Groupe de travail Réseau
Request for Comments: 5656
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

D. Stebila, Queensland University of Technology
J. Green, Queen's University
décembre 2009

Intégration d'algorithme de courbe elliptique dans la couche de transport Secure Shell

Résumé
Le présent document décrit les algorithmes fondés sur le chiffrement par courbe elliptique (ECC, *Elliptic Curve Cryptography*) à utiliser dans le protocole de transport Secure Shell (SSH). En particulier, il spécifie l'accord de clé de courbe elliptique Diffie-Hellman (ECDH, *Elliptic Curve Diffie-Hellman*) de courbe elliptique Menezes-Qu-Vanstone (ECMQV, *Elliptic Curve Menezes-Qu-Vanstone*) et l'algorithme de signature numérique à courbe elliptique (ECDSA, *Elliptic Curve Digital Signature Algorithm*) à utiliser dans le protocole de couche de transport SSH.

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Table des matières

1. Introduction

Le présent document ajoute les algorithmes de chiffrement à courbe elliptique suivants à l'arsenal de Secure Shell : courbe elliptique Diffie-Hellman (ECDH, *Elliptic Curve Diffie-Hellman*) et algorithme de signature numérique à courbe elliptique (ECDSA, *Elliptic Curve Digital Signature Algorithm*) ainsi que l'utilisation de la famille SHA2 d'algorithmes de hachage sûrs. De plus, est fournie la prise en charge des courbes elliptiques Menezes-Qu-Vanstone (ECMQV, *Elliptic Curve Menezes-Qu-Vanstone*).

Du fait de la petite taille de ses clés et de son inclusion dans la suite B de l'Agence Nationale de Sécurité (des USA) le chiffrement par courbe elliptique (ECC, *Elliptic Curve Cryptography*) devient un système de chiffrement à clé publique largement utilisé et attrayant.

Comparé aux systèmes de chiffrement comme RSA, l'algorithme de signature numérique (DSA, *Digital Signature Algorithm*) et l'échange de clés Diffie-Hellman (DH) les variantes de ECC sur ces schémas offrent une sécurité équivalente avec de plus petites tailles de clé. Ceci est illustré dans le tableau suivant, fondé sur le paragraphe 5.6.1 du document NIST 800-57 [NIST-800-57], qui donne des tailles de clé approximativement comparables pour les systèmes de chiffrement à clé symétrique et asymétrique fondés sur les algorithmes les plus connus pour les attaquer. L est la taille de champ et N est la taille de sous champ.

Symétrique	**Log discret (par exemple, DSA, DH)**	**RSA**	**ECC**
80	L = 1024, N = 160	1024	160-223
112	L = 2048, N = 256	2048	224-255
128	L = 3072, N = 256	3072	256-383
192	L = 7680, N = 384	7680	384-511
256	L = 15360, N = 512	15360	512+

La mise en œuvre de cette spécification requiert une certaine familiarité avec SSH [RFC4251], [RFC4253], [RFC4250] et ECC [SEC1] (des informations supplémentaires sur ECC sont disponibles dans [HMV04], [ANSI-X9.62], et [ANSI-X9.63]).

Le présent document est concerné par les détails de mise en œuvre de SSH ; la spécification des algorithmes de chiffrement sous-jacents est donnée dans d'autres documents de normalisation.

2. Notation

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Les types de données booléen, octet, uint32, uint64, chaîne, et mpint sont à interpréter dans ce document comme décrit dans la [RFC4251].

La taille d'un ensemble de paramètres de domaine de courbe elliptique sur une courbe première est définie comme le nombre de bits dans la représentation binaire de l'ordre des champs, généralement notée p. La taille sur une courbe de caractéristique 2 est définie comme le nombre de bits dans la représentation binaire du champ, généralement notée m. Un ensemble de paramètres de domaine de courbe elliptique définit un groupe d'ordre n généré par un point de base P.

3. Algorithme de clé publique ECC SSH

L'algorithme de clé publique ECC SSH est défini par son format de clé, l'algorithme de signature ECDSA correspondant, le codage de signature, et les identifiants d'algorithme.

Cette section définit la famille de formats de clé publique "ecdsa-sha2-*" et des formats de signature correspondants. Chaque mise en œuvre conforme de SSH ECC DOIT mettre en œuvre ce format de clé publique.

3.1 Format de clé

Les formats de clé "ecdsa-sha2-*" ont tous le codage suivant :

```
chaîne "ecdsa-sha2-[identifiant]"
  octet[n]  ecc_key_blob
```

La valeur ecc_key_blob a le codage spécifique suivant :

```
chaîne [identifiant]
  chaîne   Q
```

La chaîne [identifiant] est l'identifiant des paramètres de domaine de courbe elliptique. Le format de cette chaîne est spécifié au paragraphe 6.1. Les informations sur les ensembles EXIGÉS et RECOMMANDÉS de paramètres de domaine de courbe elliptique à utiliser avec cet algorithme se trouvent à la Section 10.

