RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

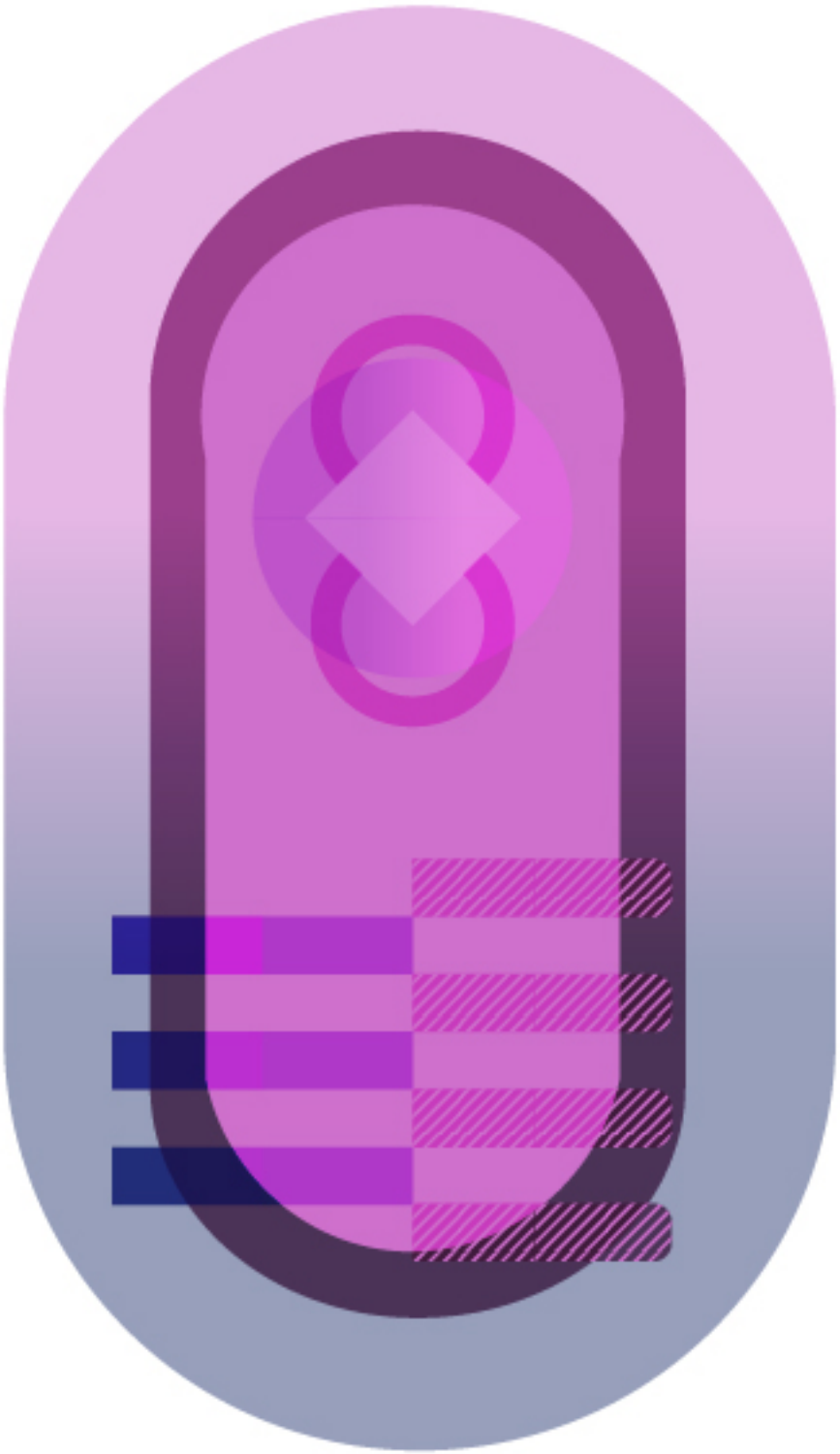
RÈGLES ET RECOMMANDATIONS CONCERNANT LE CHOIX ET LE DIMENSIONNEMENT DES MÉCANISMES CRYPTOGRAPHIQUES

GUIDE ANSSI

ANSSI-PG-083
2026-03-20

PUBLIC VISÉ :

Développeur Administrateur RSSI DSI Utilisateur



Informations

Attention

Ce document rédigé par l'ANSSI s'intitule **« Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques »**. Il est téléchargeable sur le site `cyber.gouv.fr`.

Il constitue une production originale de l'ANSSI placée sous le régime de la « Licence Ouverte v2.0 » publiée par la mission Etalab.

