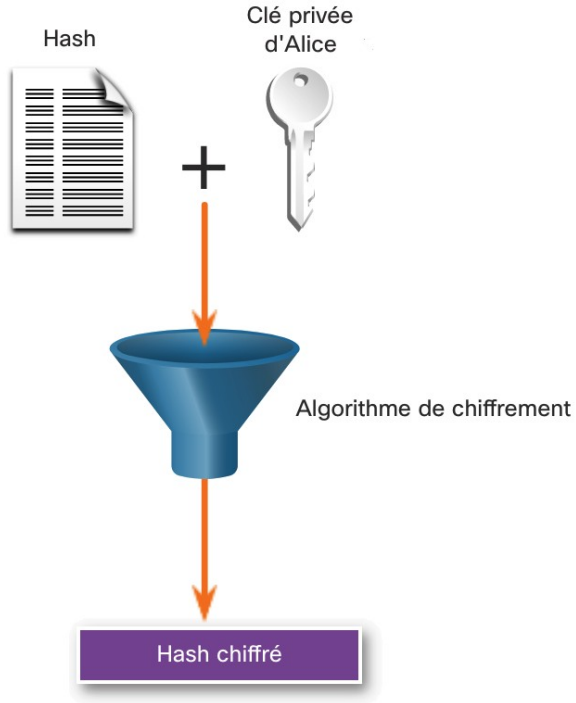


Confidentialité

Chiffrement asymétrique – Intégrité (suite)

Exemple : Bob et Alice échangent des données – Intégrité

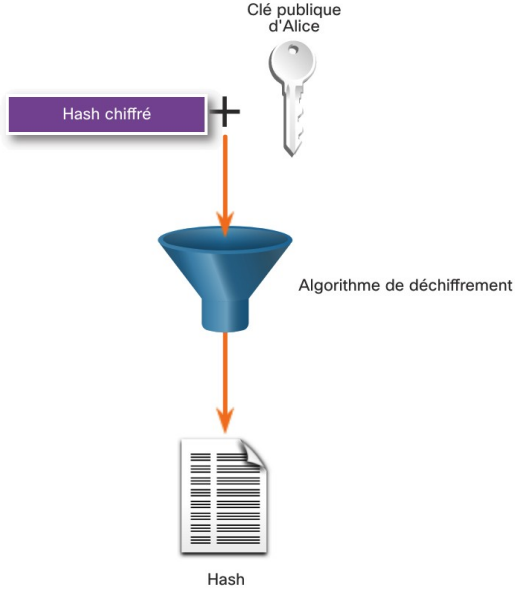
- Alice veut également garantir l'intégrité et l'authentification des messages.
- L'authentification garantit à Bob que le document a bien été envoyé par Alice et intégrité lui confirme que celui-ci n'a pas été modifié.
- Alice utilise sa clé privée pour chiffrer un hachage du message.
- Alice envoie le message chiffré avec son hash chiffré à Bob.



Confidentialité

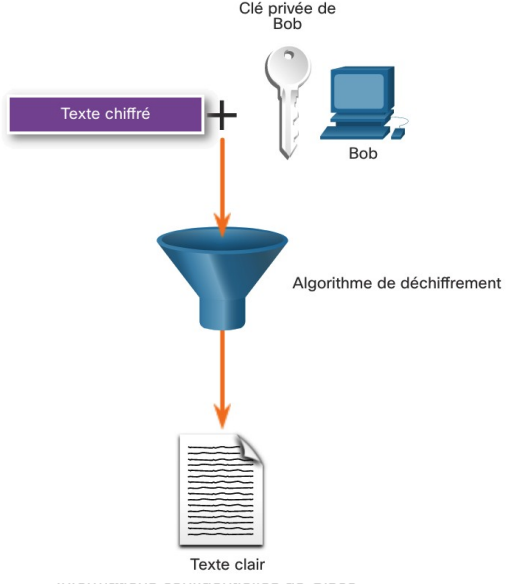
Chiffrement asymétrique – Intégrité (suite)

Exemple : Bob et Alice échangent des données – Intégrité



- Bob utilise la clé publique d'Alice pour vérifier que le message n'a pas été modifié.
- Le hash reçu correspond au hash déterminé localement à partir de la clé publique d'Alice.
- En outre, il confirme qu'Alice est véritablement l'expéditrice du message étant donné que personne d'autre ne possède la clé privée d'Alice.

- Bob utilise sa clé privée pour déchiffrer le message.



Confidentialité

L'échange de clés Diffie-Hellman

- L'échange de clé **DH** est un **algorithme mathématique asymétrique** qui permet à deux ordinateurs de **générer** un **secret partagé identique**, et cela **sans avoir communiqué auparavant**.

- En réalité, la **nouvelle clé** partagée **n'est pas véritablement échangée** entre l'émetteur et le récepteur.

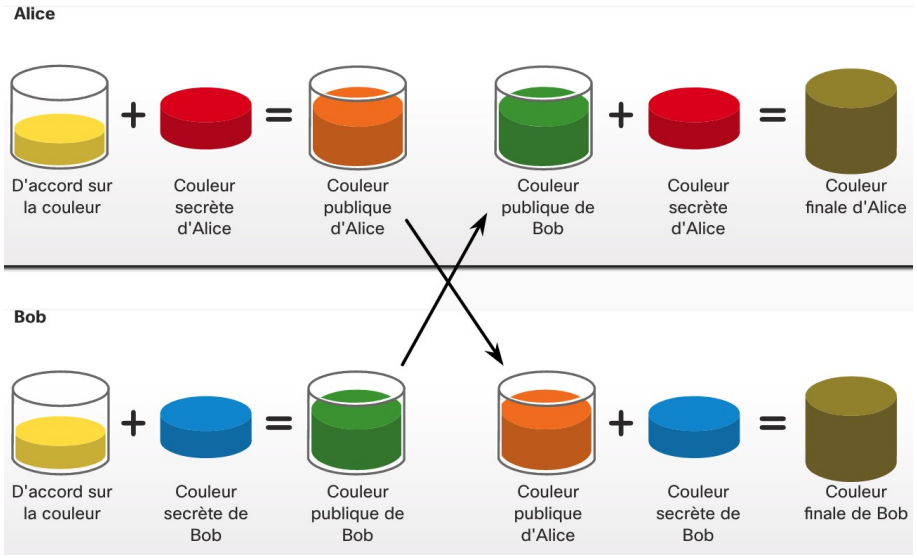
- Comme les deux parties connaissent la clé, elle peut cependant être utilisée par un algorithme de chiffrement pour chiffrer le trafic entre les deux systèmes.