Q est la clé publique codée à partir d'un point de courbe elliptique dans une chaîne d'octets comme défini au paragraphe 2.3.3 de [SEC1] ; une compression de point PEUT être utilisée.

L'algorithme pour la génération de clé ECC se trouve au paragraphe 3.2 de [SEC1]. Connaissant certains paramètres de domaine de courbe elliptique, une paire de clé ECC peut être générée, qui contient une clé privée (un entier d) et une clé publique (un point Q de la courbe elliptique).

3.1.1 Algorithme de signature

La signature et la vérification sont faites en utilisant l'algorithme de signature numérique à courbe elliptique (ECDSA, *Elliptic Curve Digital Signature Algorithm*). ECDSA est spécifié dans [SEC1]. L'algorithme de hachage de message DOIT être de la famille de fonctions de hachage SHA2 [FIPS-180-3] et est choisi en accord avec la taille de courbe comme spécifié au paragraphe 6.2.1.

3.1.2 Codage de signature

Les signatures sont codées comme suit :

```
chaîne  "ecdsa-sha2-[identifiant]"
   chaîne   ecdsa_signature_blob
```

La chaîne [identifiant] est l'identifiant des paramètres de domaine de courbe elliptique. Le format de cette chaîne est spécifié au paragraphe 6.1. Les informations sur les ensembles EXIGÉS et RECOMMANDÉS de paramètres de domaine de courbe elliptique à utiliser avec cet algorithme se trouvent à la Section 10.

La valeur ecdsa_signature_blob a le codage spécifique suivant :

```
   mpint   r
   mpint   s
```

Les entiers r et s sont le résultat de l'algorithme ECDSA.

La largeur des champs d'entier est déterminée par la courbe utilisée. Noter que les entiers r et s sont des entiers modulo l'ordre du sous groupe cryptographique, qui peut être plus grand que la taille du champ fini.

4. Échange de clés ECDH

La méthode d'échange de clés de courbe elliptique Diffie-Hellman (ECDH) génère un secret partagé provenant d'une clé privée à courbe elliptique éphémère locale et d'une clé publique à courbe elliptique éphémère distante. Cette méthode d'échange de clés fournit une authentification explicite du serveur comme défini dans la [RFC4253] en utilisant une signature sur le hachage de l'échange. Chaque mise en œuvre conforme de SSH ECC DOIT mettre en œuvre l'échange de clés ECDH.

La primitive utilisée pour la génération de clé partagée est ECDH avec multiplication de cofacteur, dont la spécification complète se trouve au paragraphe 3.3.2 de [SEC1]. L'algorithme pour la génération de la paire de clés se trouve au paragraphe 3.2.1 de [SEC1].

La famille de noms de méthode d'échange de clés définie pour être utilisée avec cet échange de clé se trouve au paragraphe 6.3. La négociation d'algorithme choisit l'algorithme de clé publique à utiliser pour signer et le nom de la méthode de l'échange de clés. Le nom de la méthode de l'échange de clé choisie détermine les paramètres de domaine de courbe elliptique et la fonction de hachage à utiliser dans le reste de ce paragraphe.

Les informations sur les paramètres de domaine de courbe elliptique EXIGÉE et RECOMMANDÉE à utiliser avec cette méthode se trouvent à la Section 10.

Toutes les clés publiques à courbe elliptique DOIVENT être validées après leur réception. Un exemple d'algorithme de validation se trouve au paragraphe 3.2.2 de [SEC1]. Si une clé échoue à la validation, l'échange de clés DOIT échouer.

Les clés publiques de courbe elliptique (points) qui doivent être transmises sont codées en chaînes d'octets avant leur transmission. La transformation entre points de courbe elliptique et chaînes d'octets est spécifiée aux paragraphes 2.3.3 et 2.3.4 de [SEC1] ; la compression de point PEUT être utilisée. Le résultat de la génération de clé partagée est un élément de champ xp. Le cadre de SSH exige que la clé partagée soit un entier. La conversion entre un élément de champ et un entier est spécifiée au paragraphe 2.3.9 de [SEC1].

La spécification des numéros de message SSH_MSG_KEX_ECDH_INIT et SSH_MSG_KEX_ECDH_REPLY se trouve à la Section 7.

Voici un résumé du processus d'échange de clés :

Client **Serveur**
Génère une paire de clés éphémères.
SSH_MSG_KEX_ECDH_INIT ---------->

Vérifie que la clé reçue est valide.
Génère une paire de clé éphémères.
Calcule le secret partagé.
Génère et signe le hachage de l'échange.
<---------- SSH_MSG_KEX_ECDH_REPLY

Vérifie que la clé reçue est valide.
*Vérifie que la clé d'hôte appartient au serveur.
Calcule le secret partagé.
Génère le hachage de l'échange.
Vérifie la signature du serveur.

