GUIDE DE SÉLECTION D'ALGORITHMES CRYPTOGRAPHIQUES

GUIDE ANSSI

ANSSI-PA-079
8/3/2021

PUBLIC VISÉ :

Développeur **Administrateur** RSSI **DSI** Utilisateur

Informations

Attention

Ce document rédigé par l'ANSSI présente un « **Guide de sélection d'algorithmes cryptographiques** ». Il est téléchargeable sur le site `www.ssi.gouv.fr`. Il constitue une production originale de l'ANSSI. Il est à ce titre placé sous le régime de la « Licence ouverte » publiée par la mission Etalab (`www.etalab.gouv.fr`). Il est par conséquent diffusable sans restriction.

Ces recommandations sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Évolutions du document :

VERSION	DATE	NATURE DES MODIFICATIONS
1.0	8/3/2021	Version initiale

Table des matières

1
Introduction

1.1 Objectif

L'objectif de ce document est d'apporter une aide à la sélection des mécanismes cryptographiques. Pour ce faire, il propose, sous forme de *recommandations*, une liste de mécanismes cryptographiques standardisés à l'état de l'art qui permettent de remplir des objectifs de sécurité classiquement assurés par des moyens cryptographiques, tels que la confidentialité, l'intégrité et l'authentification. Ces recommandations s'adressent en particulier aux développeurs qui intègrent des mécanismes cryptographiques à leurs produits et aux administrateurs qui configurent les mécanismes cryptographiques des produits qu'ils mettent en œuvre.

Ce document formule également des *notes d'implémentation* à destination des développeurs qui implémentent des mécanismes cryptographiques et des évaluateurs qui analysent la robustesse de ces implémentations. En particulier, ce document émet des mises en garde contre des erreurs d'implémentation relevées dans des produits et dans la littérature qui dégradent fortement la sécurité des mécanismes cryptographiques.

La conception et l'implémentation de mécanismes cryptographiques sont des processus exigeant un niveau d'expertise technique élevé. Par conséquent, ce document prend le parti de recommander des mécanismes standards. La moindre variation par rapport à leur spécification est susceptible de compromettre entièrement la sécurité du mécanisme final. Par ailleurs, les notes d'implémentation que ce document contient ne sauraient se substituer au suivi de l'état de l'art. Un développeur non spécialisé est encouragé à se tourner vers des bibliothèques cryptographiques éprouvées, en s'assurant de bien suivre leur guide d'usage.

Ce document présente des mécanismes cryptographiques considérés comme fournissant des garanties de sécurité acceptables à ce jour. Ces mécanismes peuvent être séparés en deux catégories, selon leur niveau de robustesse estimé. Il s'agit du niveau de confiance attribué à leur capacité à résister aux stratégies d'attaque constitutives de l'état de l'art de la cryptanalyse.

- **Mécanismes recommandés.** Ces mécanismes reflètent l'état de l'art en matière de conception cryptographique, et offrent un niveau de sécurité de l'ordre de 128 bits (cf Annexe A pour une définition informelle de cette notion). La sécurité de ces mécanismes repose sur des arguments de sécurité solides, attestant de leur résistance contre les attaques connues, et ce même en tenant compte de l'évolution prévisible de la puissance de calcul disponible. Les menaces résiduelles contre la sécurité de ces mécanismes consistent en l'apparation d'avancées radicales dans leur cryptanalyse, que ce soit la découverte de nouvelles techniques de cryptanalyse ou d'innovations fondamentales en termes de moyens de calcul, comme par exemple la réalisation d'un ordinateur quantique universel de dimension suffisante.

- **Mécanismes obsolescents.** Ces mécanismes figurent dans ce guide en raison de leur très large déploiement. De ce fait les développeurs et administrateurs se trouvent souvent confrontés à ces mécanismes, et il est utile de savoir comment se positionner vis-à-vis de leur emploi. Ces mécanismes offrent un niveau de sécurité d'au moins 100 bits, et sont considérés comme fournissant des garanties acceptables à court terme. Cependant ils ne reflètent plus l'état de l'art en matière de cryptographie : ils souffrent de limitations par rapport aux algorithmes recommandés et leur emploi devrait être abandonné dans la mesure du possible au profit de mécanismes recommandés.

La liste des mécanismes présentés dans ce guide sera mise à jour pour refléter l'évolution de l'état de l'art. Cela pourra conduire à retirer des mécanismes obsolescents de la liste des algorithmes présentés s'ils sont considérés non plus comme obsolescents mais comme obsolètes, ou à déclasser des mécanismes recommandés en mécanismes obsolescents, voire à les retirer complètement de la liste des mécanismes présentés dans le cas où de nouvelles attaques menaceraient de manière pratique leur sécurité.

Un exemple d'avancée en matière de technique de cryptanalyse susceptible d'affecter la sélection des mécanismes cryptographiques est l'implémentation d'un ordinateur quantique universel de taille suffisante pour permettre la résolution de problème cryptographique de manière substantiellement plus rapide que sur un ordinateur classique. Pour les usages requérant une protection des données au-delà de 2030, il est recommandé d'adapter d'ores et déjà la sélection des mécanismes cryptographiques. On renvoie à l'annexe A.3 pour des précisions sur ce sujet.

Ce document propose une liste intentionnellement limitée de mécanismes cryptographiques pour simplifier le processus de sélection de mécanismes cryptographiques à l'état de l'art par les utilisateurs non avertis de ce guide. Cette liste n'est cependant ni exhaustive, ni limitative : de nombreux mécanismes à l'état de l'art ne figurent pas dans ce guide et sont tout à fait acceptables pour remplir les objectifs de sécurité attendus.

1.2 Classification des mécanismes cryptographiques

1.2.1 Primitives et constructions

On peut classer les mécanismes cryptographiques selon une hiérarchie de niveaux allant de simple à composé. De nombreux mécanismes cryptographiques sont construits à partir de mécanismes élémentaires appelés *primitives cryptographiques*, afin d'atteindre des objectifs de sécurité de plus haut niveau. De tels mécanismes sont appelés *constructions cryptographiques*. La construction de mécanismes cryptographiques à partir d'une ou plusieurs primitives cryptographiques reconnues permet de fonder la confiance en ces mécanismes sur les résultats d'analyse disponibles pour ces primitives : la sécurité de la construction cryptographique peut généralement être reliée à celle de la ou des primitive(s) cryptographique(s) sous-jacente(s), parfois avec des bornes quantifiables sur la perte d'assurance entre les deux.

Ainsi, le mode GCM [1] permet de construire un schéma de chiffrement authentifié symétrique permettant d'assurer la confidentialité et l'intégrité des données en se fondant sur un algorithme

1. Galois Counter Mode

de chiffrement par bloc comme l'AES. Citons également à titre d'exemple le schéma de signature EC-DSA qui permet d'assurer l'intégrité, l'authentification d'origine de données et leur non-répudiation. Ce schéma est une construction basée sur le problème du logarithme discret sur une courbe elliptique (comme NIST P-256) et une fonction de hachage (comme SHA-256).

Un principe conducteur suivi dans ce document est que les constructions cryptographiques recommandées doivent reposer sur des primitives recommandées.

Note 1.2.a : Combinaison de primitives et de constructions

Pour qu'une construction soit considérée comme recommandée, elle doit utiliser exclusivement des primitives recommandées, sauf avis contraire *explicite*. La combinaison d'une primitive obsolescente avec une construction recommandée est considérée comme un mécanisme obsolescent. Les notes de mise en garde relatives à une primitive ou à un type de primitive s'appliquent également lorsqu'une primitive est mise en œuvre par une construction, et ce même si ces mises en garde ne sont pas systématiquement rappelées dans ce document.

1.2.2 Mécanismes symétriques et asymétriques

Les mécanismes cryptographiques sont généralement paramétrés par des *clés*. La sécurité d'un mécanisme cryptographique repose sur la confidentialité et/ou l'intégrité de ces clés. Un mécanisme cryptographique est dit *symétrique* lorsque la clé utilisée par chaque entité est connue ou peut être calculée par l'autre (ou les autres) entité(s). Les clés utilisées par toutes les entités sont généralement égales. Par exemple, dans un mécanisme symétrique de chiffrement, la clé de chiffrement est la même que la clé de déchiffrement. Une telle clé est qualifiée de *clé secrète* et sa confidentialité doit être assurée. Un mécanisme cryptographique est dit *asymétrique* lorsque les clés utilisées par les entités ne se déduisent pas toutes les unes des autres. Usuellement, on regroupe les clés en *bi-clé*, constitués d'une *clé privée*, qui est attribuée à une entité précise et dont la confidentialité doit être assurée, et d'une *clé publique*, qui peut être diffusée mais dont l'intégrité doit être assurée. Par exemple, dans un mécanisme de signature électronique, une entité possède une clé privée de signature, et la clé de vérification peut-être publiée.

Les fonctions de hachage, mécanismes cryptographiques sans clé, sont considérés comme des mécanismes cryptographiques symétriques en raison des principes de conception sur lesquels elles reposent, qu'elles partagent avec les autres primitives cryptographiques symétriques.

1.3 Structure de ce guide

Ce guide débute par un panorama général des principaux types de mécanismes cryptographiques, accompagné d'un rappel de leur cas d'emploi et de mises en garde pour leur bonne mise en œuvre.

Le corps du document dégage ensuite, pour chaque type de mécanisme cryptographique, un ou des mécanismes recommandés. Les primitives cryptographiques sont présentées par types de mécanisme. Les constructions cryptographiques sont regroupées selon le type d'objectif de sécurité auquel elles répondent. Pour chaque mécanisme, les informations suivantes sont fournies :

- une table récapitulant les mécanismes recommandés, marqués R, et obsolescents, marqués O, pour le type de mécanisme considéré. Si des restrictions s'appliquent aux valeurs de paramètres pour lesquelles ces mécanismes sont considérés comme recommandés ou obsolescents, ces restrictions apparaissent dans la colonne « taille de paramètres » de la table. Dans le cas où le statut donné ne s'applique pas directement au mécanisme, mais uniquement à des constructions reposant sur ce mécanisme, un astérisque est ajouté au statut. Ainsi un mécanisme marqué R* n'est pas recommandé en tant que tel, mais peut être utilisé par une construction pour obtenir un mécanisme recommandé. Par exemple, le mode CTR n'est pas recommandé en tant que tel mais peut être utilisé dans une construction de type Encrypt-then-MAC pour obtenir un chiffrement authentifié recommandé ;
- des notes d'implémentation visant à prévenir des implémentations défecteuses du mécanisme. Les notes générales s'appliquent à tous les mécanismes d'un type donné. Les notes spécifiques s'appliquent à un ensemble plus restreint de mécanismes ou à un seul mécanisme.

1.4 Autres documents de référence

Le présent guide vise à fournir des recommandations sur le choix et le dimensionnement des mécanismes cryptographiques. Tous les exemples de mécanismes cryptographiques recommandés qu'il contient sont compatibles avec les règles et recommandations exprimées dans le guide des mécanismes cryptographiques [ANSSI-MecCrypto]. Les recommandations figurant dans le présent guide sont largement inspirées de celles apparaissant dans le document *SOG-IS Crypto Evalulation Scheme - Agreed Cryptographic Mechanisms* du groupe de travail SOG-IS Crypto WG du SOG-IS [SCES-ACM]. Le document cité vise cependant un objectif différent : établir une liste de mécanismes cryptographiques mutuellement reconnus en vue d'établir un schéma harmonisé d'évaluation de la cryptographie dans le cadre d'évaluations selon les Critères Communs couvertes par les accords du SOG-IS. Le présent guide est assez proche, dans son esprit, du rapport [ENISA-Algo] publié par l'ENISA en 2014 et réalisant un état de l'art visant à guider le choix et le paramétrage de mécanismes cryptographiques.

2
Vade-mecum de cryptographie

2.1 Emploi de la cryptographie

À l'origine, la cryptographie est la science du secret : elle sert à transmettre des messages confidentiels. Son champ d'application dépasse depuis bien longtemps cette fonction. Des mécanismes cryptographiques permettent d'assurer que des messages ne sont pas modifiés pendant leur transit, à authentifier des utilisateurs, à permettre la vérification de mots de passe sans stocker les mots de passe eux-mêmes, etc. Ils ne requièrent plus comme il y a quelques décennies le partage de secrets avec ses correspondants, mais peuvent s'appuyer sur des méthodes plus souples dites à clé publique. Les mécanismes cryptographiques restent des outils assurant des fonctions précises, et si l'outil n'est pas adapté à la fonction recherchée, celle-ci n'est sans doute pas remplie. Ainsi, la première question à se poser avant d'envisager l'emploi d'un mécanisme cryptographique est celle du but recherché. Cherche-t-on à assurer la confidentialité d'une donnée ? Son intégrité ? Désire-t-on contrôler l'accès à une donnée ou un service ? A-t-on un objectif en dehors du champ des fonctions offertes par la cryptographie, comme la disponibilité d'une donnée ou d'une ressource ?

Dans cette section, on donne une brève description des mécanismes les plus courants et de leurs fonctions.

2.1.1 Mécanismes symétriques

Chiffrement non authentifié (à ne pas utiliser seul). Un mécanisme de chiffrement symétrique permet, à l'aide d'une clé secrète K, de transformer un message clair M en un message chiffré C. L'accès à C (par exemple sur un canal de communication public) sans connaissance de K n'apporte pas d'information sur M. La même clé K permet de déchiffrer C pour retrouver M. Les bons procédés de chiffrement symétrique sont probabilistes, c'est-à-dire qu'ils utilisent en plus de la clé K une valeur aléatoire, ou bien à état persistant, ce qui permet de garantir à chaque chiffrement l'utilisation d'une valeur auxiliaire n'ayant jamais été utilisée précédemment conjointement avec la clé K. Dans les deux cas, cela permet d'introduire de la variabilité dans le chiffré C, de telle sorte qu'un même message M ne soit pas chiffré toujours en le même message C. Ainsi, il n'est pas possible, par exemple, de comparer des chiffrés entre eux pour déterminer si les messages clairs correspondants sont égaux.