Conformément à la Licence Ouverte v2.0, le document peut être réutilisé librement, sous réserve de mentionner sa paternité (source et date de la dernière mise à jour). La réutilisation s'entend du droit de communiquer, diffuser, redistribuer, publier, transmettre, reproduire, copier, adapter, modifier, extraire, transformer et exploiter, y compris à des fins commerciales.

Sauf disposition réglementaire contraire, les recommandations n'ont pas de caractère normatif ; elles 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
1.02	2004-11-19	Première version applicable
1.10	2006-12-19	
1.20	2010-01-26	
2.00	2012-06-18	
2.02	2013-03-29	
2.04	2020-01-01	
3.00	2026-03-20	Menace quantique prise en compte

Table des matières

1
Introduction

1.1 Objectif du document

La cryptographie moderne met à la disposition des concepteurs de systèmes d'information des outils permettant d'assurer, ou de contribuer à assurer, des fonctions de sécurité telles que la confidentialité, l'intégrité, l'authenticité et la non-répudiation. Ces outils sont souvent qualifiés d'algorithmes, de primitives ou encore de **mécanismes cryptographiques**.

La science cryptographique a connu des développements majeurs au cours des dernières décennies. Bien que son évolution soit loin d'être achevée, comme le montre par exemple l'émergence relativement récente de nouvelles familles d'algorithmes asymétriques présumés résistants à la menace quantique, elle semble avoir atteint un degré de maturité suffisant pour permettre de dégager des règles générales concernant le choix et le dimensionnement corrects des mécanismes. Ce document vise à expliciter ces règles ainsi que certaines recommandations.

1.2 Positionnement du document

Ce document traite des règles et recommandations concernant le choix et le dimensionnement de mécanismes cryptographiques.

1.3 Limites du champ d'application du document

Sont explicitement exclus de ce document :

- les règles et recommandations concernant la gestion des clés cryptographiques, qui font l'objet d'un document séparé [2] ;
- la liste exhaustive des mécanismes cryptographiques recommandés permettant d'atteindre les différents niveaux de robustesse cryptographique définis dans ce document, bien que certaines primitives très classiques soient données en exemple. Une telle liste peut être trouvée dans le guide de sélection des mécanismes cryptographiques édité par l'ANSSI [3] ;
- les aspects liés à l'implémentation des mécanismes et en particulier au choix du support ainsi qu'à la sécurité de l'implémentation face aux attaques par canaux auxiliaires ou par injection de fautes ;

- les méthodes d'évaluation des mécanismes cryptographiques, qui reposent avant tout sur une connaissance précise de l'état de l'art en cryptographie ;
- les méthodes d'analyse de menaces et de développement de produits cryptographiques menant à choisir les mécanismes cryptographiques permettant d'assurer les fonctions de sécurité identifiées ainsi que les niveaux de robustesse cryptographique nécessaires ;
- les liens entre niveau de robustesse d'un mécanisme cryptographique et niveau de robustesse d'un produit tel que défini dans les processus de qualification ou d'évaluation selon une méthode normalisée telle que les Critères Communs ;
- les fournitures nécessaires à l'évaluation de mécanismes cryptographiques, qui font l'objet d'un document séparé [24] ;
- les communications circulant en superposition sur un canal quantique. Les techniques de distribution quantique de clés (QKD), et plus généralement toute la cryptographie quantique, en sont donc exclues. C'est également le cas des attaques dans lesquelles on suppose que l'adversaire dispose d'un accès en superposition à un oracle manipulant des secrets (par exemple, un oracle de déchiffrement) ;
- les mécanismes non cryptographiques assurant cependant des fonctions de sécurité tels que l'emploi de mots de passe et l'usage de la biométrie [4], même si ceux-ci utilisent des mécanismes cryptographiques.

1.4 Définition des règles et recommandations

Les **règles** définissent des principes qui doivent a priori être suivis par tout mécanisme. L'observation de ces règles est une condition nécessaire à la reconnaissance du bon niveau de sécurité du mécanisme. Un mécanisme respectant ces règles est dit *conforme au référentiel*. Cependant, le fait de suivre l'ensemble des règles, qui sont par nature très génériques, n'est pas suffisant pour garantir la robustesse du mécanisme cryptographique ; seule une analyse spécifique permet de s'en assurer.

En plus des règles, le présent document définit également des **recommandations**. Celles-ci expriment des conditions supplémentaires pour qu'un mécanisme puisse être considéré comme pleinement à l'état de l'art d'un point de vue technique et qu'il assure une marge substantielle de sécurité. Leur application n'est pas formellement requise pour garantir la sécurité d'un mécanisme cryptographique : il est possible de ne pas suivre certaines recommandations si elles représentent une contrainte majeure en termes de fonctionnalité, de performances ou de coût d'un produit.