* Il est RECOMMANDÉ que le client vérifie que la clé d'hôte envoyée est la clé d'hôte du serveur (par exemple, en utilisant une base de données locale). Le client PEUT accepter la clé d'hôte sans vérification, mais le faire va rendre le protocole non sûr contre des attaques actives ; voir la discussion du paragraphe 4.1 de la [RFC4251].

Ceci est mis en œuvre en utilisant les messages suivants.

Le client envoie :

octet SSH_MSG_KEX_ECDH_INIT
chaîne Q_C, chaîne d'octets de clé publique éphémère du client

Le serveur répond avec:

octet SSH_MSG_KEX_ECDH_REPLY
chaîne K_S, clé publique d'hôte du serveur
chaîne Q_S, chaîne d'octets de clé publique éphémère du serveur
chaîne signature sur le hachage de l'échange

Le hachage de l'échange H est calculé comme le hachage de l'enchaînement de ce qui suit :

chaîne V_C, chaîne d'identification du client (CR et LF exclus)
chaîne V_S, chaîne d'identification du serveur (CR et LF exclus)
chaîne I_C, charge utile du message SSH_MSG_KEXINIT du client
chaîne I_S, charge utile du message SSH_MSG_KEXINIT du serveur
chaîne K_S, clé publique d'hôte du serveur
chaîne Q_C, chaîne d'octets de clé publique éphémère du client
chaîne Q_S, chaîne d'octets de clé publique éphémère du serveur
mpint K, secret partagé

5. Échange de clés ECMQV

L'algorithme d'échange de clé à courbe elliptique Menezes-Qu-Vanstone (ECMQV) génère un secret partagé provenant de deux paires de clés à courbe elliptique locales et de deux clés publiques distantes. Cette méthode d'échange de clés fournit une authentification implicite du serveur comme défini dans la [RFC4253]. La méthode d'échange de clés ECMQV est FACULTATIVE.

Le nom de méthode d'échange de clés définie pour être utilisée avec cet échange de clés est "ecmqv-sha2". Ce nom de méthode donne un algorithme de hachage à utiliser pour le code d'authentification de message haché (HMAC, *Hashed Message Authentication Code*) ci-dessous. De futures RFC pourront définir de nouveaux noms de méthodes qui spécifient de nouveaux algorithmes de hachage à utiliser avec ECMQV. Plus d'informations sur le nom de méthode et HMAC se trouvent au paragraphe 6.4.

En général, l'échange de clés ECMQV est effectué en utilisant la paire de clés éphémères et à long terme du client et du serveur, ce qui fait un total de 4 clés. Dans le cadre de SSH, le client n'a pas de paire de clés à long terme qui doive être authentifiée. Donc, on génère une clé éphémère et on utilise cela comme les deux clé des clients. Ceci est plus efficace que d'utiliser deux clés éphémères différentes, et cela n'affecte pas la sécurité (c'est analogue au protocole en une passe du paragraphe 6.1 de [LMQSV98]).

Une description complète de la primitive ECMQV se trouve au paragraphe 3.4 de [SEC1]. L'algorithme pour la génération de paire de clé se trouve au paragraphe 3.2.1 de [SEC1].

Durant la négociation d'algorithme avec les messages SSH_MSG_KEXINIT, la méthode d'échange de clés ECMQV peut seulement être choisie si un algorithme de clé publique qui prend en charge les clés d'hôte ECC peut aussi être choisi. Cela est dû à l'utilisation de l'authentification implicite de serveur dans cette méthode d'échange de clés. Ce cas est traité de la même façon que sont traitées les méthodes d'échange de clés qui exigent des algorithmes de clé publique capables de chiffrement/signature au paragraphe 7.1 de la [RFC4253]. Si l'échange de clés ECMQV est choisi, alors l'algorithme de clé publique qui prend en charge les clés d'hôte ECC DOIT aussi être choisi.

ECMQV exige que toutes les clés utilisées pour générer un secret partagé soient générées sur les mêmes paramètres de domaine de courbe elliptique. Comme la clé d'hôte est utilisée dans la génération du secret partagé, permettant l'authentification implicite du serveur, les paramètres de domaine associés à la clé d'hôte sont utilisés dans cette section.

Toutes les clés publiques à courbe elliptique DOIVENT être validées après leur réception. Un exemple d'algorithme de validation se trouve au paragraphe 3.2.2 de [SEC1]. Si une clé échoue à la validation, l'échange de clés DOIT échouer.