Sauf impossibilité, il faut toujours coupler le chiffrement avec un mécanisme d'authentification de message. En effet, l'emploi d'un mécanisme de chiffrement seul n'empêche pas la modification des messages chiffrés, avec deux conséquences potentielles qui peuvent constituer des problèmes de sécurité : création d'une fuite d'information par l'observation du comportement de l'utilisateur légitime lorsqu'il traite la donnée chiffrée modifiée (en particulier le traitement d'erreur) et altération du sens de la donnée claire. Il ne faut pas penser que le chiffrement empêche un adversaire de

modifier de façon contrôlée une donnée claire : certains mécanismes de chiffrement sûrs (comme le mode CTR) le permettent. L'authentification peut être effectuée par un traitement distinct du chiffrement, auquel cas une clé distincte de la clé de chiffrement est nécessaire, ou par un mécanisme combinant le chiffrement et l'authentification de message. Ces deux fonctions sont décrites ci-dessous.

Authentification de message. Un mécanisme d'authentification de message (MAC) permet, à l'aide d'une clé secrète K, de produire pour tout message M un motif d'authentification μ (mu) de M (aussi appelé motif d'intégrité de M). Ce motif ne peut pas être calculé sans la connaissance de K. A l'inverse, connaissant K, et ayant reçu un message M' et un motif d'authentification μ', il est possible de recalculer le motif d'authentification de M' afin de vérifier qu'il est bien égal à μ'. Cela fournit l'assurance que la paire (M', μ') a bien été produite avec la clé K et donc que l'intégrité du message a été préservée. Il est à noter qu'un mécanisme d'authentification de message ne protège pas contre le rejeu de messages authentifiés. Cette protection nécessite un mécanisme supplémentaire.

Chiffrement authentifié. Combine le chiffrement et l'authentification de message en un seul mécanisme, avec une seule clé. Certains mécanismes de chiffrement authentifié permettent de plus d'associer au message chiffré M un message clair M' qui ne sera pas chiffré. L'authentification porte alors sur l'association du chiffré correspondant à M et de M'. Une telle combinaison peut être utile pour lier de façon authentique à une donnée chiffrée une donnée claire nécessaire à son traitement.

Dérivation de clé. Un mécanisme de dérivation de clé permet de dériver à partir d'une clé K et d'un identifiant non secret (entier, chaîne de caractères), une clé K'. La connaissance de K permet de dériver K', mais la connaissance de K' ne permet pas de remonter à K. Ce mécanisme permet de dériver à partir d'un secret initial unique plusieurs clés pour des usages différents. Comme nous allons le voir au paragraphe suivant, un mécanisme de dérivation de clé n'est pas suffisant pour la dérivation de clés à partir d'un mot de passe.

Hachage de mot de passe. Un mécanisme de hachage de mot de passe permet de calculer une valeur de vérification V à partir d'un mot de passe M et d'une valeur variable non secrète appelée sel. La valeur V ne permet pas d'obtenir d'information sur M autrement qu'en essayant des valeurs candidates pour M jusqu'à obtenir une coïncidence avec V. L'effort de calcul pour réaliser un hachage de mot de passe est généralement réglable, et ajusté de façon à ralentir l'essai de valeurs candidates pour M. Un utilisateur légitime en possession du mot de passe M peut réaliser l'opération pour un effort de calcul modéré car il ne réalise l'opération qu'une fois, mais l'effort de calcul pour un adversaire testant des valeurs candidates pour M sera plus élevé. La valeur V peut être stockée et servir de valeur de référence pour une authentification à base de mot de passe, ou servir à dériver des clés. Associée à un mécanisme de dérivation de clé, elle peut remplir ces deux fonctions simultanément.

2.1.2 Mécanismes asymétriques

Un mécanisme asymétrique utilise une paire de clés publique et privée (Pu, Pr) ou biclé. L'usage normal de ces mécanismes est de publier la clé Pu, alors que la clé privée Pr n'est connue que d'un utilisateur bien identifié. Si les clés publiques ne nécessitent pas de protection en confidentialité,

elles doivent généralement être distribuées de façon *authentique*, ce qui est le rôle des Infrastructures de Gestion de Clés (IGC) [2], dont la description sort du cadre de ce guide.

Chiffrement asymétrique. L'opération publique de chiffrement transforme à l'aide de la clé publique Pu un message clair M en un message chiffré C. L'opération privée de déchiffrement permet de recalculer M à partir de C et de la clé privée Pr. Le chiffrement asymétrique permet donc à toute personne ayant accès à la clé publique de chiffrer des messages à l'intention du détenteur de la clé privée.

Signature. L'opération privée de signature produit à partir d'un message M et de la clé privée Pr une signature σ (sigma) de ce message. L'opération publique de vérification permet de déterminer si σ est bien une signature valide de M produite en utilisant la clé privée correspondante à une clé publique Pu. Il n'est pas possible de produire une signature valide pour une clé publique Pu sans connaître la clé privée correspondante Pr. La signature électronique permet donc au détenteur d'une clé privée de générer des signatures vérifiables par toute personne ayant accès à sa clé publique. Comme dans le cas de l'authentification de message symétrique, ce mécanisme ne protège pas contre le rejeu de messages signés. Une signature peut être conservée avec le message signé pour prouver ultérieurement à un tiers son authenticité. De plus la signature est *opposable* au signataire, dans la mesure où il est le seul détenteur de la clé privée permettant de la produire : on parle de *non-répudiation*. Cette propriété n'est pas possible avec un mécanisme symétrique d'authentification de message, car il n'est pas possible dans ce cas de séparer la capacité de vérification du motif d'authentification de la capacité de produire de tels motifs.

Échange de clé (ou établissement de clé). L'échange de clé permet à deux correspondants dialoguant sur un canal public d'aboutir à un secret commun. Ce secret commun est en général utilisé pour générer des clés utilisées dans des communications ultérieures. Pour assurer la sécurité de ces protocoles contre les attaques actives, attaques de type *Man-in-the-middle*, où un adversaire s'insère dans la communication et manipule les messages échangés lors du protocole d'échange de clé, ces messages doivent être authentifiés. Ceci fournit de plus à chaque participant une garantie sur l'identité de son correspondant (dans certaines variantes, comme dans la mise en œuvre la plus courante de `HTTPS`, seul l'un des deux participants s'assure de l'identité de son correspondant). Habituellement, les participants authentifiés possèdent chacun une clé privée et l'authentification est rendue possible par la connaissance par les participants de la clé publique de leur correspondant. Les protocoles d'échanges de clé sont mis en œuvre dans différents protocoles réseau visant à établir un canal sécurisé (confidentiel et intègre), comme TLS ou IPSEC.

2.2 Mises en garde

Les concepteurs de produits de sécurité disposent aujourd'hui d'un accès à une vaste "boîte à outils" cryptographique, utile, et souvent indispensable, à leur travail. Beaucoup de mécanismes cryptographiques de qualité sont standardisés et disponibles dans des bibliothèques *open-source*. Toutefois l'utilisation de mécanismes cryptographiques recèle des pièges. Il n'est pas nécessaire d'être un expert du domaine pour faire un emploi simple de la cryptographie, cependant il est nécessaire de garder certaines idées à l'esprit pour obtenir la sécurité recherchée. Pour des usages

2. ou Public Key Infrastructures (PKI)

qui correspondent à des scénarios courants et bien compris, cette mise en œuvre est tout à fait envisageable par des non-spécialistes rigoureux et faisant preuve de bon sens.

2.2.1 Utiliser des mécanismes complets, et non les primitives sous-jacentes

Les mécanismes que nous avons listés dans la section précédente sont construits à partir de primitives de base qui sont souvent accessibles isolément dans les bibliothèques cryptographiques. Ces composants élémentaires ne doivent pas être utilisés tels quels. Voici une liste non exhaustive de telles briques.

Algorithmes de chiffrement par bloc. De nombreux mécanismes de chiffrement ou d'authentification de message sont obtenus par combinaison d'un algorithme de chiffrement par bloc, le plus souvent de taille $n = 128$ bits, et d'un mode opératoire. L'algorithme de chiffrement par bloc transforme un bloc clair de n bits en un bloc chiffré de n bits et inversement, tandis que le mode opératoire utilise l'algorithme de chiffrement par bloc pour traiter un message de longueur arbitraire (et prend en particulier en charge les messages de longueur non multiple de n). L'algorithme de chiffrement par bloc ne doit pas être employé seul pour quelque objectif que ce soit. En particulier, le chiffrement d'un message par simple découpage en parties de n bits et application de l'algorithme de chiffrement par bloc sur chaque partie est un mode opératoire (appelé mode ECB, pour *Electronic Codebook*) qui n'est absolument pas sûr et doit être évité.

Hachage. Le hachage cryptographique est une transformation sans clé. Celle-ci produit à partir d'une donnée de taille quelconque un haché (*digest*) de taille fixe. La transformation est conçue pour garantir l'impossibilité de trouver deux données distinctes ayant le même haché. Le hachage est un composant de nombreux mécanismes cryptographiques, comme certaines méthodes d'authentification de message (par exemple HMAC), ou les signatures numériques. La transmission d'un message accompagné de son haché peut permettre la détection d'erreurs de traitement ou de transmission par recalcul du haché de la donnée reçue et comparaison avec le haché reçu. Cependant, un tel usage ne permet pas de se protéger d'un adversaire modifiant intentionnellement la donnée. En effet, rien n'empêche l'adversaire de calculer et de transmettre le haché de la donnée modifiée par ses soins. Pour garantir l'intégrité cryptographique d'une donnée, un mécanisme à clé, de type MAC ou une signature asymétrique, doit être mis en œuvre.

Primitive RSA (« Textbook RSA ») La primitive RSA est la paire d'opérations mathématiques inverses l'une de l'autre transformant des entiers de $[0, N-1]$, où N est un grand entier résultat du produit de deux nombres premiers. Elle est utilisée dans les mécanismes cryptographiques fondés sur RSA comme RSA-OAEP pour le chiffrement ou RSA-PSS pour la signature. La primitive RSA ne doit pas être employée seule, bien qu'elle soit exposée dans certaines bibliothèques cryptographiques comme OpenSSL.

2.2.2 Mécanismes cryptographiques : une sécurité sous conditions

Les mécanismes cryptographiques remplissent leur fonction lorsque des hypothèses bien précises les concernant sont remplies sur certaines valeurs qu'ils manipulent.

Clés secrètes et biclés asymétriques. Les clés mises en œuvre par les mécanismes cryptographiques doivent être construites à partir de valeurs aléatoires qui doivent impérativement provenir

d'un générateur aléatoire de qualité, en général spécialement prévu pour cet usage [3]. Pour les mécanismes symétriques, ceci permet d'assurer leur résistance contre la *recherche exhaustive*, attaque générique consistant pour un adversaire à tester toutes les clés possibles. Pour les mécanismes asymétriques, ceci permet également d'assurer leur résistance contre la meilleure attaque générique, mais aussi de se prémunir contre les attaques reposant sur une connaissance partielle de la clé. En effet, dans le cas des mécanismes asymétriques, la recherche exhaustive n'est jamais la meilleure attaque, et une connaissance partielle de la clé peut en outre avoir des conséquences catastrophiques pour la sécurité.

Valeurs aléatoires. La mise en œuvre de mécanismes cryptographiques peut nécessiter l'utilisation de valeurs générées aléatoirement. Ces valeurs ne sont pas nécessairement destinées à rester secrètes, mais elles doivent avoir la distribution de probabilité attendue, et elles doivent aussi être tirées avec un bon générateur aléatoire ; il s'agit par exemple des vecteurs d'initialisation (*Initialization Vectors* ou IV) nécessaires à certains mécanismes de chiffrement symétrique probabilistes. De telles valeurs ne doivent jamais être utilisées plusieurs fois sous peine d'effondrement potentiel de la sécurité du mécanisme qui les emploie (les longueurs des IV employés en cryptographie – normalement au moins 128 bits – garantissent que la probabilité de répétition « par hasard » est négligeable).

Valeurs uniques, dites *nonces*. La mise en œuvre de mécanismes cryptographiques peut nécessiter l'utilisation de valeurs non aléatoires, mais avec une garantie de non-réutilisation. La violation de cette propriété d'unicité peut avoir des conséquences dramatiques sur la sécurité du mécanisme qui les emploie, comme dans le cas des IV des algorithmes de chiffrement par flot. Dans certains contextes, un compteur incrémenté à chaque utilisation peut être utilisé comme nonce s'il est de taille suffisante pour éviter le dépassement de sa capacité [4].

Génération d'aléa. Beaucoup d'environnements fournissent des générateurs aléatoires *de qualité cryptographique*. Encore faut-il les employer, ce qui peut par exemple nécessiter de s'assurer que les bibliothèques que l'on emploie ont été configurées correctement à la compilation.

2.2.3 Une clé, un usage

Bien que cela soit rarement explicité, lorsqu'un mécanisme requiert l'emploi d'une clé secrète, cela signifie en particulier que cette clé ne doit pas être utilisée dans un autre mécanisme cryptographique. Réutiliser une même clé dans plusieurs mécanismes différents peut s'avérer désastreux. Les accidents surviennent en particulier lorsque les différents mécanismes ont des points communs (par exemple, l'emploi de la même clé AES dans le chiffrement AES-CBC et dans l'authentification de message AES-CBC-MAC). De même, en cryptographie asymétrique, une même clé ne doit pas être employée pour plusieurs usages, par exemple du chiffrement et de la signature. C'est pour cela que les certificats de clés publiques comportent des champs permettant d'indiquer l'usage prévu des clés. Même dans les cas – fréquents – où la réutilisation d'une clé ne conduit pas à un problème avéré, elle entraîne l'impossibilité de garantir les propriétés normalement assurées par les mécanismes pris individuellement. Pour ce qui concerne la cryptographie symétrique, l'emploi d'un mécanisme de dérivation de clé permet de construire à partir d'un secret unique des clés pour

3. Le contre-exemple absolu étant l'utilisation de `rand` de la bibliothèque standard du langage C.
4. Usuellement, la garantie d'unicité s'entend relativement à une clé : le nonce n'est pas global, mais propre à la clé courante du mécanisme cryptographique. La réutilisation d'une valeur donnée avec des clés différentes n'est pas problématique.

chaque fonction à assurer. En cryptographie asymétrique, des clés indépendantes sont généralement générées pour des usages distincts.