- Voici deux exemples d'instances lorsque l'algorithme DH est couramment utilisé:
  - Les données sont échangées à l'aide d'un VPN IPSec ;
  - Les données SSH sont échangées.

Confidentialité

Diffie-Hellman (Cont.)

L'utilisation de couleurs sur la figure est préférée à celle de longs numéros complexes et permet de simplifier le processus d'accord par clé DH.



- L'échange de clés DH commence par la définition, par Alice et Bob, d'une **couleur commune** arbitraire dont la confidentialité n'est pas requise.
- Les **couleurs secrètes** choisies pour **ne jamais être partagées** (Alice-rouge, Bob-bleu) représentent la clé privée de chaque partie.
- Par conséquent, **Alice mélange** le **jaune** au **rouge** et obtient ainsi une **couleur publique** orange. **Bob mélange** le **jaune** au **bleu** et obtient ainsi une couleur **publique** verte.
- Alice **envoie sa couleur publique** (l'orange) à Bob et Bob envoie sa couleur publique (le vert) à Alice.
- **Alice** et **Bob mélangent chacun** la **couleur** qu'ils ont **reçue avec** leur **couleur d'origine secrète** (rouge pour Alice et bleu pour Bob). Le **résultat final** est un **mélange** brun **identique** à celui du partenaire.
- La couleur brune représente la clé secrète partagée obtenue par Bob et Alice.

Confidentialité

Diffie-Hellman (Cont.)

- La sécurité de **l'algorithme DH** repose sur l'utilisation de **nombres extrêmement grands** dans ses calculs.

- **L'algorithme** Diffie-Hellman utilise **différents groupes** DH pour déterminer la **puissance** de la **clé** utilisée dans le processus d'accord par clé.

- **Plus** les **nombres** du **groupe** sont **grands**, **plus** celui-ci est **sûr**, mais plus le **délai** de **calcul** de la clé est **important**.

- Vous trouverez ci-dessous les groupes DH pris en charge par le logiciel Cisco IOS et leur valeur en nombre premier :
    - Groupe Diffie-Hellman 1 : 768 bits
    - Groupe Diffie-Hellman 2 : 1024 bits
    - Groupe Diffie-Hellman 5 : 1536 bits
    - Groupe Diffie-Hellman 14 : 2048 bits
    - Groupe Diffie-Hellman 15 : 3072 bits
    - Groupe Diffie-Hellman 16 : 4096 bits

4.2 La dissimulation des données

Dissimulation des données

Techniques de masquage de données

- La **technologie** de **masquage** de **données sécurise** les **données** en **remplaçant** les **informations sensibles** par une **version non sensible**.

- **La version non sensible ressemble** et **agit comme la version d'origine**, de sorte qu'un **processus organisationnel peut utiliser des données non sensibles sans** qu'il soit nécessaire de **modifier** les **applications** ou les installations de stockage de données.

- Dans la plupart des cas, **la dissimulation limite la propagation des données sensibles** dans les systèmes IT en utilisant des données de substitution à **des fins de test et d'analyse**.

- Le masquage des données peut s'opérer de manière dynamique si le système ou l'application détermine qu'une demande d'informations sensibles introduite par l'utilisateur est risquée.

- Le masquage de données peut remplacer des données sensibles dans les environnements hors production afin de protéger les informations sous-jacentes.
- Plusieurs techniques de masquage des données peuvent être utilisées.

Dissimulation des données

Techniques de masquage de données (suite)

Toutes les méthodes ci-dessous garantissent que les données restent pertinentes, mais suffisamment modifiées pour les protéger.

- **La substitution** remplace les données par des valeurs authentiques en apparence afin de rendre anonymes les enregistrements de données.

- **Le brassage** déduit un ensemble de remplacement de la même colonne de données que celle qu'un utilisateur souhaite masquer.
  - Cette technique convient parfaitement aux données financières dans une base de données test, par exemple.

- **La technique d'annulation** applique une valeur nulle à un champ donné, ce qui empêche toute visibilité des données.

Dissimulation des données

Stéganographie

- La **stéganographie dissimule** les **données** (le message) **dans** un **autre fichier**, comme un graphique, un fichier audio ou un autre fichier texte.

- La stéganographie présente un **avantage** par rapport à la cryptographie : le **message secret n'attire** pas spécialement **l'attention**.

- Personne ne saurait jamais qu'une image contient un message secret en regardant le fichier, que ce soit sous forme électronique ou sur papier.

- **Plusieurs éléments** interviennent dans la **dissimulation** des **données**:
  - Il y a tout d'abord les **données intégrées**, c'est-à-dire le message secret.
  - Le texte, l'image ou le son de **couverture dissimule les données intégrées** en générant le texte, l'image ou le son stéganographique.
  - C'est une **clé** de **stéganographie** qui **contrôle** le processus de **dissimulation**.

Dissimulation des données

Stéganographie (suite)

Techniques de stéganographie	La méthode du bit de poids faible (LSB) est utilisée pour intégrer les données dans une image de couverture. Cette méthode utilise des bits de chaque pixel de l'image.Le pixel est l'unité de base d'une couleur programmable dans une image informatique. La couleur exacte d'un pixel est une combinaison de trois couleurs : le rouge, le vert et le bleu (RVB).Trois octets de données définissent la couleur d'un pixel (un octet par couleur). Un système couleur 24 bits utilise les trois octets. La méthode LSB utilise un bit de chacune des composantes couleur rouge, verte et bleue. Chaque pixel peut stocker trois bits.
Stéganographie sociale	Le stéganographie sociale consiste à dissimuler des informations en créant un message lisible d'une certaine manière par une certaine audience.Le message n'est pas compréhensible par les autres personnes qui lisent les informations « normalement ». Les adolescents actifs sur les réseaux sociaux utilisent cette technique pour circonscrire certaines informations à un cercle social spécifique (leurs plus proches amis, par exemple), en s'assurant qu'elles ne sortent pas du contexte de ces relations. La stéganographie sociale est également utilisée dans les pays qui censurent les médias. Pour faire passer un message, les utilisateurs peuvent faire volontairement des fautes d'orthographe ou utiliser des références obscures. En effet, ils communiquent avec différents publics simultanément, envoyant deux messages différents : le message apparent et le message secret.
Détection	La stéganalyse a pour vocation de détecter si un élément est susceptible de contenir des informations cachées Le but de la stéganalyse est de découvrir cette information cachée. Les motifs de l'image stéganographique paraissent suspects. Les utilitaires d'analyse de disque peuvent signaler les informations cachées dans les clusters inutilisés des supports de stockage. Les filtres peuvent capturer des paquets de données dont les en-têtes contiennent des informations cachées.Ces deux méthodes utilisent des signatures de stéganographie.En comparant l'image d'origine à l'image stéganographique, un analyste peut relever visuellement des schémas répétitifs.