Les clés publiques éphémères de courbe elliptique (points) qui doivent être transmises sont codées en chaînes d'octets avant d'être transmises. La transformation entre points de courbe elliptique et chaînes d'octets est spécifiée aux paragraphes 2.3.3 et 2.3.4 de [SEC1] ; la compression de points PEUT être utilisée. Le résultat de la génération de clé partagée est un élément de champ xp. Le cadre SSH exige que la clé partagée soit un entier. La conversion entre un élément de champ et un entier est spécifiée au paragraphe 2.3.9 de [SEC1].

Voici un résumé du processus d'échange de clés :

Client **Serveur**
Génère la paire de clés éphémères.
SSH_MSG_KEX_ECMQV_INIT ---------->

Vérifie que la clé reçue est valide.
Génère une paire de clés éphémère.
Calcule le secret partagé.
Génère le hachage de l'échange et calcule
le HMAC sur lui en utilisant le secret partagé.
<---------- SSH_MSG_KEX_ECMQV_REPLY

Vérifie que les clés reçues sont valides.
*Vérifie que la clé d'hôte appartient au serveur.
Calcule le secret partagé.
Vérifie le HMAC.

* Il est RECOMMANDÉ que le client vérifie que la clé d'hôte envoyée est la clé d'hôte du serveur (par exemple, en utilisant une base de données locale). Le client PEUT accepter la clé d'hôte sans vérification, mais le faire rend le protocole non sûr contre des attaques actives.

La spécification des numéros de message SSH_MSG_ECMQV_INIT et SSH_MSG_ECMQV_REPLY se trouve à la Section 7.

Cet algorithme d'échange de clé est mis en œuvre avec les messages suivants :

Le client envoie :
octet SSH_MSG_ECMQV_INIT
chaîne Q_C, chaîne d'octets de clé publique éphémère du client

Le serveur envoie :
octet SSH_MSG_ECMQV_REPLY
chaîne K_S, clé publique d'hôte du serveur
chaîne Q_S, chaîne d'octets de clé publique éphémère du serveur
chaîne étiquette HMAC calculée sur H en utilisant le secret partagé

Le hachage H est formé en appliquant l'algorithme HASH sur un enchaînement de ce qui suit :
chaîne V_C, chaîne d'identification du client (CR et LF exclus)
chaîne V_S, chaîne d'identification du serveur (CR et LF exclus)
chaîne I_C, charge utile du message SSH_MSG_KEXINIT du client
chaîne I_S, charge utile du message SSH_MSG_KEXINIT du serveur
chaîne K_S, clé publique d'hôte du serveur
chaîne Q_C, chaîne d'octets de clé publique éphémère du client
chaîne Q_S, chaîne d'octets de clé publique éphémère du serveur
mpint K, secret partagé

6. Noms de méthodes

Le présent document définit une nouvelle famille de noms de méthode d'échange de clés, un nouveau nom de méthode d'échange de clés, et une nouvelle famille de noms d'algorithme de clé publique dans le registre de noms SSH.

6.1 Identifiants de paramètre de domaine de courbe elliptique

Ce paragraphe spécifie les identifiants de codage de paramètre de domaine de courbe elliptique désignés. Ces identifiants sont utilisés dans le présent document pour identifier la courbe utilisée dans le format de clé publique SSH ECC, le module de signature ECDSA, et le nom de méthode ECDH.

Pour les courbes elliptiques EXIGÉES nistp256, nistp384, et nistp521, les identifiants de codage de paramètre de domaine de courbe elliptique sont les chaînes "nistp256", "nistp384", et "nistp521".

Pour toutes les autres courbes elliptiques, y compris toutes les autres courbes du NIST et toutes les autres courbes RECOMMANDÉES, l'identifiant de paramètre de domaine de courbe elliptique est la représentation ASCII décimale séparée par des points de l'identifiant d'objet (OID, *Object Identifier*) en notation numéro un de syntaxe abstraite (ASN.1, *Abstract Syntax Notation One*) [X.680] des paramètres de domaine de courbe désignée qui sont associés aux clés d'hôte EEC du serveur. Cet identifiant est défini pourvu que l'enchaînement de l'identifiant de format de la clé publique et de l'identifiant de paramètre de domaine de courbe elliptique (ou le nom de méthode et l'identifiant de paramètre de domaine de courbe elliptique) n'excède pas le maximum spécifié par l'architecture de protocole SSH [RFC4251], à savoir 64 caractères ; autrement, l'identifiant pour cette courbe est indéfini, et la courbe n'est pas prise en charge par la présente spécification.

Une liste des courbes EXIGÉES et RECOMMANDÉES et de leurs OID se trouve à la Section 10.

Noter que les mises en œuvre DOIVENT utiliser les identifiants de chaîne pour les trois courbes du NIST EXIGÉES, même quand un OID existe pour cette courbe.