2.2.4 Cycle de vie des clés

Les clés ont un cycle de vie dont la bonne conception a un rôle crucial pour la sécurité des fonctions qui les emploient. En premier lieu, l'effacement de clés éphémères après usage doit garantir leur destruction. De façon plus générale, il doit être possible de décrire le cycle de vie complet des clés d'un système, permettant de comprendre comment et où elles sont générées, et à tout instant jusqu'à leur effacement, où elles sont stockées, par quels mécanismes elles sont protégées (autres secrets, protections matérielles ou logicielles), quels individus ou quelles entités peuvent y avoir accès, etc. Les éventuels mécanismes de séquestre de clé, c'est-à-dire de conservation par une autorité tierce de clés cryptographiques de chiffrement d'utilisateur à des fins de recouvrement, doivent faire l'objet d'une attention toute particulière dans cette description car leur raison d'être est de permettre, sous conditions, de contourner les protections mises en place par des mécanismes cryptographiques. Ces problématiques peuvent êtres simples pour des clés éphémères ayant une existence purement locale, et s'avérer beaucoup plus complexes pour des secrets à vie longue et présents de façon distribuée, ce qui plaide pour une limitation et une ségrégation des rôles des secrets dans un système.

2.2.5 Utiliser des bibliothèques éprouvées

L'implémentation logicielle de mécanismes cryptographiques est une tâche délicate qui ne peut pas être effectuée correctement par des non-spécialistes. Il est en effet non seulement nécessaire de s'assurer que les opérations réalisées sont correctes en toutes circonstances, y compris pour ce qui concerne le traitement des erreurs, mais également de prendre en compte les fuites d'information qui peuvent survenir via des canaux inattendus [5]. C'est pourquoi il est impératif de n'employer que des bibliothèques éprouvées bénéficiant d'un suivi de leur sécurité pour tout appel à des mécanismes cryptographiques. C'est également vrai pour la génération de clés symétriques ou asymétriques. La génération de clés asymétriques fait en particulier appel à des méthodes mathématiques non triviales, ce qui est une raison supplémentaire de ne pas la réimplémenter soi-même.

2.2.6 La créativité n'est pas recommandée !

La conception d'algorithmes cryptographiques est une tâche très spécialisée, et les contextes dans lesquels des algorithmes originaux sont nécessaires sont extrêmement peu nombreux. La sécurité de tels algorithmes ne saurait uniquement reposer sur le secret de leurs spécifications. L'histoire montre que la dissémination de produits mettant en œuvre un mécanisme cryptographique peut conduire à la publication de ses spécifications. Il est donc fondamental que la sécurité des mécanismes cryptographiques soit assurée face à des attaquants disposant de leurs spécifications. Cette exigence est connue sous le nom de principe de Kerckhoffs.

Les algorithmes standards sont publics et leur sécurité a pu être étudiée de manière ouverte par

5. On sait depuis longtemps que le temps de réalisation d'un calcul peut révéler de l'information sur les secrets manipulés. Mais il a été montré ces dernières années que la complexité des architectures des processeurs modernes pouvait créer de nombreux autres chemins de fuite d'information entre processus, via des interactions malheureuses entre mécanismes comme la prédiction de branchement et les caches de données par exemple.

la communauté académique. On dispose ainsi d'un certain recul sur leur robustesse. Le niveau de confiance qui peut être attribué à un algorithme non standard est a priori moindre, car il n'a pu bénéficier d'un examen aussi large.

De plus, l'implémentation d'algorithmes non standards pose les problèmes évoqués au paragraphe 2.2.5. Cette pratique doit donc être évitée. L'invention de protocoles interactifs originaux comportant des objectifs de sécurité est également fortement déconseillée à des non-spécialistes. En effet, l'analyse des comportements inattendus dans un contexte interactif, en présence d'adversaires éventuels susceptibles de communiquer de façon anormale avec les agents honnêtes, nécessite elle aussi des compétences spécifiques. En cas de nécessité de propriétés de sécurité nouvelles, qui ne peuvent être offertes par les mécanismes cryptographiques disponibles dans les bibliothèques cryptographiques établies, il est conseillé de se rapprocher de spécialistes de la cryptographie en mesure de mettre au point les méthodes nécessaires, ou capables de les rapprocher de techniques existantes.

3
Primitives cryptographiques symétriques

En cryptographie symétrique, la sécurité est fondée sur la confidentialité d'une clé connue par les utilisateurs légitimes. La présente section présente les primitives cryptographiques recommandées et donne quelques exemples de primitives cryptographiques obsolescentes.

3.1 Algorithmes de chiffrement par bloc

Un algorithme de chiffrement par bloc est un algorithme de chiffrement élémentaire permettant de traiter des données de tailles fixes, appelées *blocs* à l'aide d'un paramètre, appelé clé. Lorsque la clé considérée est fixe, l'opération de chiffrement des blocs est inversible et l'opération inverse est appelée opération de déchiffrement. On note n et k respectivement les tailles en bits des blocs et des clés de l'algorithme. Le comportement « en boîte noire » d'un algorithme de chiffrement par bloc à l'état de l'art et de la fonction de déchiffrement associée, utilisés avec une clé tirée aléatoirement, est indiscernable de celui d'une fonction inversible tirée aléatoirement et de son inverse. En particulier, des paires (clair, chiffré) n'apportent pas d'information exploitable sur la clé mise en œuvre, ni sur l'association établie par la fonction de chiffrement entre les autres clairs et les autres chiffrés.

R1

Algorithmes de chiffrement par bloc

AES [FIPS197, ISO18033-3] est recommandé pour ses trois tailles de clés (128, 192 et 256 bits).

Primitive	Taille de clé	Taille de bloc	R/O	Notes
AES [FIPS197, ISO18033-3]	k = 128 bits	n = 128 bits	R	
	k = 192 bits		R	
	k = 256 bits		R	
Triple-DES [SP800-67, ISO18033-3]	k = 112 bits	n = 64 bits	O	3.1.a, 3.1.b
	k = 168 bits		O	3.1.b

Notes d'implémentation.

Note 3.1.a : Triple-DES 2 clés

Dans le cas du Triple-DES 2 clés, le nombre de blocs traités par une même clé ne doit pas dépasser 2^{20} blocs, soit environ 8 millions d'octets.

Note 3.1.b : Petite taille de bloc

Les algorithmes de chiffrement par bloc sont mis en œuvre par des modes opératoires permettant de satisfaire une palette variée d'objectifs de sécurité. Dans la grande majorité des cas, ces modes remplissent leur objectif de sécurité tant que le nombre de blocs de n bits traités par l'algorithme de chiffrement par bloc avec la même clé ne dépasse pas $2^{n/2-5}$. En particulier, pour le Triple-DES (à 2 ou 3 clés), une clé ne doit pas être utilisée pour chiffrer plus de 2^{27} blocs, soit 2^{30} octets. Dans un contexte d'application haut débit, par exemple le chiffrement de communications d'un réseau gigabit, cette borne n'est pas hors de portée. Il faut donc s'assurer que les clés de l'algorithme de chiffrement par bloc soient renouvelées suffisamment rapidement.

Lorsque la taille de bloc est de $n = 128$ bits, comme pour l'AES, une même clé peut traiter jusqu'à 2^{59} blocs, ce qui ne pose généralement pas de contrainte de mise en œuvre.

3.2 Algorithmes de chiffrement par flot

Un algorithme de chiffrement par flot (synchrone) permet de chiffrer un message clair de taille arbitraire en générant une suite chiffrante de même taille à partir d'une clé de taille k bits et d'un vecteur d'initialisation de taille n bits, puis en combinant message clair et suite chiffrante par une opération de XOR. Le résultat constitue le chiffré. Les suites chiffrantes produites par un algorithme de chiffrement par flot à l'état de l'art sont indiscernables de sorties d'une source d'aléa idéale, et ce même lorsque l'adversaire dispose de la liberté de choisir les vecteurs d'initialisation.

R2

Algorithmes de chiffrement par flot

L'algorithme de chiffrement par flot Chacha20 [RFC8439] est recommandé.

Primitive	R/O	Notes
ChaCha20 [RFC8439]	R	

Note d'implémentation.

Note 3.2.a : Non réutilisation de vecteur d'initialisation

Une clé donnée ne doit pas être utilisée avec un même vecteur d'initialisation pour chiffrer à l'aide d'un algorithme de chiffrement par flot deux messages différents. En effet, en cas de réutilisation, la même suite chiffrante est réutilisée, et le XOR des clairs est obtenu en prenant le XOR des chiffrés, ce qui constitue un défaut de confidentialité rédhibitoire.

3.3 Fonctions de hachage

Une fonction de hachage cryptographique est une fonction sans clé qui permet de transformer une donnée de taille arbitraire [6] en une chaîne de bits de taille h fixe, appelée haché. On exige généralement qu'une fonction de hachage cryptographique satisfasse plusieurs propriétés de sécurité. Tout d'abord il doit être difficile de déterminer deux messages distincts dont les images par la fonction de hachage soient identiques : on parle de *résistance en collision*. On attend également qu'étant donné un message, il soit difficile de déterminer un autre message ayant la même image par la fonction de hachage : on parle de *résistance en seconde préimage*. Enfin, étant donnée une valeur de haché, il doit être difficile de déterminer un message dont l'image par la fonction de hachage soit égale à cette valeur de haché : on parle de *résistance en préimage*.

R3

Fonctions de hachage

Les fonctions de hachage de la famille SHA-2 [FIPS180] et de la famille SHA-3 [FIPS202] dont la taille de sortie est supérieure ou égale à 256 bits sont recommandées.

Primitive	Taille de paramètre	R/O	Notes
SHA-2 [FIPS180, ISO10118-3]	h = 256 bits (SHA-256)	R	
	h = 384 bits (SHA-384)	R	
	h = 512 bits (SHA-512)	R	
	h = 256 bits (SHA-512/256)	R	
SHA-3 [FIPS202]	h = 256 bits	R	
	h = 384 bits	R	
	h = 512 bits	R	

6. En pratique, les fonctions de hachage standards limitent la taille des entrées traitables, mais ces limites sont suffisamment grandes pour ne pas poser de contrainte de mise en œuvre.

4
Constructions cryptographiques symétriques

Les constructions symétriques sont des mécanismes cryptographiques basés sur des primitives cryptographiques symétriques et qui permettent de couvrir un large éventail d'objectifs de sécurité.

4.1 Modes opératoires de confidentialité : modes de chiffrement

Les modes de chiffrement sont des schémas contenant une fonction de chiffrement, qui transforme un message clair en un message chiffré à l'aide d'une clé secrète, et une fonction de déchiffrement qui permet de retrouver le message clair à partir du message chiffré et de la clé. Ils sont fondés sur un algorithme de chiffrement par bloc. Il existe plusieurs notions de sécurité pour ces mécanismes. Informellement, il doit être difficile de distinguer les chiffrés de chaînes aléatoires.

R4

Supériorité du chiffrement authentifié

Les modes de chiffrement authentifié, présentés en section 4.5, apportent des garanties de confidentialité supérieures à celles des modes de chiffrement seul. Par conséquent, bien que les modes évoqués dans ce paragraphe soient recommandés à des fins de construction de mode de chiffrement authentifié, il est recommandé de ne pas les utiliser seuls.

R5

Modes de chiffrement seul

Les modes de chiffrement CTR, OFB, CBC, CFB [SP800-38A], ainsi que le mode CBC-CS [SP800-38Aadd] sont recommandés à des fins de construction de mode de chiffrement authentifié.

Schéma	R/O	Notes
CTR [SP800-38A, ISO10116]	R*	4.1.b
OFB [SP800-38A, ISO10116]	R*	4.1.b
CBC [SP800-38A, ISO10116]	R*	4.1.c
CBC-CS (CiphertextStealing) [SP800-38Aadd]	R*	
CFB [SP800-38A, ISO10116]	R*	4.1.c

Notes d'implémentation.

Pour assurer la sécurité au sens le plus fort, le mode opératoire de chiffrement doit ou bien générer un *vecteur d'initialisation* (IV) aléatoire en début de chiffrement, ou bien utiliser comme vecteur d'initialisation un argument dont la valeur ne peut être utilisée qu'une fois pour une clé donnée, c'est-à-dire un *nonce* [7]. Pour chaque mode, sa spécification précise la stratégie adoptée. Tout écart à la spécification modifie le mode et peut aboutir à un mode dont la sécurité est insuffisante, comme par exemple le mode CBC utilisé avec un IV constant ou de manière plus générale avec un IV prédictible.

Note 4.1.a : Type d'IV

Il est nécessaire de s'assurer que l'implémentation d'un mode de chiffrement satisfait les hypothèses de sa spécification concernant la nature du vecteur d'initialisation, par exemple utiliser un IV aléatoire ou utiliser pour IV un nonce.

Quelques modes opératoires de chiffrement, dits modes flot, comme OFB et CTR, opèrent en masquant le message clair par une suite chiffrante dépendant de la clé et du vecteur d'initialisation. Lors de l'utilisation de ces modes opérant à la manière d'un algorithme de chiffrement par flot, il est essentiel de s'assurer qu'un même vecteur d'initialisation n'est pas utilisé pour chiffrer plusieurs messages sous la même clé.

Note 4.1.b : Modes flot

Pour les modes CTR et OFB, on s'assurera qu'une même paire (clé, vecteur d'initialisation) n'est jamais utilisée pour chiffrer des messages distincts, cf Note 3.2.a.