4.3 Intégrité et authenticité

Intégrité et authenticité

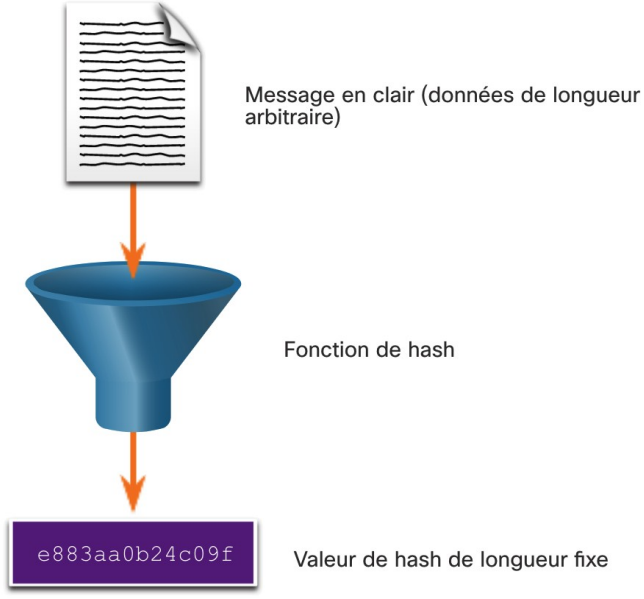
Sécuriser vos communications

- Les entreprises doivent fournir une assistance pour sécuriser le trafic interne et externe.
- Ce sont les **quatre éléments** des communications sécurisées:
  - **L'intégrité des données** : elle garantit que le message n'a pas été modifié et toute modification des données en transit sera détectée L'intégrité est assurée par la mise en œuvre de l'un ou l'autre des algorithmes de hachage sécurisé (SHA-2 ou SHA-3). L'algorithme de résumé des messages MD5 est toujours en cours d'utilisation, mais il doit être évité car il est non sécurisé et crée des vulnérabilités dans un réseau.
  - **Authentification d'origine** - Garantit que le message n'est pas une contrefaçon et provient réellement de qu'il déclare. De nombreux réseaux modernes assurent l'authentification au moyen d'algorithmes tels que le code d'authentification des messages basé sur le hachage (HMAC : Hash-based Message Authentication Code).
  - **Confidentialité des données** - Garantit que seuls les utilisateurs autorisés peuvent lire le message. Si le message est intercepté, il ne peut être déchiffré dans un délai raisonnable. La confidentialité des données est implémentée à l'aide d'algorithmes de chiffrement symétrique et asymétrique.
  - **La non-répudiation des données** - garantit que l'expéditeur ne peut pas répudier ni réfuter la validité d'un message envoyé. La non-répudiation repose sur le fait que seul l'expéditeur dispose des caractéristiques uniques ou de la signature relative au traitement du message.

Intégrité et authenticité

Fonctions de hash cryptographique

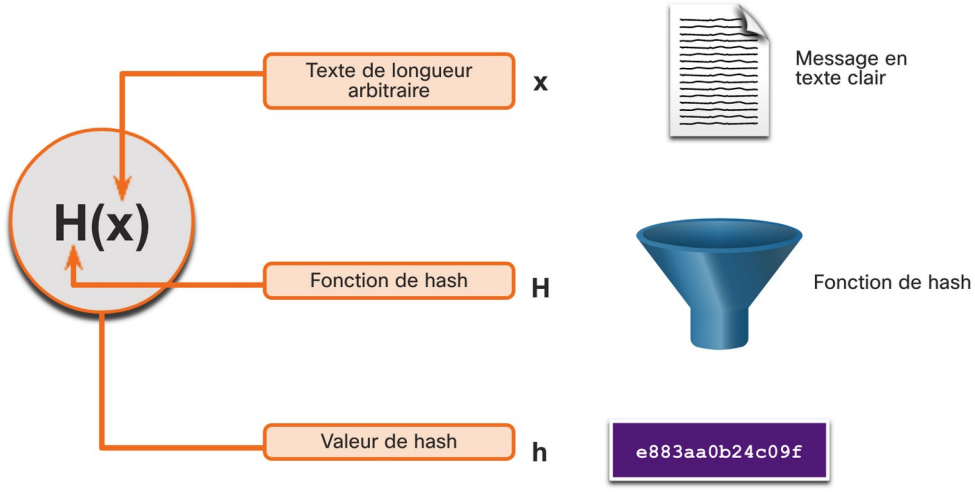
- Le **hash** est une **fonction** mathématique **unidirectionnelle** relativement simple à calculer, mais extrêmement difficile à inverser.
- La fonction de **hash** cryptographique **permet** également de **vérifier l'authentification**.
- Une fonction de **hash** prend un bloc variable de données binaires (le message) et **produit** une représentation **condensée** de **longueur fixe** (le hash).
- Avec les fonctions de hash, deux ensembles de données différents ne peuvent pas générer de hashs identiques sur le plan informatique.
- **Chaque fois** que les **données** sont **modifiées** ou **altérées**, la valeur de **hash change** également.
- La fonction de **hash** cryptographique s'applique à de nombreuses **situations**, à des fins **d'authentification** de l'entité, **d'intégrité** des données et **d'authenticité** des données.



Intégrité et authenticité

Opération de hash cryptographique

- Mathématiquement, l'équation **h= H(x)** sert à expliquer comment fonctionne un algorithme de hachage.
- Comme le représente la figure, une fonction de hash H prend une entrée x et renvoie une valeur de hash de chaîne de taille fixe h.
- Une fonction de **hash** cryptographique doit posséder les **propriétés** suivantes :
  - Il n'y a **pas de limite de longueur** pour le texte saisi.
  - La **longueur** du **résultat** est **fixe**.
  - H(x) est relativement facile à calculer pour toute valeur x.
  - H(x) est **unidirectionnel** et **irréversible**.
  - H(x) est **libre** de **toute collision** : deux valeurs d'entrée distinctes génèrent des valeurs de hash différentes.