6.2 Algorithme de clé publique ECC (ecdsa-sha2-*)

L'algorithme de clé publique SSH ECC est spécifié par une famille d'identifiants de format de clé publique. Chaque identifiant est l'enchaînement de la chaîne "ecdsa-sha2-" avec l'identifiant de paramètre de domaine de courbe elliptique comme défini au paragraphe 6.1. Une liste des courbes exigées et recommandées et de leurs OID se trouve à la Section 10.

Par exemple, le nom de méthode pour l'échange de clé ECDH avec les clés éphémères générées sur la courbe nistp256 est "ecdsa-sha2-nistp256".

6.2.1 Algorithme de signature numérique à courbe elliptique

L'algorithme de signature numérique à courbe elliptique (ECDSA, *Elliptic Curve Digital Signature Algorithm*) est spécifié pour être utilisé avec l'algorithme de clé publique SSH ECC.

L'algorithme de hachage défini par cette famille de noms de méthode est la famille SHA2 d'algorithmes de hachage [FIPS-180-3]. L'algorithme de la famille SHA2 qui va être utilisé est choisi sur la base de la taille de la courbe spécifiée dans la clé publique:

Taille de courbe	**Algorithme de hachage**
$b \leq 256$	SHA-256
$256 < b \leq 384$	SHA-384
$384 < b$	SHA-512

6.3 Noms de méthode d'échange de clé ECDH (ecdh-sha2-*)

L'échange de clés de courbe elliptique Diffie-Hellman (ECDH) est défini par une famille de noms de méthodes. Chaque nom de méthode est l'enchaînement de la chaîne "ecdh-sha2-" avec l'identifiant de paramètre de domaine de courbe elliptique comme défini au paragraphe 6.1. Une liste des courbes exigées et recommandées et de leurs OID se trouve à la Section 10.

Par exemple, le nom de méthode pour l'échange de clés ECDH avec des clés éphémères générées sur la courbe sect409k1 est "ecdh-sha2-1.3.132.0.36".

L'algorithme de hachage défini par cette famille de noms de méthodes est la famille SHA2 d'algorithmes de hachage [FIPS-180-3]. L'algorithme de hachage est défini dans le nom de méthode pour laisser de la place pour d'autres algorithmes à

définir dans de futurs documents. L'algorithme de la famille SHA2 qui va être utilisé est choisi sur la base de la taille de la courbe spécifiée dans le nom de méthode conformément au tableau du paragraphe 6.2.1.

L'enchaînement de tout OID codé en ASN.1 spécifiant un ensemble de paramètres de domaine de courbe elliptique avec "ecdh-sha2-" est implicitement enregistré dans la présente spécification.

6.4 Nom de méthode d'échange et de vérification de clé ECMQV (ecmqv-sha2)

L'échange de clés de courbe elliptique Menezes-Qu-Vanstone (ECMQV) est défini par le nom de méthode "ecmqv-sha2". À la différence de la méthode d'échange de clés ECDH, ECMQV s'appuie sur un algorithme de clé publique qui utilise des clés ECC : il n'a pas besoin d'une famille de noms de méthode parce que les informations de courbe peuvent être obtenues de l'algorithme de clé publique.

Les algorithmes de hachage et de code d'authentification de message sont définis par le nom de méthode pour laisser de la place pour que d'autres algorithmes soient définis avec ECMQV dans de futurs documents.

L'algorithme de hachage défini par ce nom de méthode est la famille SHA2 d'algorithmes de hachage [FIPS-180-3]. L'algorithme provenant de la famille SHA2 qui va être utilisé est choisi sur la base de la taille de la courbe spécifiée pour être utilisée avec ECMQV par l'algorithme de clé publique choisi en accord avec le tableau du paragraphe 6.2.1.

Le code d'authentification de message chiffré-haché qui est utilisé pour identifier le serveur et vérifier les communications est fondé sur le hachage choisi ci-dessus. Les informations sur qui met en œuvre le HMAC sur la base de l'algorithme de hachage choisi se trouvent dans la [RFC2104].

7. Messages d'échange de clés

Les numéros de message de 30 à 49 sont spécifiques de l'échange de clé et dans un espace de noms privé défini dans la [RFC4250] qui peut être redéfini par toute méthode d'échange de clés de la [RFC4253] sans exiger de processus d'enregistrement de la part de l'IANA.

Les numéros de message suivants ont été définis dans le présent document:

7.1 Numéros de message ECDH

#define SSH_MSG_KEX_ECDH_INIT : 30
#define SSH_MSG_KEX_ECDH_REPLY : 31

7.2 Numéros de message ECMQV

#define SSH_MSG_ECMQV_INIT : 30
#define SSH_MSG_ECMQV_REPLY : 31

8. Considérations de gestion

Comme le présent document fournit seulement de nouveaux algorithmes de clé publique et méthodes d'échange de clés au sein de l'architecture existante du protocole Secure Shell, il y a peu de considérations de gestion au delà de celles qui s'appliquent aux mises en œuvre existantes de Secure Shell. Des considérations de gestion supplémentaires sont données ci-dessous.