Certains modes de chiffrement, comme le mode CBC, ne traitent naturellement que des messages dont la longueur est un multiple de la taille de bloc. Pour traiter des messages de longueur arbitraire avec ces modes, il est nécessaire d'opérer une transformation supplémentaire sur le dernier bloc. Une procédure répandue consiste à compléter le dernier bloc par un *padding*, usuellement une donnée présentant une redondance. La vérification du format du padding pendant le déchiffrement peut laisser fuir de l'information sur le message déchiffré, ce qui est susceptible d'être exploité pour remettre en cause la confidentialité des données protégées par ces mécanismes de chiffrement [Vau02]. Par exemple, si l'entité qui réalise le déchiffrement se comporte différemment selon que le padding du résultat du déchiffrement est correct ou non, un adversaire peut apprendre de cette différence de comportement la validité du padding. En agrégeant de telles informations apprises sur des messages dérivés d'un chiffré cible, il devient possible pour l'adversaire de réaliser le déchiffrement de ce chiffré cible. De manière plus générale, toute vérification de conformité à un format sur le résultat d'un déchiffrement est susceptible de faire fuir de l'information. Par exemple, le format HTML est exploité par les attaques décrites dans [PDM+18]. On qualifie de telles fuites d'*oracle de padding* ou encore d'oracle de format.

Note 4.1.c : Oracle de padding

On s'assurera que le déchiffrement de message chiffré ne met pas à disposition d'éventuels adversaires un oracle de padding ou de format.

7. Une telle valeur peut être générée par exemple par un compteur

De manière concrète, il s'agit de vérifier que l'implémentation ne présente pas de différence de comportement observable selon la validité d'un padding ou d'un format. Par exemple, elle ne produit pas de message d'erreur permettant de distinguer les deux cas, n'interrompt pas l'exécution d'un protocole de manière prématurée, et n'envoie pas de message vers l'extérieur selon la validité d'un padding ou d'un format. Une autre manière de procéder est de s'assurer que la probabilité pour un adversaire de soumettre à déchiffrement un chiffré non rejoué correspondant à un message respectant le padding ou format est négligeable.

En général, il est difficile d'obtenir de telles assurances pour un mécanisme de chiffrement non authentifié : l'étude doit en effet rentrer dans le détail de l'exploitation des messages déchiffrés. Une manière systématique de se prémunir contre ce type d'attaque est d'utiliser un mécanisme de chiffrement authentifié, dont la fonction de déchiffrement est implémentée de manière à vérifier l'intégrité du chiffré avant tout autre traitement, cf Section 4.5. Cette architecture est beaucoup plus robuste, car elle permet d'obtenir des assurances de sécurité fortes indépendamment du détail de traitement du déchiffré.

4.2 Modes opératoires de confidentialité : cas particulier du chiffrement de disque

Afin d'obtenir une sécurité au sens le plus fort, les modes de chiffrement génériques génèrent des chiffrés dont la taille est plus grande que la taille du clair d'au moins un bloc à cause de l'ajout d'un IV ou d'un nonce. Ils peuvent de plus ne pas fournir de moyen efficace de déchiffrer la partie du chiffré située à une position arbitraire. Ces propriétés peuvent être considérées gênantes dans le contexte du chiffrement de disque.

Des modes de chiffrement spécifiques ont été conçus afin de garantir des tailles de chiffrés égales aux tailles des clairs correspondants et de permettre un déchiffrement efficace en toute position du chiffré. Ces propriétés sont satisfaites en limitant la perte de sécurité qui en résulte grâce à l'utilisation de métadonnées additionnelles, comme les adresses des positions des chiffrés sur le médium de stockage.

R6

Modes de chiffrement de disque

Le mode de chiffrement XTS [SP800-38E] est recommandé.

Schéma	R/O	Notes
XTS [SP800-38E]	R	4.2.b, 4.2.c
CBC-ESSIV	O	4.2.d, 4.2.e, 4.2.f

Notes d'implémentation.

Note 4.2.a : Chiffrement de disque et modes flot

Les modes de chiffrement de disque sont par nature des modes déterministes, pour lesquels les valeurs jouant le rôle de vecteurs d'initialisation sont dérivées des positions où les données chiffrées sont stockées. Ceci rend l'utilisation de modes flot inappropriée, dans la mesure où les chiffrés de deux messages clairs stockés successivement à la même position font fuir la différence des messages clairs.

Note 4.2.b : Unicité Tweak

Les modes comme XTS utilisent pour chaque bloc chiffré par l'algorithme de chiffrement par bloc sous-jacent un diversifiant appelé *tweak*. Un (ensemble de) disque(s) chiffré(s) ne doit pas contenir deux blocs chiffrés avec la même paire (clé, tweak).

Note 4.2.c : Tweaks dérivés d'adresses logiques

La valeur de tweak peut être dérivée de l'adresse physique où le bloc de chiffré est stocké. Lorsque la valeur de tweak est déterminée à partir d'adresses logiques plutôt que physiques, l'unicité des tweaks n'est pas garantie a priori et une attention particulière doit être portée à la prise en compte de la note 4.2.b.

Note 4.2.d : Unicité du numéro de secteur

Pour les modes de chiffrement de disque utilisant comme diversifiant les numéros de secteurs des disques, un (ensemble de) disque(s) chiffré(s) sous une même clé ne doit pas contenir deux blocs chiffrés avec la même paire (clé, numéro de secteur).

Note 4.2.e : Numéros de secteur logique

La valeur de numéro de secteur peut être dérivée d'une adresse physique sur le medium de stockage. Pour les périphériques de stockage qui déterminent la valeur de numéro de secteur à partir d'adresses logiques plutôt que physiques, l'unicité des numéros de secteur n'est pas garantie a priori et une attention particulière doit être portée à la prise en compte de la note 4.2.d.

Note 4.2.f : Malléabilité du mode CBC

Le mode de chiffrement CBC est malléable : en modifiant un bloc d'un chiffré CBC, un adversaire peut contrôler en partie les modifications induites sur le clair obtenu par le destinataire de ce chiffré. Plus précisément, en modifiant le i-ème bloc de chiffré il peut contrôler entièrement les modifications induites sur $i+1$-ème bloc de clair. En raison de cette malléabilité, un adversaire est susceptible de pouvoir manipuler un disque chiffré à l'aide du mode CBC-ESSIV pour obtenir un déchiffré malveillant [Lel13]. Dans un contexte d'usage où ce type d'attaque doit être pris en compte, CBC-ESSIV ne doit donc pas être utilisé.

4.3 Modes d'intégrité et codes d'authentification de messages (MAC)

Un mode d'intégrité consiste en : une fonction de génération de code d'authentification de message (MAC) prenant en entrées une clé secrète K et un message M et retournant un MAC μ ; une fonction de vérification de MAC prenant en entrées K, M et μ et retournant Vrai ou Faux. Il ne doit pas être possible pour un adversaire de générer une paire message/MAC valide originale, même s'il dispose de la possibilité d'obtenir des paires message/MAC valides de la part des utilisateurs légitimes.

Un mode d'intégrité peut reposer sur divers types de primitives, comme un algorithme de chiffrement par bloc, une fonction de hachage, ou une fonction de hachage universel, cf R10.

Pour des raisons de bande passante, il est courant de tronquer la sortie d'une fonction de génération de MAC. Afin que la probabilité pour un adversaire de produire aléatoirement un MAC valide reste faible, la taille t de la sortie après troncature doit rester suffisamment grande.

R7

Troncature de MAC

Dans le cas général, il est recommandé de ne pas tronquer la sortie d'une fonction de génération de MAC recommandée à strictement moins de 96 bits.

Dans des environnements contraints, de type cartes à puce, on rencontre parfois des MAC tronqués à 64 bits.

Note 4.3.a : Troncature de MAC à 64 bits

Dans le cas spécifique où le nombre maximal de requêtes de vérification faites pour une clé donnée est au plus de 2^{20}, il est toléré de tronquer la sortie d'une fonction de génération de MAC recommandée jusqu'à 64 bits.

Cette recommandation et cette note s'appliquent dans le cas général. Cependant, pour certains schémas de MAC, il peut être nécessaire d'être plus restrictif en matière de troncature de MAC. C'est notamment le cas de GMAC, cf. 4.5.f.

R8

Modes d'intégrité basés sur un algorithme de chiffrement par bloc

Le mécanisme d'intégrité CMAC basé sur un algorithme de chiffrement par bloc recommandé est recommandé.

Schéma	R/O	Notes
CMAC [SP800-38B, ISO9797-1]	R	
CBC-MAC [ISO9797-1, Algorithme 1, Padding 2]	R	4.3.b

Note d'implémentation.

Note 4.3.b : Taille d'entrée constante

CBC-MAC n'est recommandé que dans le cas où, pour une clé donnée, tous les messages ont la même taille. Des attaques par extension de message sont trivialement possibles lorsque des messages de taille variable sont autorisés, et son utilisation est donc proscrite dans ce cas.

R9

Modes d'intégrité basés sur une fonction de hachage

Le mécanisme d'intégrité HMAC basé sur une fonction de hachage recommandée est recommandé.

Schéma	Taille de clé	R/O	Notes
HMAC [RFC2104, ISO9797-2]	$k \geq 128$	R	
	$100 \leq k \leq 128$	O	
HMAC-SHA-1 [RFC2104, ISO9797-2, FIPS180]	$k \geq 100$	O	4.3.c

Note d'implémentation.

Note 4.3.c : HMAC-SHA-1

La sécurité de HMAC ne repose pas sur la résistance à la recherche de collisions de la fonction de hachage sous-jacente. Bien que SHA-1 soit une fonction de hachage proscrite pour une utilisation générale, et n'apparaisse donc pas dans ce guide comme un mécanisme obsolescent acceptable, l'utilisation de HMAC-SHA1 est tolérée.

R10

Modes d'intégrité basés sur le hachage universel

Le hachage universel est une méthode qui consiste à sélectionner de manière aléatoire une fonction de hachage non cryptographique dans une famille de fonctions possédant de bonnes propriétés de répartition, ce qui assure pour deux messages distincts que la probabilité sur la fonction sélectionnée qu'ils possèdent la même image par cette fonction est faible. Le mécanisme d'intégrité GMAC basé sur une famille de fonctions de hachage universel est recommandé.

Schéma	R/O	Notes
GMAC [SP800-38D]	R	4.5.c 4.5.d 4.5.f

Les notes relatives à GMAC sont partagées avec GCM, et peuvent être trouvées dans la section 4.5 relative au chiffrement authentifié.

4.4 Schémas symétriques d'authentification d'entité

Ces schémas permettent à une entité de prouver son identité à un correspondant en utilisant la connaissance d'un secret. Par nature, ces schémas sont interactifs. Ils consistent généralement à mettre en oeuvre un protocole de défi-réponse reposant sur l'utilisation d'aléa et d'une primitive symétrique ou d'un mécanisme de chiffrement ou d'intégrité. Nous ne donnons pas de liste pour ce type de schéma. Il est important de garder à l'esprit que même si ces schémas peuvent reposer sur le calcul d'un motif d'intégrité, leurs objectifs de sécurité sont différents de ceux des schémas d'authentification de données. Par conséquent, une même clé ne doit pas être utilisée pour un mode d'intégrité et pour un schéma symétrique d'authentification d'entité.

Note 4.4.a : Collision de défis

Le vérifieur doit s'assurer qu'une valeur de défi ne peut être rejouée qu'avec une probabilité négligeable. Le défi peut par exemple être instancié par une valeur aléatoire de taille suffisamment grande et générée par le vérifieur.

R11

Taille pour la valeur aléatoire utilisée comme défi pour l'authentification d'entité

Une taille d'au moins 128 bits est recommandée.

Taille du défi	R/O
128	R
96	O

4.5 Chiffrement authentifié

L'objectif du chiffrement authentifié (généralement abrégé *AE* pour *authenticated encryption*) est de garantir la confidentialité et l'intégrité de données, ainsi que d'authentifier leur origine. Un schéma d'AE fournit une fonction de chiffrement qui transforme un message clair en un chiffré en utilisant une clé, ainsi qu'une fonction de déchiffrement qui permet de retrouver le clair à partir du chiffré et de la clé. Contrairement à celle d'un schéma de chiffrement seul, la fonction de déchiffrement d'un schéma d'AE ne renvoie pas de valeur de déchiffré si le chiffré ne respecte pas un certain motif de redondance permettant de vérifier l'intégrité du chiffré. Généralement, une partie du chiffré contient une valeur de vérification qui doit être testée au cours du déchiffrement.

Les schémas d'AE fournissent souvent une fonctionnalité supplémentaire, qui permet d'authentifier, en plus des données chiffrées, des données dites additionnelles qui sont transmises en clair. Un tel schéma est souvent appelé chiffrement authentifié avec données associées (AEAD, pour *AE with Associated Data*). Un schéma d'AE ou d'AEAD peut être obtenu par combinaison d'un schéma de chiffrement et d'un code d'authentification de message.

R12

Mécanisme de chiffrement authentifié

Les mécanismes Encrypt-then-MAC, GCM, CCM, EAX sont recommandés dès lors qu'ils sont instanciés avec des primitives recommandées.
Le mécanisme ChaCha20-Poly1305 est recommandé.

Schéma	R/O	Notes
Encrypt-then-MAC [BN00]	R	4.5.b
CCM [SP800-38C, ISO19772]	R	
GCM [SP800-38D, ISO19772]	R	4.5.c, 4.5.d 4.5.e 4.5.f
EAX [ISO19772]	R	
ChaCha20-Poly1305 [RFC8439]	R	4.5.c
MAC-then-Encrypt [BN00]	O	4.5.b
Encrypt-and-MAC [BN00]	O	4.5.b

Notes d'implémentation.

Les considérations sur la troncature des motifs d'intégrité, Recommandation R7 et Note 4.3.a, s'appliquent également aux schémas symétriques de chiffrement authentifié.