Certaines **règles post-quantiques** et **recommandations post-quantiques** viennent compléter si nécessaire l'énoncé des règles et recommandations par des conditions supplémentaires applicables seulement lorsqu'une résistance aux attaques assistées par ordinateurs quantiques, appelées attaques quantiques, est visée. L'observation d'une règle, qu'elle soit post-quantique ou non, est une condition nécessaire à la reconnaissance qu'un mécanisme cryptographique offre des assurances de protection contre un adversaire doté d'un ordinateur quantique, appelé dans la suite adversaire quantique. Une telle règle

doit donc être suivie par tout mécanisme visant à assurer une protection contre la menace posée par le calcul quantique, aussi appelée menace quantique. L'observation d'une recommandation post-quantique est une condition nécessaire à la reconnaissance qu'il est conforme à l'état de l'art de la cryptographie post-quantique et qu'il offre des assurances de protection contre un adversaire quantique avec des marges de sécurité substantielles.

Dans un souci de transparence, les règles et recommandations contenues dans ce document sont le plus souvent accompagnées de justifications et d'informations complémentaires. Les choix sont effectués en tenant compte le plus rigoureusement possible de l'état de l'art actuel en cryptographie ainsi que des contraintes pratiques liées à sa mise en œuvre. À fins d'illustration, des exemples de mécanismes conformes ou non conformes aux règles et recommandations sont parfois donnés. Ces listes n'ont pas vocation à être exhaustives et des recommandations plus détaillées peuvent être trouvées dans le guide ANSSI de sélection des mécanismes cryptographiques [3].

La définition des règles et des recommandations prend également en compte certaines hypothèses classiques telles que la loi de Moore ou d'autres plus actuelles sur l'évolution de la puissance de calcul disponible, ce qui permet de définir des règles et des recommandations de manière suffisamment objective et scientifique pour fixer un cadre acceptable par tout professionnel en sécurité des systèmes d'information. Il va cependant de soi qu'une telle analyse ne peut tenir compte d'éventuels événements « catastrophiques » tels qu'une cryptanalyse opérationnelle de l'AES ou la découverte d'une méthode de factorisation efficace sur de grands nombres.

Par ailleurs, l'estimation des niveaux de résistance qui seront nécessaires afin de garantir la sécurité à 10 ou 20 ans des informations est délicate. Elle est cependant requise par de nombreuses applications comme le maintien de la confidentialité de certaines informations ou la signature électronique qui nécessite souvent une validité à long terme. De plus, lors de la définition d'un produit, il est nécessaire d'avoir une vision dont le terme est dicté par la durée de vie envisagée. Il est possible de résoudre certains problèmes par des moyens techniques (surchiffrement régulier d'informations devant être protégées à long terme, horodatage et signature régulière de documents notariés, etc.); cette approche est parfois indispensable mais ne peut être généralisée à cause des contraintes qu'elle impose. Par conséquent, bien que l'analyse développée dans ce guide doive être prise avec précaution, elle vise à être valable pendant au moins 15 ans.

1.5 Organisation du document

Ce document est organisé de la manière suivante :

- l'ensemble des règles et recommandations sont contenues dans la section 2; elles sont repérées selon la codification suivante : les premières lettres (Règle ou Reco) indiquent si l'on a affaire à une règle ou à une recommandation, le domaine d'application est ensuite précisé et, finalement, un chiffre permet de distinguer les règles d'une même catégorie. Par exemple, **RègleFactorisation.3** désigne la règle numéro 3 concernant le problème de la factorisation. Une liste non-exhaustive de mécanismes conformes ou non conformes est parfois proposée en illustration des règles et recommandations;

- un rappel non mathématique des principaux concepts cryptographiques nécessaires à la compréhension de ce document est proposé dans l'annexe A ;
- des informations issues de publications du milieu académique sur le dimensionnement des mécanismes cryptographiques sont regroupées dans l'annexe B ;
- les références bibliographiques apparaissent dans l'annexe C.

Ce document ne comporte volontairement aucune table récapitulative des tailles minimales de paramètres requis. La concision a été privilégiée dans l'expression des règles et recommandations ; vouloir les résumer à une simple valeur numérique pourrait être une source d'erreur et de confusion.