8.1 Contrôle de fonction par configuration et politique

La Section 10 spécifie les paramètres EXIGÉS et RECOMMANDÉS de domaine de courbe elliptique à utiliser avec les algorithmes de clé publique et méthodes d'échange de clés définis dans le présent document. Les mises en œuvre DEVRAIENT permettre aux administrateurs de système de désactiver certaines courbes, y compris des courbes EXIGÉES ou RECOMMANDÉES, pour satisfaire la politique de sécurité locale.

8.2 Impact sur le fonctionnement du réseau

Comme le présent document fournit une nouvelle fonction au sein de l'architecture du protocole Secure Shell, le seul impact sur le fonctionnement du réseau est l'impact sur les mises en œuvre existantes de Secure Shell. Le protocole Secure Shell fournit des mécanismes de négociation pour les algorithmes de clé publique et méthodes d'échange de clés : toute mise en œuvre qui ne reconnaît pas les algorithmes et méthodes définis dans le présent document va les ignorer dans la négociation et utiliser le prochain algorithme ou méthode mutuellement pris en charge, ne causant aucun impact négatif sur la rétro compatibilité.

L'utilisation du chiffrement à courbe elliptique ne devrait pas faire peser une charge de calcul significative sur un serveur qui le met en œuvre. En fait, grâce à sa plus petite taille de clé, le chiffrement à courbe elliptique peut être mis en œuvre plus efficacement pour le même niveau de sécurité que RSA, Diffie-Hellman à champ fini, ou DSA.

9. Considérations sur la sécurité

Le présent document fournit de nouveaux algorithmes de clé publique et de nouvelles méthodes d'accord de clés pour le protocole Secure Shell. Pour leur plus grande partie, les considérations sur la sécurité impliquées par l'utilisation du protocole Secure Shell s'appliquent. De plus, les mises en œuvre devraient avoir connaissance des considérations sur la sécurité spécifiques du chiffrement à courbe elliptique.

Pour les trois classes de fonctionnalités ajoutées par le présent document (les algorithmes de clé publique impliquant ECDSA, l'échange de clés impliquant ECDH, et l'échange de clé authentifié impliquant ECMQV) la meilleure technique connue actuellement pour casser les cryptosystèmes est de résoudre le problème du logarithme discret de la courbe elliptique (ECDLP, *elliptic curve discrete logarithm problem*).

La difficulté de casser l'ECDLP dépend de la taille et de la qualité des paramètres de la courbe elliptique. Certains types de courbes peuvent être plus susceptibles que d'autres d'attaques connues. Par exemple, les courbes sur des champs GF(2^m) finis, où m est composite, peuvent être susceptibles d'une attaque fondée sur la descente de Weil. Toutes les courbes RECOMMANDÉES à la Section 10 n'ont pas ce problème. Les administrateurs de système devraient faire attention quand ils activent des courbes autres que celles spécifiées à la Section 10 et devraient faire une étude plus détaillée de la sécurité de la courbe en question.

Il est estimé (voir, par exemple, le paragraphe B.2.1 de [SEC1]) que quand les paramètres d'une courbe sont générés au hasard, les courbes vont être résistantes à des attaques particulières, et doivent s'appuyer sur les attaques les plus générales. Les courbes EXIGÉES à la Section 10 ont toutes été générées pseudo aléatoirement de façon vérifiable. Le moment de lancement des attaques générales dépend de l'algorithme utilisé. À présent, l'algorithme le mieux connu est la méthode Pollard-rho. (L'algorithme de Shor pour ordinateurs quantiques peut résoudre le ECDLP en temps polynomial, mais à présent des ordinateurs quantiques à grande échelle n'ont pas été construits et des travaux expérimentaux significatifs en physique et en ingénierie devront être effectués avant que des ordinateurs quantiques à grande échelle puissent être construits. Il n'y a pas d'estimation ferme du moment où cela pourrait arriver, mais on pense généralement que c'est au moins dans 20 ans (de la date de la rédaction).

Sur la base de ces projections de la puissance de calcul, il est possible d'estimer le moment des meilleures attaques connues sur la base de la taille de champ fini. Le tableau de la Section 1 donne une estimation de l'équivalence entre taille de champ de courbe elliptique et taille de clé symétrique. En gros, une courbe elliptique à N bits offre la même sécurité qu'un chiffrement symétrique à N/2 bits, de sorte qu'une courbe elliptique à 256 bits (comme la courbe EXIGÉE nistp256) convient pour être utilisée avec AES à 128 bit, par exemple.