Note 4.5.a : Ordre de déchiffrement

Si le déchiffrement n'est pas conditionné par le succès de la vérification de l'intégrité des chiffrés, le développeur (ou l'évaluateur le cas échéant) doit s'assurer que l'implémentation ne fournit à un adversaire potentiel aucun oracle de padding ou de format, cf Note 4.1.c. Plus précisément, l'implémentation ne doit faire fuir aucune information sur le format du message obtenu après déchiffrement d'un chiffré arbitraire (telle que la validité ou non du padding). Dans le cas contraire, un adversaire pourrait être capable de décrypter un chiffré-cible en exploitant de telles fuites d'information, qui peuvent être accessibles notamment via des messages d'erreur explicites ou via des canaux auxiliaires d'information tels que le temps de calcul. Les messages déchiffrés ne doivent jamais être renvoyés à l'application qui demande le déchiffrement avant que leur intégrité n'ait été vérifiée.

Note 4.5.b : AE obtenu par composition

Les mécanismes de chiffrement authentifié obtenus par composition générique d'un mécanisme de chiffrement et d'un mécanisme d'intégrité sont recommandés ou acceptés sous condition que les clés utilisées pour le chiffrement et le calcul du MAC soient indépendantes [a], et que l'IV ou le nonce éventuellement utilisé pour le chiffrement soit couvert par le motif d'intégrité.

a. ou issues d'un mécanisme de dérivation de clé, cf Section 4.7

Note 4.5.c : Non-réutilisation de l'IV

L'IV doit être traité dans le périmètre de sécurité du schéma de chiffrement authentifié. Par exemple, il est essentiel de s'assurer qu'un adversaire ne peut jamais provoquer l'utilisation d'une même valeur d'IV pour protéger deux paires (message, données associées) différentes avec la même clé. Une réutilisation d'un IV peut remettre en cause la confidentialité de certains modes, cf Note 3.2.a. En outre dans le cas particulier de GCM, la réutilisation d'un IV remet en cause l'intégrité des chiffrés.

Note 4.5.d : Options GMAC-GCM

Le mécanisme GCM n'est recommandé que si les conditions suivantes sont remplies : la longueur de l'IV doit être de 96 bits ; la méthode de construction déterministe d'IV de [SP800-38D, Section 8.2.1] doit être employée.

Note 4.5.e : Longueur de clair GCM

La spécification de GCM impose une limite sur la taille des messages clairs qui peuvent être chiffrés lors d'un appel à la fonction de chiffrement, avec une clé et un IV donnés : celle-ci est limitée à $2^{39} - 256$ bits, soit $2^{32} - 2$ blocs étant donné que GCM repose sur une taille de bloc de 128 bits. Dans les environnements où la longueur des clairs pourrait excéder cette valeur, il est essentiel de vérifier explicitement que ce n'est jamais le cas.

Note 4.5.f : Bornes GMAC-GCM

La résistance aux contrefaçons de GMAC et GCM, c'est-à-dire à la possibilité pour un adversaire ne disposant pas de la clé de créer une paire (message, MAC) ou (message, chiffré) valide, n'étant pas optimale, la longueur de MAC doit être d'au moins 128 bits. De ce fait, aucune troncature réduisant la longueur du MAC final (par exemple à 96 bits) ne peut être appliquée à GMAC et GCM.

4.6 Chiffrement de clé

Un mécanisme de chiffrement de clé permet le stockage ou la transmission sécurisée de clés, en garantissant leur confidentialité, leur intégrité et l'authenticité de leur origine.

Un mécanisme de chiffrement de clé est un schéma de chiffrement authentifié. Le fait que les données protégées soient aléatoires permet d'utiliser un schéma de chiffrement authentifié déterministe.

R13

Mécanismes de chiffrement de clé

Les mécanismes de chiffrement de clé SIV et AES-KeyWrap sont recommandés.

Schéma	R/O	Notes
SIV [RFC5297]	R	
AES-Keywrap [SP800-38F, algorithms KW and KWP]	R	

4.7 Fonctions de dérivation de clés

Un mécanisme de dérivation de clés permet de calculer une ou plusieurs clés à partir d'un secret maître. Il repose sur une primitive symétrique, usuellement une fonction de hachage. Un tel mécanisme prend généralement trois arguments en entrée : une valeur secrète K, une valeur N potentiellement publique, qui consiste usuellement en une chaîne de caractères constante jouant le rôle de diversifiant, et une longueur n ; et renvoie n bits pouvant être divisés le cas échéant en plusieurs clés qui peuvent être considérées comme indépendantes.

De nombreuses méthodes répandues permettent d'obtenir cette fonctionnalité avec un bon niveau de sécurité. La liste suivante de mécanismes de dérivation de clés recommandés n'a par conséquent pas vocation à être exhaustive.

R14

Mécanismes de dérivation de clés

Les mécanismes de dérivation de clés HKDF, NIST SP800-56 A, B, et C, ANSI-X9.63-KDF et PBKDF2 sont recommandés.

Schéma	R/O	Notes
HKDF [RFC5869]	R	
NIST SP800-56C [SP800-56C]	R	
ANSI-X9.63-KDF [ANSI-X9-63]	R	
PBKDF2 [RFC8018]	R	4.7.a

Note d'implémentation.

Note 4.7.a : PBKDF2-PRF

PBKDF2 utilise une fonction pseudo-aléatoire, qui peut être instanciée avec un code d'authentification de message. La fonction pseudo-aléatoire utilisée doit être recommandée. Si HMAC est utilisé, il est nécessaire de faire attention à la longueur de la clé. En effet, si la clé de HMAC est plus longue que les blocs de la fonction de hachage sous-jacente, la clé est préalablement hachée, ce qui peut réduire l'entropie des clés dérivées en utilisant PBKDF2.

4.8 Mécanismes de hachage de mot de passe

Les mécanismes de hachage de mots de passe associent une valeur de vérification à un mot de passe. Les systèmes qui utilisent les mots de passe pour authentifier les utilisateurs conservent ces valeurs de vérification. Ceci leur permet de vérifier les mots de passe des utilisateurs lors des ten-

tatives d'authentification, sans avoir à conserver leur valeur. En cas de compromission des valeurs de vérification, un attaquant ne doit pas pouvoir retrouver un mot de passe fort.

R15

Mécanisme de hachage de mots de passe

Le mécanisme de hachage de mots de passe PBKDF2 est recommandé.

Schéma	R/O	Notes
PBKDF2 [RFC8018]	R	4.8.a, 4.8.b

Notes d'implémentation.

Note 4.8.a : Nombre d'itérations

Les mécanismes de hachage de mots de passe sont paramétrables de manière à pouvoir choisir la complexité d'une exécution de calcul de haché. Une telle mesure permet d'augmenter le temps de calcul d'une attaque par force brute : un usage légitime du hachage d'un mot de passe nécessite une seule exécution du mécanisme de hachage, alors qu'une attaque par force brute implique un grand nombre d'exécutions. Par conséquent, il est conseillé de choisir le nombre d'itérations de PBKDF2 le plus grand possible, tant que cela n'affecte pas un usage légitime.

Note 4.8.b : Sel

Un sel est une valeur aléatoire qui est générée au moment de l'enregistrement d'un mot de passe et qui est conservée avec la valeur de vérification du mot de passe. L'ajout d'un sel à un mécanisme de hachage de mot de passe permet de se prémunir contre des attaques impliquant une phase de précalcul. Les mécanismes de hachage de mot de passe doivent être utilisés avec un sel, et la longueur du sel doit être d'au moins 128 bits.

5
Primitives cryptographiques asymétriques

En cryptographie asymétrique, la sécurité d'une primitive repose sur l'existence de deux fonctions ou opérations mathématiques inverses l'une de l'autre et présentant une asymétrie, comme par exemple la multiplication et la factorisation. Par asymétrie, on entend que le calcul de la première est réalisable de manière efficace, alors que le calcul de la seconde nécessite de résoudre un problème difficile, et n'est donc pas réalisable en pratique si la taille des objets manipulés est suffisamment importante.

Une telle asymétrie est nécessaire pour générer des couples clé publique/clé privée pour lesquels il n'est calculatoirement pas possible de retrouver la clé privée ou des informations sur la clé privée à partir de la clé publique. Une telle asymétrie est également nécessaire pour assurer qu'un attaquant connaissant la clé publique n'est pas capable d'effectuer des opérations impliquant la clé privée, et ce même s'il dispose en plus de résultats de calculs effectués avec la clé privée dont il connaît ou a choisi les entrées.

Ce chapitre traite des primitives asymétriques classiques ainsi que des problèmes mathématiques auxquels elles sont reliées. Dans le chapitre suivant, nous verrons comment utiliser de telles primitives pour construire des schémas cryptographiques plus complexes.

5.1 Primitive RSA et factorisation de grands entiers

La sécurité de nombreux schémas cryptographiques asymétriques, et notamment du schéma de chiffrement RSA, repose sur la difficulté de résoudre le problème de la factorisation d'entiers de grande taille : étant donné un entier N, écrire N comme produit d'entiers premiers. Il n'existe, à l'heure actuelle, auncun algorithme de résolution de ce problème ayant un temps de calcul raisonnable en pratique pour traiter des entiers quelconques de grandes tailles.

Dans la primitive RSA, on considère deux grands nombres premiers p et q et leur produit $N = pq$ que l'on appele *module* RSA. On note $n = \lfloor \log_2(N) \rfloor + 1$ la taille en bit du module. La clé publique Pu est constituée du couple (e, N) où e est un entier appelé exposant public ; la clé privée Pr est constituée d'un entier d appelé exposant privé, lié à p, q et e. La permutation publique prend en entrée un entier m modulo N et effectue le calcul de l'exponentiation modulaire de m à la puissance e (exposant public) modulo N. La permutation privée, inverse de la permutation publique, prend en entrée un entier modulo N et effectue le calcul de l'exponentiation modulaire de cet entier à la puissance d.

Cette construction est utilisée dans de nombreux schémas de chiffrement ou de signature RSA,

qui seront présentés dans les sections suivantes. Dans ces schémas, on fait intervenir, en plus de la primitive, de l'aléa et un padding, pour lequel il existe d'ailleurs de nombreuses conventions. Il est à noter que, comme décrit en section 2.2, la primitive seule ne peut, en aucun cas, être considérée comme un schéma de chiffrement ou de signature valide : l'absence de padding la rend particulièrement fragile.

R16

Dimensionnement du schéma asymétrique RSA

On recommande d'utiliser des modules RSA d'au moins 3072 bits et des exposants publics e de taille strictement supérieure à 2^{16}.

Primitive	Taille des paramètres	R/O	Notes
RSA	$n \geq 3072$, $\log_2(e) > 16$	R	
	$n \geq 2048$, $\log_2(e) > 16$	O	

Notes d'implémentation. Une bonne génération de paires de clés RSA repose non seulement sur l'utilisation d'un générateur d'aléa de bonne qualité, mais aussi sur le respect de conditions additionnelles. On renvoie à la section 7.3.3 pour le traitement de la génération des biclés RSA.

5.2 Logarithme discret sur un corps fini (FF-DLOG)

La sécurité de nombreux schémas cryptographiques asymétriques repose sur la difficulté de résolution du problème du logarithme discret (DLOG) sur le groupe multiplicatif d'un corps fini.

Le problème du logarithme discret peut être décrit de la façon suivante : soit h un élément d'un groupe G ; on note ω_h l'ordre de h. La fonction d'exponentiation en base h dans G prend en entrée un entier x (entre 1 et $\omega_h - 1$) et retourne $X = h^x$. Calculer la fonction inverse revient à résoudre le problème du logarithme discret, c'est-à-dire retrouver x étant donné X.

On peut utiliser comme groupe G le groupe multiplicatif d'un corps fini. Les corps finis les plus largement utilisés sont ceux qui donnent les meilleures garanties en termes de sécurité, soit les corps premiers $\mathsf{GF}(p)$ où p est un nombre premier. Dans la suite de ce document on se restreint à leur cas. Les éléments du groupe G peuvent être assimilés aux entiers compris entre 1 et $p - 1$ inclus, et la fonction d'exponentiation est appelée exponentiation modulaire : $X = h^x \mod p$.

Le problème du logarithme discret sur un corps premier est considéré comme difficile à résoudre : il n'existe, à l'heure actuelle, aucun algorithme de résolution ayant un temps de calcul raisonnable en pratique.

Le problème du logarithme discret dans le groupe multiplicatif d'un corps $\mathsf{GF}(p)$ peut être utilisé dans de nombreux protocoles d'échange de clé ou schémas de signature ou pour effectuer du chiffrement hybride. Ces schémas seront décrits plus en détail en section 6. Selon la manière d'utiliser la primitive en question, x et X peuvent parfois représenter une clé privée et la clé publique associée, ou encore un exposant Diffie-Hellman et la valeur publique associée.

Les paramètres qui définissent un groupe multiplicatif sur un corps fini sont constitués d'un nombre premier p appelé module, un entier g appelé générateur et son ordre q.

R17

Groupes multiplicatifs d'un corps fini pour le DLOG

Les groupes MODP dont le module est de taille supérieure ou égale à 3072 bits sont recommandés.

Familles	Groupes	R/O	Notes
MODP [RFC3526]	3072-bit MODP	R	
	4096-bit MODP	R	
	6144-bit MODP	R	5.2.a
	8192-bit MODP	R	
	2048-bit MODP	O	

Notes d'implémentation.

Quelques précautions doivent être prises pour éviter les attaques de type petit sous-groupe ou sous-groupe incorrect.

Note 5.2.a : Résistance aux attaques par petit sous-groupe

Lorsqu'une exponentiation modulaire modulo p d'un entier h par un exposant secret est effectuée, on s'assurera que $1 \leq h \leq p-1$ et $h^{\frac{p-1}{2}} = 1 \mod p$. En particulier, si h est égal au générateur g du groupe, cette propriété est satisfaite par construction.

5.3 Logarithme discret sur courbes elliptiques (EC-DLOG)

On peut aussi définir le problème du logarithme discret sur le groupe des points rationnels d'une courbe elliptique (définie sur un corps fini). La loi de groupe est notée de manière additive. Dans ce cas, le problème en question est également considéré comme étant difficile à résoudre. Il est à noter que l'on recommande uniquement d'utiliser des courbes elliptiques définies sur un corps fini premier.