- Difficile à inverser signifie que, compte tenu d'une valeur de hachage de h, il est impossible de trouver une entrée pour x telle que h=h (x).

Intégrité et authenticité

MD5 et SHA

- Il existe quatre fonctions de hachage bien connues.
  - **MD5 avec un condensé de 128 bits** : fonction unidirectionnelle qui génère un message haché de 128 bits. La fonction MD5 est considérée comme un algorithme obsolète qu'il convient d'éviter ; elle doit uniquement être utilisée en l'absence d'autres solutions. Il est recommandé d'utiliser SHA-2 ou SHA-3 à la place.

  - **SHA-1** : très similaire aux fonctions de hash MD5. Elle se décline en plusieurs versions. SHA-1 crée un message haché de 160 bits et est légèrement plus lent que MD5. Elle présente des défauts et apparaît comme un algorithme obsolète.

  - **SHA-2**- cela inclut SHA-224 (224 bits), SHA-256 (256 bits), SHA-384 (384 bits) et SHA-512 (512 bits). Si vous utilisez SHA-2, les algorithmes SHA-256, SHA-384 et SHA-512 doivent être utilisés chaque fois que possible.

  - **SHA-3** - SHA-3 est le plus récent algorithme de hachage et a été introduit par NIST comme une alternative pour la famille SHA-2 d'algorithmes de hachage. Il comprend SHA3-224 (224 bits), SHA3-256 (256 bits), SHA3-384 (384 bits) et SHA3-512 (512 bits). Les SHA-3 sont des algorithmes de nouvelle génération et qu'il convient d'utiliser dès que possible.

Intégrité et authenticité

Authentification de l'origine

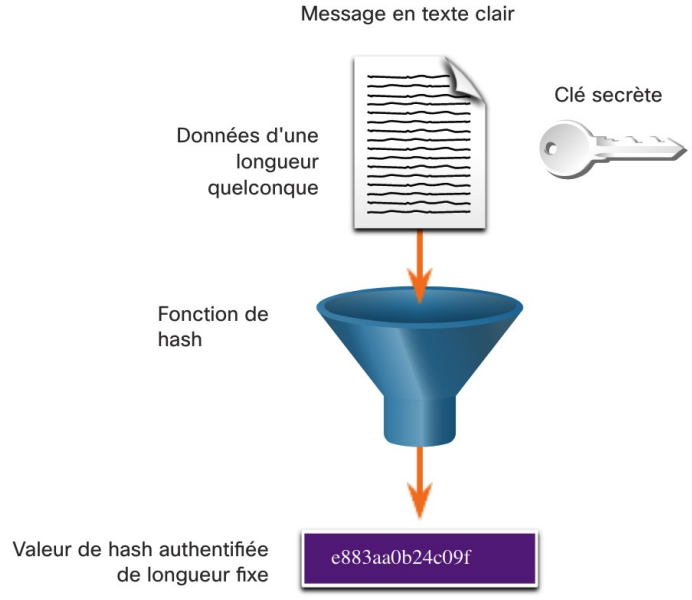
- Pour ajouter l'authentification de l'origine et l'assurance d'intégrité, utilisez un code d'authentification de message de hachage à clé (HMAC).

- HMAC utilise une clé secrète supplémentaire comme entrée pour la fonction de hachage.

- **Algorithme de hachage HMAC**
  - Un HMAC est calculé à l'aide de tout algorithme cryptographique qui combine une fonction de hachage cryptographique avec une clé secrète.

  - Seuls l'expéditeur et le récepteur connaissent la clé secrète, et le résultat de la fonction de hash dépend à présent des données d'entrée et de la clé secrète.

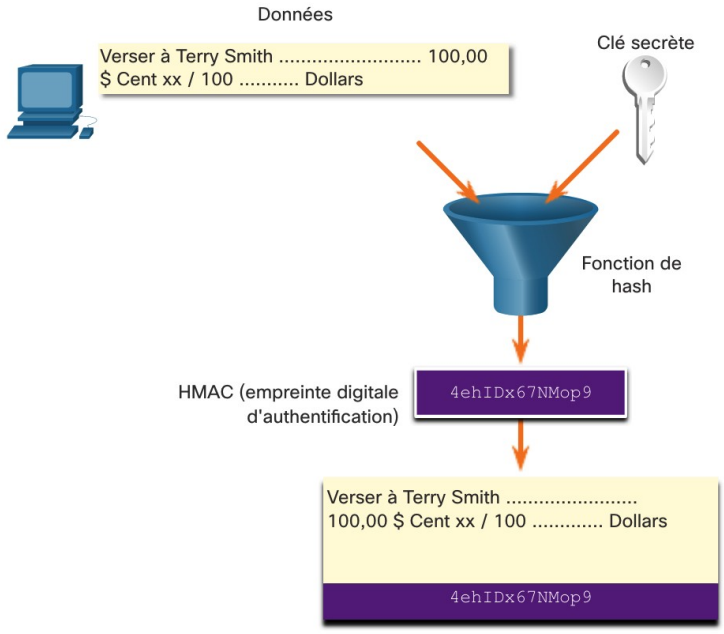
  - Si les deux parties partagent une clé secrète et utilisent les fonctions HMAC pour l'authentification, un condensé HMAC correctement constitué d'un message qu'une partie a reçu indique que l'autre partie est l'auteur du message.



Intégrité et authenticité

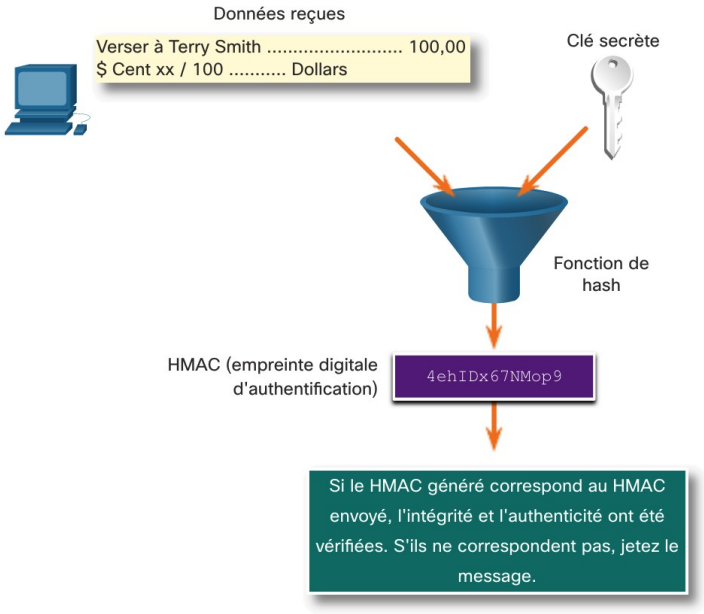
Authentification de l'origine (suite)

- **Création de la valeur HMAC**
  - Comme illustré à la Figure 2, l'appareil émetteur saisit des données (telles que la paie de 100 dollars de Terry Smith et la clé secrète) dans l'algorithme de hash et calcule le condensé HMAC de longueur fixe.
  - Ce condensé authentifié est ensuite joint au message et envoyé au récepteur.