1.6 Mise à jour du document

Ce document ayant en particulier pour but de fixer des bornes numériques, par exemple en termes de tailles de clés, il convient de le mettre à jour tous les deux à cinq ans, en fonction des révisions qu'appellent l'évolution de l'état de l'art et le traitement des commentaires ou retours reçus à son sujet. Les commentaires peuvent être adressés par e-mail à conseil.technique@ssi.gouv.fr.

2
Règles et recommandations

Les règles et recommandations contenues dans ce document sont organisées de manière très comparable aux rappels cryptographiques proposés en annexe A. Elles s'adressent à un lecteur familier avec ces concepts, qui ne sont par conséquent pas systématiquement rappelés.

Bien qu'il subsiste des contextes où la sécurité post-quantique n'est pas une obligation, il est possible de formuler la recommandation générale suivante.

Reco

RecoSécuLongTerme

En cas d'utilisation prévue au-delà du 1er janvier 2030 d'un mécanisme cryptographique ou de l'existence d'un risque d'attaque rétroactive sur celui-ci, il est recommandé de viser une sécurité post-quantique.

Une autre recommandation générale, qui vaut pour tous les types de mécanismes, est la suivante.

Reco

RecoMécanismesÉprouvés

Il est recommandé d'employer des mécanismes cryptographiques éprouvés et reconnus par la communauté académique.

Information

- Certains mécanismes d'échange de clé ou de chiffrement asymétrique sont vulnérables à une attaque rétroactive. Par exemple, un adversaire ayant enregistré un échange de clé de Diffie-Hellman ainsi que les communications chiffrées sous la clé déduite de cet échange pourra, lorsqu'il disposera d'un ordinateur quantique, résoudre le problème de Diffie-Hellman calculatoire et compromettre la confidentialité des données échangées. On parle d'attaque de type « *store now, decrypt later* ».
- Les mécanismes d'authentification sont dans de nombreux cas moins vulnérables aux attaques rétroactives. Toutefois, pour des produits à longue durée de vie, il est nécessaire dès à présent d'utiliser des mécanismes d'authentification post-quantiques.
- La condition temporelle donnée dans la recommandation **RecoSécuLongTerme** correspond à une estimation très approximative (à la date de la rédaction du présent document) d'un ordre de grandeur du temps minimal

nécessaire à la maturité des technologies de calcul quantique ; cette durée pourra être révisée en fonction des progrès constatés dans ces technologies.

2.1 Cryptographie symétrique

2.1.1 Taille de clé symétrique

Dans cette section sont définies les propriétés attendues de clés utilisées par des mécanismes symétriques. Dans ce document, la taille d'une clé est le nombre de bits effectifs de cette clé. Par exemple, le DES utilise des clés de 64 bits mais seuls 56 de ces bits peuvent être choisis aléatoirement, les 8 bits restants servant de contrôle de parité. C'est pourquoi les clés DES considérées ont une taille de 56 bits.

Les tailles minimales définies ci-dessous n'ont de valeur que sous l'hypothèse que la meilleure attaque classique permettant de mettre en défaut le mécanisme symétrique employé consiste à effectuer une recherche exhaustive sur l'espace des clés. Cette attaque étant générique, le respect des règles définies ci-dessous est une condition nécessaire qui ne peut être considérée comme suffisante. Une analyse cryptographique complémentaire du mécanisme est indispensable.

Règle

RègleTailleCléSym

La taille minimale des clés symétriques est de 128 bits.

RecoPQ

RecoPQTailleCléSym

Lorsqu'une sécurité post-quantique est visée, il est recommandé d'employer des clés symétriques d'au moins 192 bits.

i

Information

- Bien que la capacité de calcul d'une organisation motivée soit complexe à estimer précisément, l'emploi de clés dont la longueur ne dépasse pas significativement 100 bits est risqué (voir annexes A.1.1 et B.1). En particulier, une taille de clé de 64 bits est clairement insuffisante puisqu'il est possible aujourd'hui de retrouver une clé de cette taille par recherche exhaustive. De plus, on estime que les clés de 80 bits ne sont pas hors de portée des moyens de calcul classique actuels. Seul l'emploi de clés de longueur significativement supérieure à 100 bits offre des marges de sécurité suffisantes. En pratique, l'usage extrêmement répandu de l'algorithme AES favorise l'emploi de clés de 128 bits ou plus.
- L'emploi de clés de 128 bits permet de s'assurer que les attaques génériques classiques par recherche exhaustive seront inopérantes à long terme. Ceci n'implique cependant pas que tout mécanisme utilisant de telles clés soit cryptographiquement sûr.
- Dans les contextes où une résistance à la menace quantique est visée, il est nécessaire de se prémunir durablement contre les accélérations quan-

tiques de la recherche générique de clé permises par exemple par l'algorithme de Grover [23]. Des clés de longueur 192 bits ou 256 bits fournissent une marge de sécurité substantielle contre de telles accélérations.