Le problème du logarithme discret sur courbe elliptique peut être défini de la manière suivante : soit p un nombre premier et $\mathsf{GF}(p)$ le corps à p éléments ; soit E une courbe elliptique définie sur $\mathsf{GF}(p)$ et Q un point d'ordre ω_Q de la courbe. La fonction de multiplication scalaire en base Q sur E prend en entrée un entier x (entre 1 et $\omega_Q - 1$) appelé scalaire et retourne un point $X = [x]Q$. Résoudre le problème du logarithme discret sur E, c'est parvenir à retrouver l'entier x étant donné le point X. Selon le schéma cryptographique, x et X peuvent représenter (une partie de) la clé privée et la clé publique associée, ou bien un scalaire éphémère Diffie-Hellman et le point public associé.

Les paramètres de courbes elliptiques sont constitués d'un nombre premier p, des coefficients de l'équation de la courbe, d'un point P de la courbe appelé point de base et de son ordre q.

R18

Paramètres de courbes elliptiques pour le DLOG

Les paramètres de courbes elliptiques P256r1, P384r1 et P512r1 de la famille Brainpool, les paramètres de courbes elliptiques P-256, P-384 et P-521 du NIST et les paramètres de courbes Curve25519 et Curve448 sont recommandés.

Familles de courbes	Courbes	R/O	Notes
Brainpool [RFC5639]	BrainpoolP256r1	R	5.3.a
	BrainpoolP384r1	R	
	BrainpoolP512r1	R	
NIST [FIPS186] (voir Annexe D.1.2)	NIST P-256	R	5.3.a
	NIST P-384	R	
	NIST P-521	R	
IETF [RFC7748]	Curve25519	R	5.3.b
	Curve448	R	

Notes d'implémentation.

Quelques précautions doivent être prises pour éviter les attaques de type petit sous-groupe ou sous-groupe incorrect.

Note 5.3.a : Points sur la courbe

Pour les paramètres de courbe tels que tous les points de la courbe peuvent être obtenus comme multiples du point de base P d'ordre q premier, on s'assurera que les points manipulés appartiennent bien à la courbe elliptique, c'est-à-dire qu'ils vérifient l'équation de la courbe.

Note 5.3.b : Respect des fonctions d'encodage

Pour les paramètres de courbes Curve25519 et Curve448, lorsqu'une multiplication scalaire d'un point Q par un entier secret x est réalisée, on s'assurera soit que Q est un multiple du point de base P, soit que x est généré par application de la fonction de décodage définie en [RFC7748, Section 5] et que le résultat de la multiplication scalaire n'est pas égal à la valeur nulle.

Les spécifications des schémas cryptographiques mettant en œuvre les courbes Curve25519 et Curve448 satisfont usuellement ces conditions par construction.

5.4 Autres problèmes difficiles

Les problèmes de la factorisation d'entiers de grande taille et de la résolution du logarithme discret sur corps fini et sur courbe elliptique sont aujourd'hui les plus utilisés et les plus largement étudiés dans le monde de la cryptographie. Il est important de savoir qu'il existe d'autres types

de problèmes mathématiques difficiles à résoudre qui permettent de construire des primitives asymétriques.

Pour n'en citer que quelques uns :

- les problèmes fondés sur les réseaux tels que le problème LWE, *apprentissage avec erreurs* (qui fournit des familles de primitives résistant potentiellement aux ordinateurs quantiques) ;
- la cryptographie fondée sur les codes, qui repose sur la difficulté de décoder un code correcteur en présence d'erreurs aléatoires (qui fournit des familles de primitives résistant potentiellement aux ordinateurs quantiques) ;
- la cryptographie multivariée qui repose sur la difficulté à résoudre des systèmes d'équations polynomiales faisant intervenir un grand nombre de variables (qui fournit des familles de primitives résistant potentiellement aux ordinateurs quantiques) ;
- les problèmes de recherche de chemins dans des graphes d'isogénies de courbes elliptiques.

Comme mentionné précédemment, ces différentes familles de primitives ainsi que les problématiques liées à leur implémentation ont reçu beaucoup moins d'attention de la part de la communauté cryptographique que les problèmes classiques de factorisation et de résolution du logarithme discret. Par conséquent nous ne recommandons l'utilisation indépendante d'aucune primitive asymétrique particulière reposant sur l'un de ces problèmes.

Si l'on souhaite utiliser de tels mécanismes, ce qui paraît raisonnable lorsque les informations traitées doivent être protégées au-delà de 2030, ils doivent l'être avec beaucoup de précautions et combinés avec des primitives classiques.

R19

Hybridation

Il est recommandé de ne pas utiliser à ce stade de manière indépendante un schéma basé sur ces primitives asymétriques, et lorsqu'un tel schéma est mis en œuvre, de combiner son emploi avec un schéma basé sur des primitives asymétriques éprouvées ou avec un schéma symétrique mettant en œuvre une clé pré-partagée.

Il existe aujourd'hui un réel besoin de standardiser des schémas résistants face à la menace quantique. Ainsi des candidats post-quantiques reçoivent actuellement une attention particulière de la part de la communauté scientifique et des organismes de standardisation.

6
Constructions cryptographiques asymétriques

Certains problèmes mathématiques asymétriques peuvent être utilisés pour construire des mécanismes cryptographiques asymétriques. Dans de tels mécanismes, un utilisateur dispose d'un *biclé* asymétrique, c'est-à-dire d'une clé publique Pu (qui peut être publiée), et d'une clé privée Pr (qui doit nécessairement rester confidentielle). Il est possible, pour certains mécanismes asymétriques, de faire la preuve mathématique que la sécurité du mécanisme est équivalente à la difficulté de résoudre le problème mathématique sous-jacent, c'est-à-dire qu'il est impossible, sauf à avoir résolu ce problème, d'attaquer le mécanisme, par exemple en retrouvant la clé privée, en déchiffrant un message, ou en contrefaisant une signature.

La sécurité de tout mécanisme asymétrique est limitée par celle du problème mathématique sous-jacent. Ce problème doit donc être d'un niveau de difficulté suffisant. En particulier, les conditions de taille de clé des paragraphes précédents sont également valables pour les mécanismes asymétriques.

La sécurité d'un mécanisme asymétrique est également dépendante de la protection en confidentialité et en intégrité de la clé privée et de la protection en authenticité et en intégrité de la clé publique. Les aspects propres à chaque mécanisme sont traités ci-dessous dans le paragraphe qui lui est consacré.

6.1 Chiffrement asymétrique

Un mécanisme de chiffrement asymétrique se compose de trois fonctions. La fonction de génération de clés construit un biclé asymétrique, c'est-à-dire une clé privée Pr et la clé publique correspondante Pu. La fonction de chiffrement utilise un message clair M et la clé publique Pu pour calculer un message chiffré C. La fonction de déchiffrement, à partir du chiffré C et de la clé privée Pr, retrouve le message clair M.

Note 6.1.a : Padding aléatoire

Les schémas de chiffrement asymétrique nécessitent un mécanisme d'encapsulation probabiliste. L'aléa utilisé par un tel mécanisme doit être issu d'un générateur aléatoire recommandé, tel que décrit dans la section 7.

R20

Mécanismes de chiffrement asymétrique

Les mécanismes de chiffrement asymétrique RSA-OAEP, ECIES-KEM et DLIES-KEM sont recommandés.

Primitive	Mécanisme	R/O	Notes
RSA	OAEP [PKCS1, RFC8017]	R	6.1.c
FF-DLOG	DLIES-KEM [ISO18033-2]	R	
EC-DLOG	ECIES-KEM [ISO18033-2]	R	
RSA	PKCS#1v1.5 [PKCS1, RFC8017]	O	6.1.b

Notes d'implémentation.

Note 6.1.b : Attaque avec oracle de padding PKCS#v1.5 en chiffrement

Les attaques par oracle de padding, dont le principe est décrit en section 4.1 peuvent également s'appliquer à des mécanismes de chiffrement asymétrique. De telles attaques sont susceptibles d'affecter la version 1.5 du mécanisme d'encapsulation RSA-PKCS#1 [Ble98, BFK+12]. On s'assurera que le déchiffrement de chiffré RSA-PKCS#1v1.5 ne met pas à disposition d'éventuels attaquants un oracle de padding.

Note 6.1.c : Attaque avec oracle de padding OAEP

Une implémentation incorrecte du déchiffrement OAEP, c'est-à-dire n'effectuant pas les vérifications décrites dans la fonction de déchiffrement EME-OAEP, peut également être vulnérable à certaines attaques en présence d'un oracle [Man01].

6.2 Signature électronique

Un mécanisme de signature électronique se compose de trois fonctions. La fonction de génération de clés construit un biclé asymétrique, c'est-à-dire une clé privée Pr et la clé publique correspondante Pu. La fonction de génération de signature utilise un message M et la clé privée Pr pour calculer une signature σ. La fonction de vérification de signature, à partir du message M, de la signature σ, et de la clé publique Pu, répond si la signature est valide (réponse Vrai/Faux). Les mécanismes de signature électronique utilisent généralement une fonction de hachage.

R21

Mécanismes de signature électronique

Les mécanismes de signature électronique RSA-PSS, EC-DSA, EC-KCDSA, ainsi que les autres mécanismes mentionnés dans la table ci-dessous sont recommandés.

Primitive	Mécanisme	R/O	Notes
RSA	PSS [PKCS1, RFC8017]	R	
FF-DLOG	KCDSA [ISO14888-3]	R	6.2.b
	SDSA [ISO14888-3]	R	
	DSA [FIPS186, ISO14888-3]	R	
EC-DLOG	EC-KCDSA [ISO14888-3]	R	6.2.b
	EC-DSA [FIPS186, ISO14888-3]	R	
	EC-GDSA [BSI-TR-03111]	R	
	EC-SDSA [ISO14888-3]	R	
	EC-FDSA [ISO14888-3]	R	
RSA	PKCS#1v1.5 [PKCS1, RFC8017, ISO9796-2]	O	6.2.a

Il est à noter que les schémas ci-dessus sont recommandés à condition d'être basés sur une fonction de hachage recommandée et d'utiliser des tailles de paramètres recommandées pour la primitive asymétrique sous-jacente, cf Section 5.

Notes d'implémentation.

Note 6.2.a : Vérification de format PKCS#1v1.5 en signature

Pour prévenir une attaque de type Bleichenbacher [Ble06], les vérifications de conformité de format doivent être effectuées correctement.

Note 6.2.b : Aléa DSA

Les fonctions de génération de signature du mécanisme de signature DSA et de ses variantes, basées aussi bien sur $\mathrm{GF}(p)$ que sur courbe elliptique, utilisent une valeur aléatoire. Une fuite d'information, même partielle, de cette valeur aléatoire menace la confidentialité de la clé secrète associée. Il faut donc absolument éviter une telle fuite d'information. Ceci s'applique aussi bien aux biais statistiques (dûs au générateur aléatoire) qu'à une connaissance de certains bits (pouvant par exemple être déduits de variation dans le temps d'exécution du calcul de signature, ou de manière plus générale être obtenus par toute attaque par canaux auxiliaires).
Il est donc recommandé d'utiliser un générateur aléatoire dont les sorties soient retraitées conformément aux recommandations énoncées à la section 7.3.

6.3 Authentification d'entité

Un tel mécanisme permet à une entité de prouver son identité, en prouvant qu'elle connaît la clé privée associée à une clé publique connue de son interlocuteur. Les mécanismes d'authentification d'identité consistent en des protocoles interactifs utilisant des primitives de génération d'aléa, de hachage, de chiffrement asymétrique ou de signature électronique. Ces mécanismes, même s'ils utilisent des primitives de signature, forment une catégorie différente qui ne répond pas aux mêmes objectifs de sécurité. Par conséquent, une même clé ne doit jamais être utilisée par un mécanisme de signature et un mécanisme d'authentification d'entité.

Les recommandations et notes d'implémentation données en 4.4 s'appliquent également au cas des mécanismes d'authentification asymétriques.

6.4 Établissement de clé

Un mécanisme d'établissement de clé permet à deux entités (ou plus) de se mettre d'accord sur une valeur de clé secrète. Un tel mécanisme est habituellement combiné avec une authentification d'identité, reposant sur des clés d'authentification long-terme, soit des biclés asymétriques utilisés pour calculer des signatures, soit des clés symétriques partagées utilisées pour calculer des MAC.

Le mécanisme le plus utilisé pour l'établissement de clé secrète entre deux entités est le protocole de Diffie-Hellman. Il repose sur une instanciation du problème du logarithme discret dans un groupe où ce problème est suffisamment difficile. Ce mécanisme est le suivant :

1. Les deux utilisateurs conviennent d'un groupe G et d'un générateur g.
2. Les deux utilisateurs choisissent chacun un secret r_i ($i = 1, 2$), et communiquent à leur correspondant la valeur g^{r_i}.
3. Chaque utilisateur, à partir de son secret et de l'élément émis par l'autre utilisateur, peut alors calculer $g^{r_1 r_2} = (g^{r_1})^{r_2} = (g^{r_2})^{r_1}$.

Tel quel, ce protocole est vulnérable entre autres à une attaque de type *Man-in-the-middle*. Il faut donc lui ajouter des étapes garantissant l'authentification des deux parties en communication et l'authentification de l'échange.

R22

Mécanismes d'établissement de clé

Les mécanismes d'établissement de clé DH et EC-DH sont recommandés.

Primitive	Mécanisme	R/O	Notes
FF-DLOG	DH [SP800-56A, ISO11770-3]	R	6.4.a
EC-DLOG	EC-DH [SP800-56A, ISO11770-3]	R	6.4.a

Il est à noter que les schémas ci-dessus sont recommandés à condition d'utiliser des tailles de paramètres recommandées pour les primitives asymétriques sous-jacentes, cf Section 5.

Note d'implémentation.

Note 6.4.a : DH authentifié

Le mécanisme d'établissement de clé de Diffie-Hellman n'est pas authentifié et est donc vulnérable aux attaques de type *Man-in-the-middle*. Sa sécurité dépend donc de l'authentification des interlocuteurs et des données échangées (telles que les éléments du groupe considéré). Une telle authentification n'est possible qu'en présence de secrets long terme.