Intégrité et authenticité

Authentification de l'origine (suite)



- **Vérification de la valeur HMAC**
  - Dans la figure, le périphérique de réception supprime le résumé du message et utilise le message en clair avec sa clé secrète comme entrée dans la même fonction de hachage.
  - Lorsque le condensé calculé par l'appareil récepteur équivaut au condensé envoyé, le message n'a pas été modifié.
  - En outre, l'origine du message est authentifiée, car seul l'expéditeur possède une copie de la clé secrète partagée.
  - La fonction HMAC garantit l'authenticité du message.

4.4 Utiliser des hashs

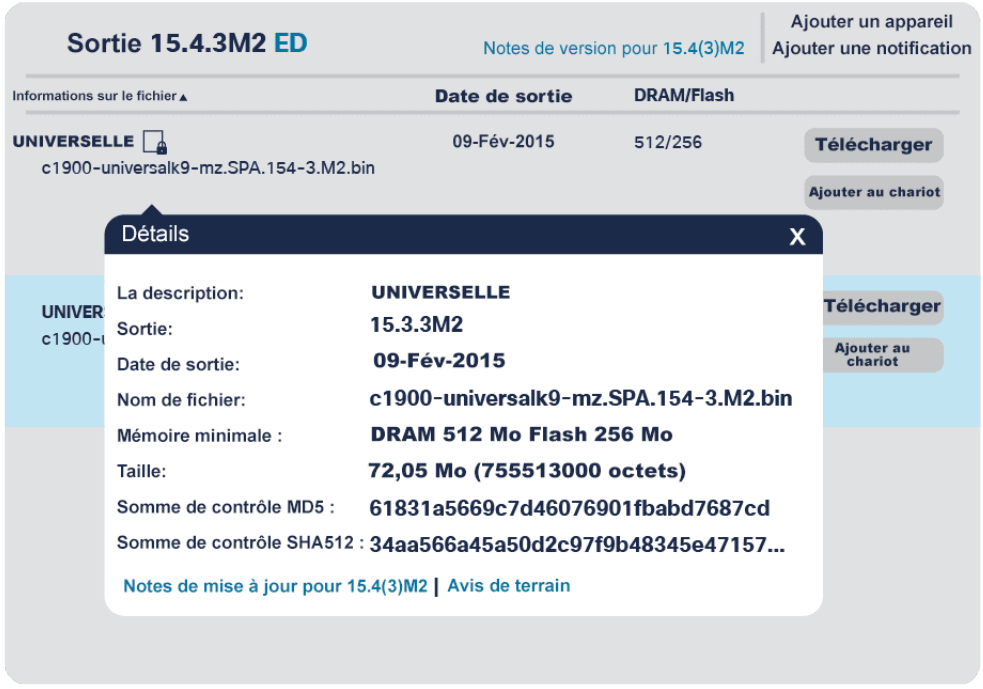
Utilisation des hachages

Hash de fichiers et de supports numériques

- **L'intégrité garantit** que les données et informations sont **complètes** et ne subissent **aucune altération** au moment de leur acquisition.

- Pour vérifier l'intégrité de toutes les images IOS, Cisco propose des sommes de contrôle MD5 et SHA sur son site web de téléchargement de logiciels.

- L'utilisateur peut comparer ce condensé MD5 avec le condensé MD5 d'une image Cisco IOS installée sur un appareil et peut être certain que personne n'a trafiqué ou modifié le fichier image Cisco IOS.



Utilisation des hachages

Hash de fichiers et de supports numériques (suite)

- Dans le **domaine** de **l'expertise judiciaire** en **informatique, le hash** est **utilisé** pour **vérifier tous les supports numériques** qui **contiennent des fichiers.**

- Par exemple, **l'enquêteur crée** un **hash** et une **copie bit à bit** du **support** contenant les fichiers pour produire un clone numérique.
- **L'examinateur compare** le **hash** du **support d'origine** avec la **copie** et, si les deux valeurs correspondent, les copies sont identiques.
- Le fait qu'un ensemble de bits soit identique à l'ensemble de bits initial établit la fixité.
- Cette fixité permet de répondre à plusieurs questions :
  - L'enquêteur dispose-t-il des fichiers auxquels il s'attendait ?
  - Les données ont-elles été altérées ou modifiées ?
  - L'enquêteur peut-il prouver que les fichiers n'ont pas été altérés ?
- L'expert scientifique peut, à présent, examiner la copie à la recherche de preuves numériques, en laissant intacte la version d'origine.

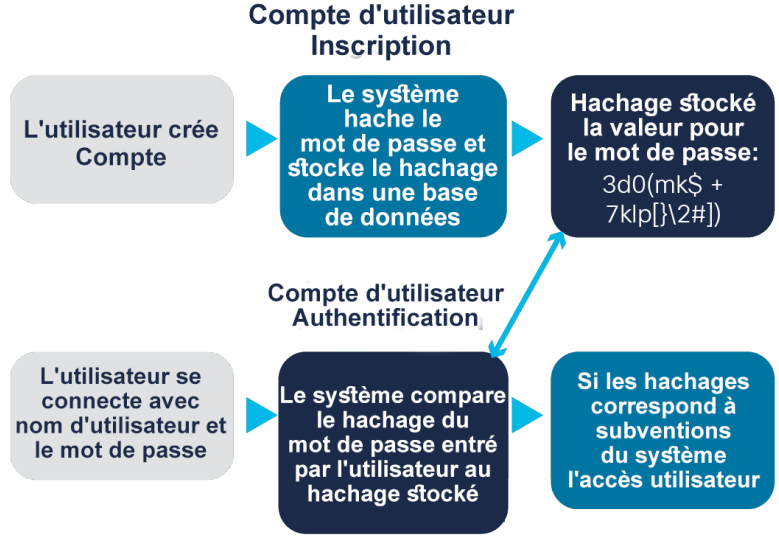
```
R1# verify / md5 flash:
c1900-universalk9-mz.SPA.154-3.M2.bin
.....
.....
.....
.....

.....MD5 of flash0:
c1900-universalk9-mz.SPA.154-3.M2.bin Done!
verify / md5 (flash0 : c1900-universalk9-mz . SPA .
154 - 3 . M@ . bin) -
61831a5669c7d46076901fbabd7687cd
```