Les estimations considèrent que 2^80-2^90 opérations sont au-delà du faisable, de sorte que cela suggérerait d'utiliser des courbes elliptiques d'au moins 160 à 180 bits. Les courbes EXIGÉES dans ce document sont des courbes à 256, 384, et 521 bits ; les mises en œuvre NE DEVRAIENT PAS utiliser de courbes plus petites que 160 bits.

Une discussion détaillée des considérations sur la sécurité des paramètres de domaine de courbe elliptique et les algorithmes ECDH, ECDSA, et ECMQV se trouve dans l'Appendice B de [SEC1].

De plus, les méthodes d'échange de clés définies dans le présent document s'appuient sur la famille SHA2 de fonctions de hachage, définie dans [FIPS-180-3]. Les considérations appropriées sur la sécurité de ce document s'appliquent. Bien que

des faiblesses aient été découvertes dans son prédécesseur, SHA-1, aucune faiblesse n'est connue à présent dans la famille SHA2. La famille SHA2 comporte quatre variantes -- SHA-224, SHA-256, SHA-384, et SHA-521 -- nommées d'après leur longueur de résumé. En l'absence d'attaques dédiées pour exploiter la structure spécifique de la fonction de hachage, la difficulté de trouver des collisions, des pré-images, et des secondes pré-images pour la fonction de hachage est relative à la longueur du résumé. Le présent document spécifie au paragraphe 6.2.1 quelle variante de SHA2 devrait être utilisée avec quelle taille de courbe elliptique sur la base de ces lignes directrices.

Comme ECDH et ECMQV permettent des courbes elliptiques de tailles arbitraires et donc une force de sécurité arbitraire, il est important que la taille de la courbe elliptique soit choisie pour correspondre à la force de sécurité des autres éléments de la prise de contact SSH. En particulier, les tailles de clé des hôtes, les algorithmes de hachage et les algorithmes de chiffrement brut doivent être choisis de façon appropriée. Des informations concernant l'équivalence estimée des tailles de clé sont disponibles dans [NIST-800-57] ; la discussion dans la [RFC3766] est aussi pertinente. On note en particulier que quand ECDSA est utilisé comme algorithme de signature et quand ECDH est utilisé comme méthode d'échange de clés, si des courbes de différentes tailles sont utilisées, il est alors possible que des fonctions de hachage différentes de la famille SHA2 pourraient être utilisées.

Les courbes EXIGÉES et RECOMMANDÉES dans le présent document sont à présent estimées offrir la sécurité aux niveaux indiqués dans cette section et comme spécifié dans le tableau de la Section 1.

Les administrateurs de systèmes et les mises en œuvre devraient considérer avec attention les questions de sécurité quand ils utilisent des courbes autres que celles EXIGÉES ou RECOMMANDÉES dans le présent document. Toutes les courbes elliptiques ne sont pas sûres, même si elles sont sur un large champ.

Les mises en œuvre DEVRAIENT s'assurer que toutes les clés privées éphémères ou valeurs aléatoires -- y compris la valeur k utilisée dans la génération de signature ECDSA et les valeurs de clé privée éphémère dans ECDH et ECMQV -- sont générées à partir d'un générateur de nombres aléatoires ou d'un générateur de nombres pseudo aléatoires dont le germe est convenable, sont protégées contre les fuites, ne sont pas réutilisées en dehors du contexte du protocole de ce document, et sont supprimées de la mémoire quand elles ne sont plus nécessaires.

10. Paramètres de domaine de courbe elliptique désignés

Les mises en œuvre PEUVENT prendre en charge tout identifiant d'objet ASN.1 dans l'arborescence d'objets ASN.1 qui définit un ensemble de paramètres de domaine de courbe elliptique [X.680].

10.1 Courbes exigées

Chaque mise en œuvre de SSH ECC DOIT prendre en charge les courbes désignées ci-dessous. Ces courbes sont définies dans [SEC2] ; les courbes du NIST étaient à l'origine définies dans [NIST-CURVES]. Ces courbes DEVRAIENT toujours être activées sauf spécifiquement désactivées par la politique de sécurité locale.

NIST*	SEC	OID
nistp256	secp256r1	1.2.840.10045.3.1.7
nistp384	secp384r1	1.3.132.0.34
nistp521	secp521r1	1.3.132.0.35

* Pour ces trois courbes EXIGÉES, l'identifiant de paramètre de domaine de courbe elliptique est la chaîne dans la première colonne du tableau, le nom NIST de la courbe (voir le paragraphe 6.1).

10.2 Courbes recommandées

Il est RECOMMANDÉ que les mises en œuvre de SSH ECC prennent aussi en charge les courbes suivantes. Ces courbes sont définies dans [SEC2].