7
Génération d'aléa

7.1 Source d'aléa

Une source d'aléa est un processus probabiliste duquel on peut extraire une séquence de bits aléatoires. Il est difficile de s'assurer de la qualité des sorties d'une source d'aléa. Il y a principalement deux approches :

- **Réaliser des tests statistiques sur les sorties de la source.** Dans cette approche en boîte noire, aucune connaissance spécifique de la source n'est requise pour conduire le test. Cependant, cette approche souffre de deux défauts. D'abord, les tests statistiques sont génériques et ne peuvent être utilisés que pour détecter des défauts classiques de la source d'aléa par rapport à une source d'aléa idéale. De plus, cette approche n'apporte pas d'assurance sur la distribution des sorties de la source d'aléa. Malgré ces limitations, les tests statistiques restent pertinents pour détecter d'éventuelles défaillances de la source d'aléa.
- **Modéliser le processus stochastique de la source.** Cette approche requiert une compréhension approfondie du principe de construction de la source d'aléa, et essaie d'estimer la qualité de la source à partir de l'étude d'une modélisation théorique de son fonctionnement. Cette approche apporte une meilleure assurance, mais est difficile à mettre en œuvre, aussi bien de par l'accès qu'elle nécessite aux principes de conception de la source que par le niveau d'expertise qui doit être déployé par les évaluateurs, dans des domaines variés, notamment en physique, électronique et statistiques. Certains aspects, comme l'adéquation entre la source d'aléa et sa modélisation, restent difficiles à évaluer.

7.2 Générateur d'aléa déterministe

Le guide des mécanismes cryptographiques [ANSSI-MecCrypto] préconise la mise en œuvre d'un retraitement algorithmique, c'est-à-dire d'un générateur d'aléa déterministe (Deterministic Random Generator, DRG), initialisé, et éventuellement rafraîchi, par une source d'aléa. L'utilisation directe des sorties d'une source d'aléa à des fins cryptographiques ne respecte pas les exigences du RGS. Un DRG est un mécanisme cryptographique déterministe, construit autour d'un état interne qui peut être initialisé et rafraîchi par la sortie d'une source d'aléa, et dont des bits pseudo-aléatoires peuvent être extraits.

R23

Générateur d'aléa déterministe

HMAC-DRBG, Hash-DRBG et CTR-DRBG [FIPS197, ISO18033-3] sont recommandés.

Schéma	R/O	Notes
HMAC-DRBG [SP800-90A, ISO18031]	R	
Hash-DRBG [SP800-90A, ISO18031]	R	
CTR-DRBG [SP800-90A, ISO18031]	R	

Note d'implémentation.

Note 7.2.a : DRG-Seeding

La sécurité des générateurs d'aléa déterministes est conditionnée par la bonne initialisation de leur état interne à l'aide d'une source d'aléa qui fournit un germe [a]. L'entropie du germe [b] requise par les spécifications des générateurs d'aléa déterministes doit être respectée. De plus cette entropie doit être d'au moins 128 bits.

7.3 Génération de nombres aléatoires avec distribution de probabilité spécifique

En cryptographie asymétrique, il est fréquent d'avoir à générer des nombres aléatoires suivant certaines distributions spécifiques. Dans de telles situations, un générateur de bits aléatoires ne peut pas être utilisé directement, car il ne produit pas directement des nombres avec la loi de probabilité désirée. Par conséquent, il est nécessaire de retraiter les sorties des générateurs de bits aléatoires pour obtenir des nombres qui obéissent, au moins de manière suffisamment proche, à la distribution voulue.

7.3.1 Génération d'un entier modulo q

Des mécanismes asymétriques requièrent l'utilisation d'entiers tirés de manière aléatoire uniformément dans un intervalle de la forme $[0, q-1]$, où q n'est pas une puissance de 2. C'est notamment le cas lorsqu'il s'agit de tirer un élément dans un groupe de cardinal une puissance d'un nombre premier.

R24

Génération d'un entier modulo q

Les méthodes de génération aléatoire d'un entier modulo q par rejet et par utilisation d'aléa additionnel sont recommandées.

Schéma	R/O	Notes
Technique par rejet [FIPS186, Appendix B.1.2]	R	
Technique par utilisation d'aléa additionel [FIPS186, Appendix B.1.1]	R	

a. en anglais *seed*
b. mesure de sa quantité de variabilité

Note 7.3.a : Réduction modulaire de valeurs aléatoires

La technique de tirage aléatoire d'un entier modulo q consistant à générer un entier de $\ell = \lceil \log_2(q) \rceil$ bits et à le réduire modulo q introduit un biais non négligeable dans la loi de probabilité du résultat, ce qui peut conduire à des attaques sur le mécanisme cryptographique reposant sur le générateur de nombre aléatoire.

La technique par rejet assure la génération uniforme modulo q au prix d'appels additionels, en nombre variable, au générateur de bits aléatoires. La technique par utilisation d'aléa additionnel permet de réduire le biais de la méthode de génération d'un entier aléatoire modulo q de manière à ce que ce biais devienne négligeable, au prix de 64 bits d'aléa supplémentaire.

7.3.2 Génération de nombres premiers aléatoires

La génération de paramètres asymétriques et de clés RSA nécessite de générer des nombres premiers. La génération de premiers suit essentiellement la stratégie suivante. Pour commencer, un nombre de la taille appropriée est généré de manière aléatoire. Puis la primalité de ce nombre est testée. Tant que le test de primalité échoue, le nombre candidat est mis à jour et le test de primalité est réitéré. Le choix de la fonction de mise à jour offre un compromis entre la quantité d'aléa additionnel devant être utilisée et la distance entre la distribution des premiers finalement obtenue et la distribution d'un tirage uniforme parmi tous les nombres premiers d'une taille donnée. Le test de primalité peut être un test de pseudo-primalité, ou encore un test de primalité prouvée.

R25

Mécanismes de génération de nombres premiers

Les méthodes de génération de nombres premiers décrites en annexe de ce document sont recommandées.

Schéma	R/L	Notes
Génération de premier par rejet (Méthode 1) Annexe B.1	R	
Génération de premier par rejet améliorée (Méthode 2) Annexe B.2	R	

Ces méthodes ne sont pas vulnérables à des attaques de type ROCA [NSS+17].

R26

Tests de primalité

Le test de pseudo-primalité de Miller-Rabin est recommandé.

Schéma	R/L	Notes
Miller-Rabin [MVV96, Algorithm 4.24]	R	7.3.b

Note 7.3.b : Pseudo-primalité

Dans le cas d'une méthode de génération ne prouvant pas la primalité de son résultat, la probabilité que la sortie soit composite doit être inférieure à 2^{-128}.

Selon [FIPS186, Appendix F.1], la table 7.1 donne le nombre t d'itérations du test de Miller-Rabin qui est nécessaire pour atteindre cette probabilité quand la primalité d'un nombre aléatoire impair de k bits est testée. En particulier, 6 itérations sont nécessaires pour tester des candidats de 1024 bits, et 4 itérations sont nécessaires pour des candidats de 1536 bits. On note que chacune de ces itérations opère sur une base aléatoire a.

TABLE 7.1 – Nombre t d'itérations du test de Miller-Rabin en fonction de la taille en bit du candidat impair généré aléatoirement testé.

t	k
6	$950 \leq k < 1041$
5	$1041 \leq k < 1297$
4	$1297 \leq k < 1729$
3	$1729 \leq k < 2626$
2	$2626 \leq k < 5701$
1	$5701 \leq k$

7.3.3 Génération d'un module RSA

La génération d'un module RSA consiste à générer une paire de nombres premiers satisfaisant un ensemble de contraintes permettant d'assurer la taille de la clé, la cohérence du biclé, et d'éviter des clés faibles.

R27

Génération d'un module RSA

Le mécanisme de génération d'un module RSA décrit en annexe B.2.1 est recommandé.

Schéma	R/L	Notes
Génération de module RSA B.2.1	R	

Notes d'implémentation.

La génération de module RSA s'appuie sur un générateur de nombres premiers aléatoires. De plus, des conditions additionnelles doivent être satisfaites.

Note 7.3.c : Génération de biclé RSA

Les nombres premiers p et q doivent être générés aléatoirement, être de même taille, avec un produit de taille donnée. Les deux premiers ne doivent pas être trop proches, c'est-à-dire on doit avoir

$$|p - q| \geq 2^{\frac{n}{2}-100}.$$

Note 7.3.d : Taille de l'exposant privé

La taille de l'exposant privé d doit être suffisamment grande, c'est-à-dire $d > 2^{n/2}$, où n désigne la taille en bits du module. Ceci est garanti pour des petites valeurs de e.

Glossaire

Adversaire. Entité malveillante ayant pour but de remettre en cause l'objectif de sécurité d'un mécanisme cryptographique.

Algorithme cryptographique. Procédure de calcul bien définie qui prend un nombre d'entrées donné, produit une valeur de sortie, et satisfait une propriété de sécurité.

Authentification d'entité. Preuve de l'identité proclamée par une entité.

Authentification de l'origine des données. Confirmation que la source prétendue d'une donnée reçue est légitime.

Biclé asymétrique. Paire de clés liées mathématiquement, telles que la clé privée définit une transformation secrète et la clé publique définit une transformation publique.

Clé cryptographique. Séquence de symboles qui contrôle l'exécution d'une fonction cryptographique.

Clé secrète. Clé cryptographique utilisée dans des techniques de cryptographie symétrique par un ensemble d'entités prédéfini.

Confidentialité. Propriété qui assure qu'une information n'est ni disponible ni divulguée à des entités non autorisées.

Construction cryptographique. Mécanisme cryptographique construit à partir d'au moins un autre mécanisme cryptographique (voir *Primitive cryptographique*).

Gestion de clés. Application d'une politique de sécurité pour la génération, l'enregistrement, la certification, la radiation, la distribution, l'installation, le stockage, l'archivage, la révocation, la dérivation et la destruction de clés cryptographiques

Intégrité. Propriété qui assure que des données n'ont été ni modifiées ni détruites d'une manière non autorisée.

Mécanisme cryptographique. Terme général pour désigner une fonction de sécurité utilisant de la cryptographie.

Mode opératoire. Construction cryptographique basée sur un algorithme de chiffrement par bloc.

Objectif de sécurité. Objectif de prévention de menaces spécifiques et/ou de satisfaction d'une politique de sécurité. Par exemple, des objectifs de sécurité peuvent être la confidentialité des données, leur intégrité, etc.

Non-répudiation. Propriété qui assure qu'une entité ne peut pas renier un engagement ou une action passés.

Nonce. Valeur qui ne peut être utilisée plus d'une fois.

Primitive cryptographique. Mécanisme cryptographique utilisé pour construire un mécanisme de plus haut niveau (voir *Construction cryptographique*).

Protocole cryptographique. Protocole qui implémente une fonction de sécurité utilisant de la cryptographie et qui peut être considéré comme un schéma cryptographique interactif.

Schéma cryptographique. Algorithme cryptographique distribué faisant intervenir plusieurs entités (possédant généralement un secret partagé) qui atteint un objectif de sécurité.

Annexe A
Niveau de sécurité

A.1 Complexité des attaques

La complexité d'une attaque contre un mécanisme cryptographique mesure la quantité de ressources nécessaire à l'exécution de l'attaque.

- *Complexité en temps.* Elle correspond à la quantité de calculs requise par l'attaque à partir des données fournies par l'exécution du mécanisme cryptographique. Par exemple, la complexité en temps de la recherche exhaustive d'une clé AES-128 est d'environ 2^{128} opérations.
- *Complexité en mémoire.* Elle mesure la quantité de mémoire nécessaire pour réaliser l'attaque.
- *Complexité en données.* Elle mesure la quantité de données d'entrée et/ou de sortie du mécanisme que l'adversaire doit collecter afin de pouvoir exécuter son attaque (par exemple des clairs connus ou choisis et les chiffrés correspondants dans le cas d'un mécanisme de chiffrement).

Toutes ces mesures de complexité peuvent également être combinées en considérant les compromis temps/mémoire/données : par exemple il est parfois possible de diminuer la complexité en temps d'une attaque en en augmentant la complexité en mémoire ou en données.

A.2 Niveau de sécurité

Dans le présent guide, le *niveau de sécurité* d'un mécanisme cryptographique est défini comme la complexité en temps, c'est-à-dire le nombre d'opérations de la meilleure attaque connue contre ce mécanisme (sous certaines hypothèses sur les ressources d'un adversaire), exprimée comme un logarithme en base 2. Par exemple, 128 bits de sécurité signifie que 2^{128} opérations sont nécessaires à un adversaire.

Les recommandations et notes d'implémentation de ce guide sont guidées par les principes suivants :

1. Les mécanismes recommandés doivent offrir un niveau de sécurité contre les attaques hors ligne d'au moins 128 bits. Un niveau de sécurité de 100 bits reste actuellement toléré pour les mécanismes obsolescents, mais la marge de sécurité contre des attaques pratiques qui en résulte est plus faible.

2. Les niveaux de complexité en données acceptables sont plus bas. En effet, l'échange de données avec un processus, un équipement ou un serveur qui met en œuvre un mécanisme cryptographique est restreint par les limitations physiques du dispositif (par exemple le débit du canal de communication d'une carte à puce), ou des limitations qui peuvent être imposées par le dispositif lui-même (par exemple par l'expiration des clés cryptographiques). Il est donc suffisant d'analyser la sécurité du mécanisme contre des attaquants qui n'ont accès qu'à une quantité de données limitée. Par exemple, sur un réseau d'un débit de 100 gigabits, le temps nécessaire à l'échange de 2^{64} blocs AES est de plusieurs siècles. Il n'y a donc en pareil cas pas lieu de se préoccuper d'attaques nécessitant d'accéder à plus de 2^{64} blocs de données.