Utilisation des hachages

Hash de mots de passe

- Les **algorithmes** de **hash transforment** n'importe quelle **quantité** de données en **hash numérique** ou **empreinte** de **longueur fixe**.
- Il est impossible pour un hacker d'inverser un hash numérique pour découvrir l'entrée d'origine.
- Si la saisie subit une quelconque modification, cela se traduit par un hash différent.
- Cela fonctionne pour **la protection** des **mots de passe.**
- **Un système** doit **stocker un mot de passe** sous **une forme** qui **le protège,** tout en conservant la **possibilité de vérifier** qu'il est **correct.**
- Ce diagramme montre le workflow pour le compte d'utilisateur l'enregistrement et l'authentification à l'aide d'un hash Système.
- Le système n'écrit jamais le mot de passe sur le disque dur ; il enregistre uniquement le hash numérique.
- Ainsi, le mot de passe n'est connu que de l'utilisateur qui l'a défini.



Utilisation des hachages

Piratage de hashs

- **Pour pirater un hash, le hacker doit deviner le mot de passe.**
- Les **deux méthodes** les plus utilisées pour trouver des mots de passe sont les **attaques** par force brute et les attaques par dictionnaire.
  - **Attaque dictionnaire**
    - Une attaque par dictionnaire utilise un fichier contenant des mots, des expressions et des mots de passe courants.
    - Le fichier a calculé les hachages de ces mots de passe courants.
    - Il compare les hachages dans le fichier avec les hachages de mot de passe et si un hachage correspond, l'attaquant connaîtra un groupe de mots de passe potentiellement bons utilisés dans ce système.
  - **Attaques par force brute**
    - Une attaque par force brute essaie toutes les combinaisons de caractères possibles jusqu'à une longueur donnée.
    - Une attaque par force brute nécessite beaucoup de temps et d'énergie au niveau du processeur.
    - En théorie, ce n'est qu'une question de temps avant que cette méthode ne détecte le mot de passe.

Utilisation des hachages

Salage

- Le salage est une **méthode** permettant de **renforcer** la **sécurité** des **mots** de **passe**.
- Sans salage, si deux utilisateurs possèdent le même mot de passe, ils auront également les mêmes hashs de mot de passe.
- Une valeur de **salage**, qui correspond à une **chaîne aléatoire de caractères**, est une entrée supplémentaire du mot de passe avant le hash.
- Cela crée un résultat de hachage différent, même lorsque les deux mots de passe sont identiques.
- Une **base de données stocke** le **hash** et la **valeur de salage.**
- La valeur salt ne doit pas nécessairement être secrète, car il s'agit d'un nombre aléatoire.

Utilisation des hachages

Mise en œuvre du salage

- Un **CSPRNG** (Cryptographically Secure Pseudo-Random Number Generator) est **l'outil idéal** pour générer une valeur salt.

- Les CSPRNG **génèrent** un **nombre aléatoire** à haut degré de **stochasticité** et entièrement **imprévisible** ; par conséquent, il est sécurisé sur le plan cryptographique.

- Les recommandations suivantes vous aideront à assurer une mise en œuvre réussie du salage :
  - La valeur **salt doit être unique** pour chaque mot de passe utilisateur.
  - Ne **jamais recycler** une valeur salt.
  - La **longueur** de la **valeur salt** doit être **égale** à celle de la **sortie** de la **fonction** de **hash**.
  - Toujours **effectuer** le **hash** sur le **serveur** dans une **application web**.

- L'étirement des clés **ralentit les tentatives de détection des mots de passe** et permet également de se protéger contre les attaques.

- Cela rend moins efficace le matériel haut de gamme capable de calculer des milliards de hashs par seconde.

Utilisation des hachages

Mise en œuvre du salage (suite)

- **Pour stocker un mot de passe :**
  - **Utilisez CSPRNG** afin de **générer** une valeur **salt aléatoire** longue.
  - Ajoutez la valeur **salt** au **début** du mot de passe.
  - Effectuez un **hash avec SHA-256**, une fonction de hash cryptographique standard.
  - **Enregistrez** la valeur **salt** et le **hash** dans le cadre de **l'enregistrement** de **base** de **données** de l'utilisateur.

- **Pour valider un mot de passe :**
  - **Extrayez** la valeur salt et le hash de la base de données.
  - **Ajoutez** la valeur salt au mot de passe et effectuez un hash avec la même fonction.
  - **Comparez** le hash du mot de passe fourni à celui stocké dans la base de données.
  - Si les hashs ne correspondent pas, le mot de passe est incorrect.

Utilisation des hachages

Prévention des attaques

- Le **salage empêche** un hacker d'utiliser une **attaque par dictionnaire** pour deviner les mots de passe et rend impossible l'utilisation de tables de recherche et de tables arc-en-ciel pour déchiffrer un hash.

Tables de correspondance	Une table de correspondance stocke les hashs de mots de passe précalculés dans un dictionnaire de mots de passe, accompagnés du mot de passe correspondant. Cette table est une structure de données qui traite des centaines de recherches de hashs par seconde.
Tables de correspondance inversées	Ce type d'attaque permet à un cybercriminel de lancer une attaque par force brute ou par dictionnaire sur de nombreux hashs sans la table de correspondance précalculée. Le cybercriminel crée une table de correspondance qui mappe chaque hash de mot de passe de la base de données des comptes compromise sur une liste d'utilisateurs. Le cybercriminel hashe chaque mot de passe supposé et utilise la table de correspondance pour obtenir une liste d'utilisateurs dont le mot de passe correspond à son hypothèse, comme illustré sur cette figure.
Rainbow tables	Les rainbow tables privilégient la réduction de la taille des tables de correspondance au détriment de la vitesse de piratage des hashs. Une table plus petite permet, en effet, de stocker les solutions d'un plus grand nombre de hashs dans un espace réduit.

4.5. Cryptographie à clé publique

Cryptographie à clé publique

Utilisation des signatures numériques

- Les **signatures numériques** sont une **technique mathématique** utilisée pour fournir **l’authenticité**, **l’intégrité** et la **non répudiation** et ont des propriétés spécifiques qui permettent l’authentification des entités et l’intégrité des données.