NIST	SEC	OID*
nistk163	sect163k1	1.3.132.0.1
nistp192	secp192r1	1.2.840.10045.3.1.1
nistp224	secp224r1	1.3.132.0.33
nistk233	sect233k1	1.3.132.0.26

nistb233	sect233r1	1.3.132.0.27
nistk283	sect283k1	1.3.132.0.16
nistk409	sect409k1	1.3.132.0.36
nistb409	sect409r1	1.3.132.0.37
nistt571	sect571k1	1.3.132.0.38

* Pour ces courbes RECOMMANDÉES, l'identifiant de paramètre de domaine de courbe elliptique est la chaîne dans la troisième colonne du tableau, la représentation ASCII de l'OID de la courbe (voir le paragraphe 6.1).

11. Considérations relatives à l'IANA

Conformément à la Section 8 de la [RFC4251] et au paragraphe 4.6 de la [RFC4250], le présent document fait les enregistrements suivants :

Dans le registre des noms d'algorithmes de clé publque : la famille de noms d'algorithme de clé publique commençant par "ecdsa-sha2-" et ne contenant pas de signe arobase (@) pour désigner les algorithmes de clé publique définis à la Section 3.

Dans le registre des noms de méthode d'échange de clés : la famille des noms de méthode d'échange de clés SSH commençant par "ecdh-sha2-" et ne contenant pas de signe arobase (@) pour désigner les méthodes d'échange de clés définies à la Section 4.

Dans le registre des noms de méthode d'échange de clés : le nom de méthode d'échange de clés SSH "ecmqv-sha2" pour désigner la méthode d'échange de clés définie à la Section 5.

Le présent document ne crée pas de nouveau registre.

12. Références

12.1 Références normatives

[X.680] Recommandation UIT-T X.680 | ISO/CEI 8824-1:2002. "Technologie de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : spécification de la notation de base".<https://www.itu.int/rec/T-REC-X.680>

[FIPS-180-3] National Institute of Standards and Technology, "Secure Hash Standard", FIPS 180-3, octobre 2008.

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997, DOI 10.17487/RFC2104.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. DOI 10.17487/RFC2119, *(MàJ par* RFC8174)

[RFC3766] H. Orman, P. Hoffman, "Détermination de la force des clés publiques utilisées pour l'échange de clés symétriques", avril 2004. (BCP0086)

[RFC4250] S. Lehtinen et C. Lonvick, éd., "Numéros alloués du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8268, RFC9142)

[RFC4251] T. Ylonen et C. Lonvick, "Architecture du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8308)

[RFC4253] C. Lonvick, "Protocole de couche Transport Secure Shell (SSH)", janvier 2006. *(P.S., MàJ par* RFC6668, 8268, 8308, 8332, 8709, 9142)

[SEC1] Standards for Efficient Cryptography Group, "Elliptic Curve Cryptography", SEC 1, mai 2009, <http://www.secg.org/download/aid-780/sec1-v2.pdf>.

[SEC2] Standards for Efficient Cryptography Group, "Recommended Elliptic Curve Domain Parameters", SEC 2,

septembre 2000, <http://www.secg.org/download/aid-386/sec2_final.pdf>.

12.2 Références pour information

[ANSI-X9.62] American National Standards Institute, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62, 1998.

[ANSI-X9.63] American National Standards Institute, "Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography", ANSI X9.63, janvier 1999.

[HMV04] Hankerson, D., Menezes, A., and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer ISBN 038795273X, 2004.

[LMQSV98] Law, L., Menezes, A., Qu, M., Solinas, J., and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement", University of Waterloo Technical Report CORR 98-05, août 1998, <http://www.cacr.math.uwaterloo.ca/techreports/1998/corr98-05.pdf>.

[NIST-800-57] National Institute of Standards and Technology, "Recommendation for Key Management - Part 1: General (Revised)", NIST Special Publication 800-57, mars 2007.

[NIST-CURVES] National Institute of Standards and Technology, "Recommended Elliptic Curves for Federal Government Use", juillet 1999.

Appendice A. Remerciements

Les auteurs remercient de leur utiles commentaires James Blaisdell, David Harrington, Alfred Hoenes, Russ Housley, Jeffrey Hutzelman, Kevin Igoe, Rob Lambert, Jan Pechanek, Tim Polk, Sean Turner, Nicolas Williams, et les membres de la liste de diffusion ietf-ssh@netbsd.org.

Adresse des auteurs

Douglas Stebila
Queensland University of Technology
Information Security Institute
Level 7, 126 Margaret St
Brisbane, Queensland 4000
Australia
mél : douglas@stebila.ca

Jon Green
Queen's University
Parallel Processing Research Laboratory
Department of Electrical and Computer Engineering
Room 614, Walter Light Hall
Kingston, Ontario K7L 3N6
Canada
mél : jonathan.green@queensu.ca