Une conséquence importante de la définition précédente du niveau de sécurité d'un mécanisme cryptographique comme la quantité minimale de calcul nécessaire pour l'attaquer concerne la taille minimale des clés. En effet, pour se prémunir contre la *recherche exhaustive*, attaque consistant à tester toutes les clés possibles, il est nécessaire que le nombre de clés excède 2^{κ}, où κ est le niveau de sécurité visé. La condition nécessaire précédente sur la taille des clés est loin d'être suffisante pour la classe très nombreuse des mécanismes pour lesquels il existe des attaques beaucoup plus efficaces que la recherche exhaustive et qui englobe notamment la totalité des mécanismes asymétriques.

A.3 Sécurité post-quantique

Des ordinateurs permettant de réaliser des calculs tirant partie des principes de la physique quantique sont envisagés de manière théorique depuis quelques décennies. Ce mode de calcul ouvre la possibilité de réaliser des algorithmes beaucoup plus efficaces que ceux exécutables sur un ordinateur classique. De tels algorithmes ont été conçus, dont certains menacent la sécurité des mécanismes cryptographiques actuels, permettant de résoudre les problèmes difficiles sur lesquels reposent les schémas asymétriques usuels en temps polynomial, et permettant d'obtenir une accélération sur les attaques génériques contre les mécanismes symétriques, comme la recherche exhaustive.

Bien qu'aucun ordinateur quantique de taille suffisamment grande pour permettre d'exécuter ces algorithmes sur des paramètres cryptographiques et donc remettre en cause la sécurité de mécanismes cryptographiques à l'état de l'art n'ait été réalisé à ce jour, le développement d'un tel dispositif fait l'objet d'un effort soutenu[8], et des experts estiment qu'il pourrait aboutir dans les prochaines décennies. Le présent guide ne recommande aucun mécanisme cryptographique asymétrique dont la sécurité résisterait à des attaques réalisées sur un tel ordinateur quantique. De tels mécanismes asymétriques seront introduits dans des révisions futures de ce guide, et feront suite à l'évaluation en cours de la sécurité de mécanismes candidats de ce type.

Les données et communications chiffrées qui sont enregistrées aujourd'hui pourraient être décryptées dans le futur par à une attaque mettant en œuvre un ordinateur quantique. Par conséquent, des mesures de protection doivent d'ores et déjà être prises si l'on souhaite protéger la confidentialité à long terme de données contre d'éventuelles attaques de ce type. Pour les mécanismes symétriques, il est alors recommandé d'adopter un niveau de sécurité pré-quantique de 256 bits.

8. avec notamment la réalisation de processeurs quantiques de taille modeste

Pour les mécanismes asymétriques, il est recommandé d'utiliser des solutions *hybrides*, combinant des mécanismes asymétriques éprouvés, mais vulnérables à des attaques exécutées par un ordinateur quantique, avec des mécanismes asymétriques supposés résistants à l'ordinateur quantique, mais pour lesquels l'assurance de robustesse vis-à-vis des attaques réalisées à l'aide d'ordinateur classique est moindre. Une telle combinaison permet en effet de ne pas dégrader le niveau d'assurance vis-à-vis des attaques classiques, et d'espérer obtenir une résistance vis-à-vis d'attaques quantiques. Les développeurs de systèmes devant protéger des informations au-delà de 2030 devraient considérer l'adoption de telles mesures, et préparer les moyens d'une migration des mécanismes cryptographiques qu'ils mettent en œuvre vers de nouveaux mécanismes (possiblement hybrides) résistants à la cryptanalyse quantique.

Annexe B
Génération de nombre premier et de biclé RSA

B.1 Génération de nombre premier par rejet (Méthode 1)

Entrée : un intervalle $I = [a, b] \cap \mathbb{N}$, dans lequel doit se situer le premier à générer, un test optionnel `Test` encodant des propriétés additionnelles du premier à générer, et un test de primalité `TestPrime`.

1. Choisir un nombre premier p selon la distribution uniforme sur I.
2. Si p n'est pas impair, aller à l'étape 1.
3. Si $\texttt{Test}(p) = \texttt{False}$, aller à l'étape 1.
4. Si $\texttt{TestPrime}(p) = \texttt{False}$, aller à l'étape 1.
5. Retourner p.

B.2 Génération de nombre premier par rejet améliorée (Méthode 2)

On note Π_B le produit des nombres premiers plus petits que B.

Entrée : un intervalle $I = [a, b] \cap \mathbb{N}$ dans lequel doit se situer le premier à générer, un petit entier naturel B satisfaisant $\Pi_B \ll b - a$, un test optionnel `Test` encodant des propriétés additionnelles du premier à générer, et un test de primalité `TestPrime`

1. Choisir $r \in \mathbb{N}$ selon la distribution uniforme sur $[1, \Pi_B[$.
2. Si $\text{pgcd}(r, \Pi_B) \neq 1$, aller à l'étape 1.
3. Choisir $k \in \mathbb{N}$ selon la distribution uniforme sur $[\lceil (a - r)/\Pi_B \rceil, \lfloor (b - r)/\Pi_B \rfloor]$.
4. En posant $p := k\Pi_B + r$, on a $p \in I$
5. Si $\texttt{Test}(p) = \texttt{False}$, aller à l'étape 3.
6. Si $\texttt{TestPrime}(p) = \texttt{False}$, aller à l'étape 3.
7. Retourner p.

B.2.1 Génération de biclé RSA

Entrée : n, la longueur en bit du module généré et e l'exposant public

1. Si e est pair ou $e \leq 2^{16}$, retourner ÉCHEC.
2. En utilisant une méthode de génération de nombre premier recommandée, générer un nombre premier p dans l'intervalle $[\frac{1}{\sqrt{2}}2^{n/2}, 2^{n/2}]$, tel que $\texttt{Test}(p)$: pgcd$(p-1, e) = 1$.
3. En utilisant une méthode de génération de nombre premier recommandée, générer un nombre premier q dans l'intervalle $[\frac{1}{\sqrt{2}}2^{n/2}, 2^{n/2}]$, tel que $\texttt{Test}(q)$: pgcd$(q-1, e) = 1$ et $|p-q| \geq 2^{n/2-100}$.
4. Calculer $d = e^{-1} \mod \text{ppccm}(p-1, q-1)$.
5. Si $d \leq 2^{n/2}$, aller à l'étape 1.
6. Retourner p, q.

Bibliographie

[ANSI-X9-63] American National Standards Institute.
ANSI X9.63-2011 – Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography, 2011.

[ANSSI-MecCrypto] Agence Nationale de la Sécurité des Systèmes d'Information.
Guide des Mécanismes Cryptographiques – Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, 2021.
version 2.04.

[BFK+12] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay.
Efficient Padding Oracle Attacks on Cryptographic Hardware.
In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 608–625. Springer, 2012.

[Ble06] Daniel Bleichenbacher.
RSA signature forgery based on implementation error, 2006.
Crypto 2006 rump session, rapport disponible à `https://www.ietf.org/mail-archive/web/openpgp/current/msg00999.html`.

[Ble98] Daniel Bleichenbacher.
Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1.
In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[BN00] M. Bellare and C. Namprempre.
Authenticated encryption : Relations among notions and analysis of the generic composition paradigm.
In *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[BSI-TR-03111] Bundesamt fur Sicherheit in der Informationstechnik.
Technical Guideline TR-03111, Elliptic Curve Cryptography.
version 2.0, 28.06.2012.

[ENISA-Algo] European Network and Information Security Agency.
Algorithms, key size and parameters report, 2014.

[FIPS180] National Institute of Standards and Technology.
FIPS PUB 180-4 : Secure Hash Standard (SHS), 2015.

[FIPS186] National Institute of Standards and Technology.
FIPS PUB 186-4 : Digital Signature Standard (DSS), 2013.

[FIPS197] National Institute of Standards and Technology.
FIPS PUB 197 : Advanced Encryption Standard (AES), 2001.

[FIPS202] National Institute of Standards and Technology.
FIPS PUB 202 : SHA-3 Standard : Permutation-Based Hash and Extendable-Output Functions, 2015.

[ISO10116] ISO/IEC.
ISO/IEC 10116 :2017 – Information technology – Security techniques – Modes of operation for an n-bit block cipher, 2017.

[ISO10118-3] ISO/IEC.
ISO/IEC 10118-3 :2018 – Information technology – Security techniques – Hash-functions – Part 3 : Dedicated hash-functions, 2018.

[ISO11770-3] ISO/IEC.
ISO/IEC 11770-3 :2015 – Information technology – Security techniques – Key management – Part 3 : Mechanisms using asymmetric techniques, 2015.

[ISO14888-3] ISO/IEC.
ISO/IEC 14888-3 :2018 – Information technology – Security techniques – Digital signatures with appendix – Part 3 : Discrete logarithm based mechanisms, 2018.

[ISO18031] ISO/IEC.
ISO/IEC 18031 :2011 – Information technology – Security techniques – Random bit generation, 2011.

[ISO18033-2] ISO/IEC.
ISO/IEC 18033-2 :2006 – Information technology – Security techniques – Encryption Algorithms – Part 2 : Asymmetric ciphers, 2006.

[ISO18033-3] ISO/IEC.
ISO/IEC 18033-3 :2010 – Information technology – Security techniques – Encryption Algorithms – Part 3 : Block ciphers, 2010.

[ISO19772] ISO/IEC.
ISO/IEC 19772 :2009 – Information technology – Security techniques – Authenticated encryption, 2009.

[ISO9796-2] ISO/IEC.
ISO/IEC 9796-2 :2010 – Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2 : Integer factorization based mechanisms, 2010.

[ISO9797-1] ISO/IEC.
ISO/IEC 9797-1 :2011 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1 : Mechanisms using a block cipher, 2011.

[ISO9797-2] ISO/IEC.
ISO/IEC 9797-2 :2011 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2 : Mechanisms using a dedicated hash-function, 2011.

[Lel13] J. Lell.
Practical malleability attack against CBC-Encrypted LUKS partitions, 2013.
http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions.

[Man01] James Manger.
A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0.

In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238. Springer, 2001.

[MVV96] A. Menezes, P. von Oorschot, and O. Vanstone.
Handbook of Applied Cryptography.
CRC Press, 1996.

[NSS+17] Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec, and Vashek Matyas.
The Return of Coppersmith's Attack : Practical Factorization of Widely Used RSA Moduli.
In *24th ACM Conference on Computer and Communications Security (CCS'2017)*, pages 1631–1648. ACM, 2017.

[PDM+18] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk.
Efail : Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels.
In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 549–566. USENIX Association, 2018.

[PKCS1] RSA Laboratories.
PKCS #1 v2.2 : RSA Cryptography Standard, 2012.

[RFC2104] H. Krawczyk, M. Bellare, and R. Canetti.
HMAC : Keyed-Hashing for Message Authentication, 1997.

[RFC3526] T. Kivinen and M. Kojo.
More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), 2003.

[RFC5297] D. Harkins.
Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES), 2008.

[RFC5639] Johannes Merkle and Manfred Lochter.
Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation.
RFC 5639, 2010.

[RFC5869] H. Krawczyk and P. Eronen.
HMAC-based Extract-and-Expand Key Derivation Function (HKDF), 2010.

[RFC7748] A. Langley, M. Hamburg, and S. Turner.
Elliptic Curves for Security, 2016.

[RFC8017] Ed. K. Moriarty, B. Kaliski, J. Jonsson, and A. Rush.
Public-Key Cryptography Standard (PKCS) #1 : RSA Cryptography Specifications Version 2.2, 2016.

[RFC8018] Ed. K. Moriarty, B. Kaliski, and A. Rusch.
PKCS #5 : Password-Based Cryptography Specification Version 2.1, 2017.

[RFC8439] Y. Nir and A. Langley.
ChaCha20 and Poly1305 for IETF Protocols.
RFC 8439, 2018.

[SCES-ACM] SOG-IS Crypto Working Group.
SOG-IS Crypto Evaluation Scheme, Agreed Cryptographic Mechanisms, 2020.
version 1.2, "https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf".

[SP800-38A] National Institute of Standards and Technology.
SP800-38A : Recommendation for Block Cipher Modes of Operation, 2001.

[SP800-38Aadd] National Institute of Standards and Technology.
SP800-38A-Addendum : Recommendation for Block Cipher Modes of Operation : Three Variants of Ciphertext Stealing for CBC Mode, 2010.

[SP800-38B] National Institute of Standards and Technology.
SP800-38B : Recommendation for Block Cipher Modes of Operation : The CMAC Mode for Authentication, 2005.

[SP800-38C] National Institute of Standards and Technology.
SP800-38C : Recommendation for Block Cipher Modes of Operation : The CCM Mode for Authentication and Confidentiality, 2004.

[SP800-38D] National Institute of Standards and Technology.
SP800-38D : Recommendation for Block Cipher Modes of Operation : Galois/Counter Mode (GCM) and GMAC, 2007.

[SP800-38E] National Institute of Standards and Technology.
SP800-38E : Recommendation for Block Cipher Modes of Operation : The XTS-AES Mode for Confidentiality on Storage Devices, 2010.

[SP800-38F] National Institute of Standards and Technology.
SP800-38F : Recommendation for Block Cipher Modes of Operation : Methods for Key Wrapping, 2012.

[SP800-56A] National Institute of Standards and Technology.
SP800-56A Rev. 3 : Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, 2018.

[SP800-56C] National Institute of Standards and Technology.
SP800-56C Rev. 2 : Recommendation for Key-Derivation Methods in Key-Establishment Schemes, 2020.

[SP800-67] National Institute of Standards and Technology.
SP800-67 Rev. 2 : Recommendation for the Triple Data Encyption Algorithm (TDEA) Block Cipher, 2017.

[SP800-90A] National Institute of Standards and Technology.
SP800-90A Rev. 1 : Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2015.

[Vau02] Serge Vaudenay.
Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ...
In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.

Liste des recommandations

ANSSI-PA-079
Version 1.0 - 8/3/2021

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

ANSSI - 51, boulevard de La Tour-Maubourg, 75700 PARIS 07 SP
www.ssi.gouv.fr / conseil.technique@ssi.gouv.fr

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
Premier ministre
SGDSN