- En d'autres termes, la signature numérique constitue une **preuve légale de l'échange de données**.

- Les propriétés des signatures numériques sont les suivantes:
  - **Authentique -** La signature n'est pas falsifiable et prouve que le signataire a signé le document (et personne d'autre).
  - **Non modifié -** Une fois signé, un document ne peut pas être modifié.
  - **Non réutilisable -** La signature du document ne peut pas être transférée vers un autre document.
  - **Non répudié -** le document signé est identique à un document physique, car la signature est la preuve que le document a été signé par la personne réelle.

Utilisation des signatures numériques (suite)

- Les signatures numériques sont couramment utilisées dans les deux situations suivantes :

  - **Signature de code**
    - Elle est utilisée à des fins d'intégrité et d'authentification des données.
    - permet de vérifier l'intégrité des fichiers exécutables téléchargés à partir du site web d'un fournisseur.
    - Elle utilise également des certificats numériques pour authentifier et vérifier l'identité du site qui est la source des fichiers..

  - **Certificats numériques**
    - Ils sont semblables à une carte d'identité virtuelle et servent à authentifier l'identité d'un système avec site web fournisseur et à établir une connexion chiffrée pour l'échange de données confidentielles.

Cryptographie à clé publique

Utilisation des signatures numériques (suite)

- Trois algorithmes DSS (Digital Signature Standard) sont utilisés pour générer et vérifier les signatures numériques :

  - **Digital Signature Algorithm (DSA)** : l'algorithme DSA constitue le standard d'origine pour la génération de paires de clés publiques et privées, comme pour la génération et la vérification de signatures numériques.

  - Algorithme Rivest-Shamir-Adelman (RSA) **: l'algorithme RSA est un algorithme asymétrique couramment utilisé pour la génération et la vérification de signatures numériques.**

  - **Elliptic Curve Digital Signature Algorithm (ECDSA)** : l'algorithme ECDSA est une nouvelle variante de l'algorithme DSA. Il assure l'authentification et la non-répudiation des signatures numériques tout en offrant de nouveaux bénéfices en termes d'efficacité informatique, de petites signatures et de bande passante minimale.

Cryptographie à clé publique

Signatures numériques de signature de code

- Les signatures numériques sont couramment utilisées pour garantir **l'authenticité et l'intégrité du code logiciel**.

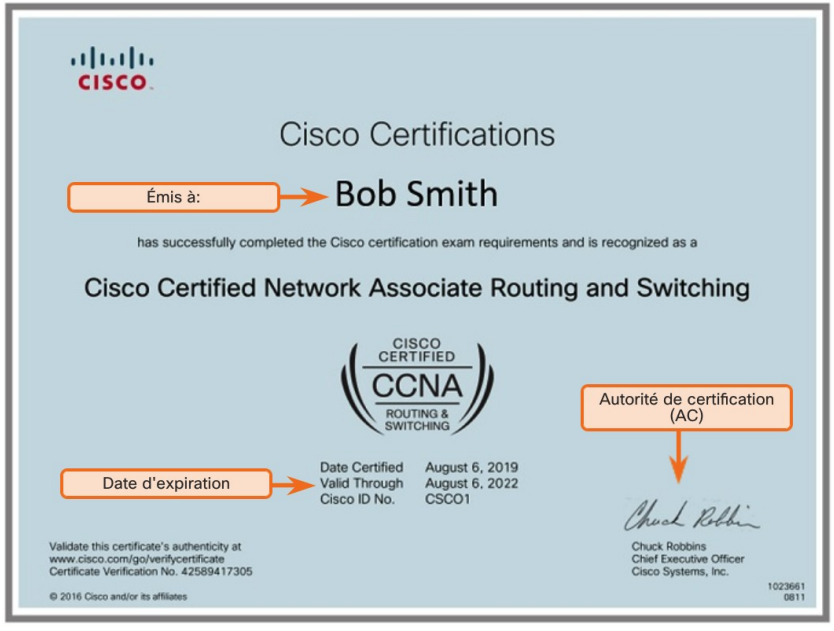
- Les **fichiers exécutables sont encapsulés dans une enveloppe signée numériquement**, ce qui permet à l'utilisateur final de vérifier la signature avant d'installer le logiciel.

- La signature numérique du code offre plusieurs garanties sur le code :
  - Le code est **authentique** et provient de l'éditeur.
  - Le code **n'a pas été modifié** depuis qu'il a quitté l'éditeur de logiciels.
  - C'est indéniablement l'éditeur qui a publié le code.
    - Cela assure la **non-répudiation** de l'acte de publication.

Cryptographie à clé publique

Signatures numériques de certificat numérique

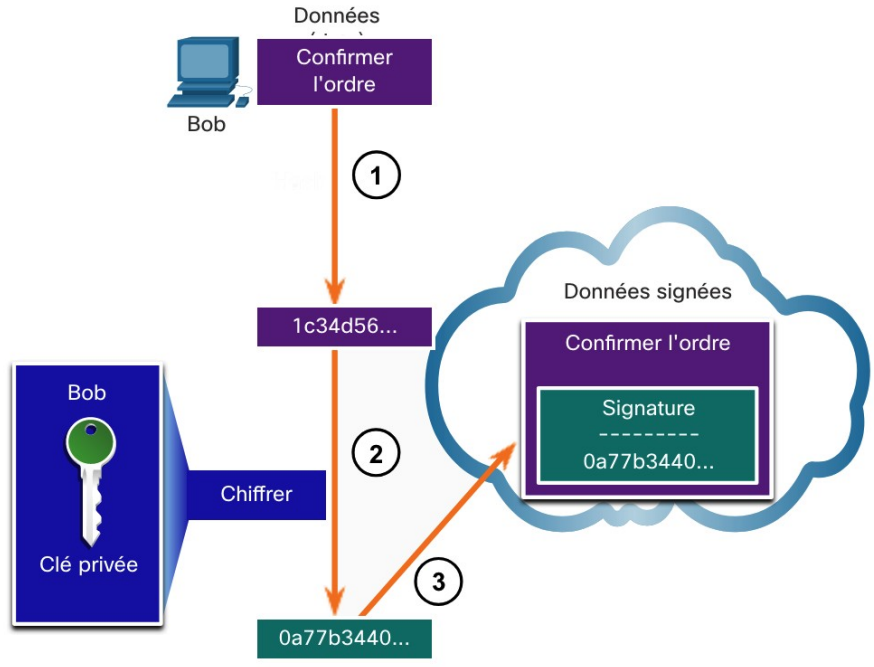
- Les **certificats numériques** permettent aux utilisateurs, aux hôtes et aux entreprises d'échanger des informations sur Internet de manière sécurisée.
- Un certificat numérique permet d'authentifier et de vérifier que l'expéditeur d'un message est bien celui qu'il prétend être.
- Les certificats numériques permettent également de garantir la confidentialité du destinataire en lui permettant de chiffrer sa réponse.
- Ce sont comme des certificats physiques.
- Par exemple, le certificat sur papier illustré à la figure identifie le destinataire, l'autorité de certification et la durée de validité du certificat.



Cryptographie à clé publique

Signatures numériques pour les certificats numériques (suite)

- Ce scénario vous aidera à comprendre comment une signature numérique est utilisée.
  - Dans ce scénario, Bob confirme une commande auprès d'Alice.
  - Alice commande sur le site Web de Bob.
  - Alice s'est connectée au site Web de Bob, et une fois que le certificat a été vérifié, le certificat de Bob est stocké sur le site Web d'Alice.
  - La clé publique est utilisée pour vérifier la signature numérique de Bob.



Cryptographie à clé publique

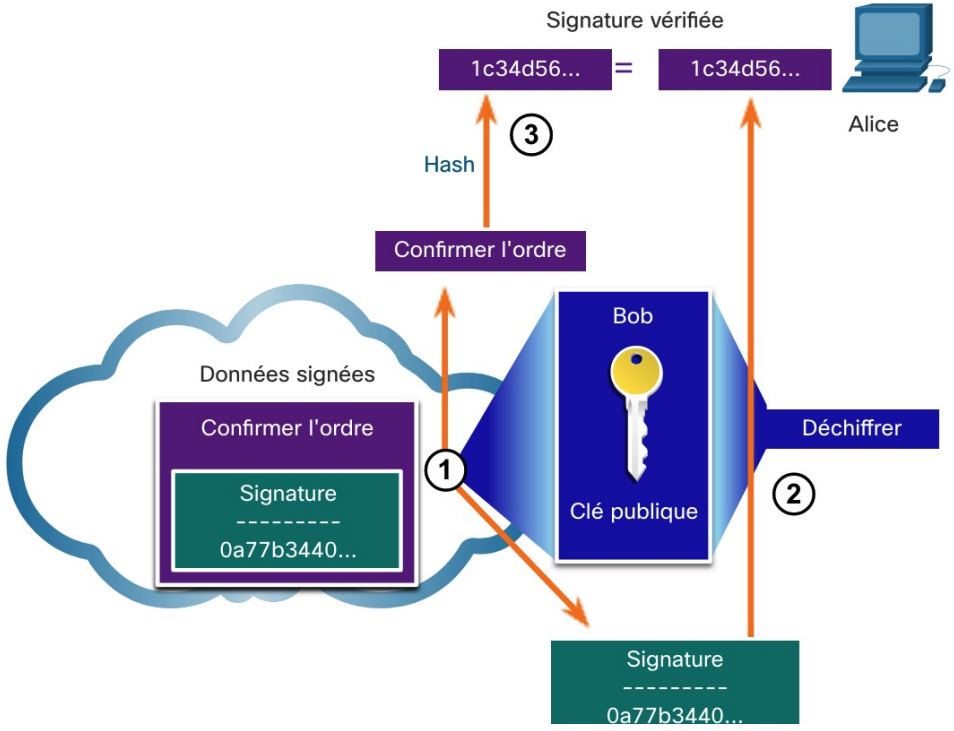
Signatures numériques pour les certificats numériques (suite)

- Lorsque Alice reçoit la signature numérique, le processus suivant se produit.

1. Le terminal d'Alice accepte la confirmation de commande avec la signature numérique et obtient la clé publique de Bob.

2. L'ordinateur d'Alice déchiffre la signature à l'aide de la clé publique de Bob. Cette étape révèle la valeur de hash présumé de l'appareil émetteur.

3. L'ordinateur d'Alice crée un hash du document reçu sans sa signature et compare ce hash au hash de la signature déchiffrée.Si les hashs correspondent, le message est considéré comme authentique. Cela signifie qu'il a effectivement été envoyé par Bob et qu'il n'a pas été modifié depuis la dernière signature.



4.8 Résumé de la quatrième partie

Résumé de la quatrième partie

Résumé

- Deux classes de chiffrement permettent de garantir la confidentialité des données symétrique et asymétrique.
- Les algorithmes de chiffrement symétrique tels que (DES), 3DES et Advanced Encryption Standard (AES) sont basés sur l'hypothèse que chaque partie communicante connaît la clé pré-partagée.
- Les algorithmes asymétriques, notamment RSA, est conçu pour utiliser des clés différentes pour le chiffrement et le déchiffrement, ainsi qu'une clé publique et une clé privée, ce qui assure la confidentialité sans pré-partage de mot de passe.
- L'échange de clé DH est un algorithme mathématique asymétrique qui permet à deux ordinateurs de générer un secret partagé identique, et cela sans avoir communiqué auparavant.
- La stéganographie dissimule les données (le message) dans un autre fichier, comme un graphique, un fichier audio ou un autre fichier texte.
- Les quatre éléments des communications sécurisées sont l'intégrité des données, l'authentification d'origine, la confidentialité des données et la non-répudiation des données.

Résumé de la quatrième partie

Résumé

- Les fonctions de hachage sont des fonctions unidirectionnelles utilisées pour vérifier et assurer l’intégrité des données et sont vulnérables aux attaques MiTM.
- Une fonction de hash prend un bloc variable de données binaires (le message) et produit une représentation condensée de longueur fixe (le hash).
- Il existe quatre fonctions de hachage bien connues: MD5 avec digest 128 bits, SHA-1, SHA-2 et SHA-3.
- Les deux principales attaques utilisées sont les attaques par dictionnaire et par force brute.
- Un salt est une entrée supplémentaire ajoutée au mot de passe, créant un résultat de hachage différent même lorsque les deux mots de passe sont identiques.
- La signature numérique est une technique mathématique utilisée pour assurer l'authenticité, l'intégrité et la non-répudiation sous la forme de certificats numériques et de signature de code.
- Trois algorithmes DSS sont utilisés pour générer et vérifier les signatures numériques : DSA, RSA et ECDSA.
- Lorsqu'ils établissent une connexion sécurisée entre deux hôtes, les hôtes échangent leurs informations de clé publique.