- L'emploi de clés de 128 bits peut néanmoins être présumé suffisant pour que les attaques génériques reposant sur l'emploi d'algorithmes quantiques tels que l'algorithme de Grover soient inopérantes à assez long terme. De tels algorithmes emploient en effet des circuits quantiques de profondeur proportionnelle au facteur d'accélération quantique visé, ce qui limite fortement ce facteur. La recommandation **RecoPQTailleCléSym** pourrait cependant devoir être convertie en une règle post-quantique en cas d'évolutions majeures dans les technologies de calcul quantique.
- L'emploi de clés d'au moins 192 bits n'induit généralement qu'une faible dégradation des performances comme l'illustre le faible surcoût d'AES-192 par rapport à AES-128.

Mécanismes conformes

- La primitive de chiffrement par bloc AES-128 est conforme à la règle **RègleTailleCléSym** mais non conforme à la recommandation **RecoPQTailleCléSym**.
- Les primitives de chiffrement par bloc AES-192 et AES-256 sont conformes à la règle et à la recommandation précédentes.

Mécanisme non conforme

La primitive de chiffrement par bloc Triple-DES deux clés ne respecte pas la règle **RègleTailleCléSym** puisque sa longueur de clé est seulement de 112 bits.

2.1.2 Chiffrement symétrique

2.1.2.1 Chiffrement par bloc

Un mécanisme de chiffrement par bloc est la combinaison d'une primitive de chiffrement par bloc et d'un mode opératoire de chiffrement. Il permet de chiffrer des données de taille quelconque en les traitant par blocs de taille fixe. Selon le contexte, le terme chiffrement par bloc peut désigner un mécanisme de chiffrement par bloc ou une primitive de chiffrement par bloc.

Primitive de chiffrement par bloc. Une primitive de chiffrement par bloc est une fonction inversible paramétrée par une clé secrète qui à un bloc de clair de n bits associe un bloc de chiffré de n bits. Les deux caractéristiques principales d'une primitive de chiffrement par bloc sont la taille effective de la clé ainsi que la taille n des blocs traités (voir annexe A.1.1). Les règles et recommandations concernant la taille effective de la clé ont été présentées dans la section précédente.

Règle

RègleTailleBlocSym

Une primitive de chiffrement par bloc doit opérer sur des blocs d'au moins 128 bits.

Information

Les primitives de chiffrement par bloc sont utilisées par des modes opératoires pour chiffrer des messages de taille quelconque ou pour calculer des codes d'authentification de message. La taille du bloc intervient dans l'estimation de la sécurité des mécanismes résultants de chiffrement ou d'authentification de message. La principale menace est la présence d'attaques exploitant le paradoxe des anniversaires, ayant une probabilité de succès non négligeable dès qu'un nombre de blocs de l'ordre de $2^{n/2}$ sont traités, où n désigne la taille en bits du bloc (voir annexe A.1). Dans le cas de blocs de 64 bits, cette limite de sécurité correspond au traitement de quelques gigaoctets de données, ce qui peut être très rapidement atteint pour certaines applications. Utiliser des blocs de 128 bits permet de se prémunir contre de telles attaques dans la plupart des cas.

Mécanisme conforme

Les primitives de chiffrement par bloc AES-128, AES-192 et AES-256 sont conformes à la règle **RègleTailleBlocSym**.

Mécanisme non conforme

Les primitives de chiffrement par bloc Triple-DES deux clés et Triple-DES trois clés ne respectent pas la règle **RègleTailleBlocSym** puisqu'elles opèrent sur une taille de bloc de 64 bits.

Le choix d'une primitive de chiffrement par bloc repose sur la prise en compte des règles et recommandations liées à la taille de la clé ainsi qu'à la taille du bloc. Au-delà de la simple considération de ces deux dimensions, il faut surtout prendre en compte la sécurité intrinsèque apportée par la primitive face à des attaques plus évoluées que la simple recherche exhaustive sur la clé (cryptanalyse linéaire, différentielle, etc.). Ces attaques ont pour but de retrouver la clé ou, plus modestement, de distinguer le chiffrement par bloc d'une permutation aléatoire. Pour ce faire, un attaquant peut par exemple observer des chiffrés de blocs de clair connus, ou plus généralement obtenir le chiffrement ou le déchiffrement de blocs de son choix.

Dans une attaque classique (respectivement quantique), on désigne par *opération de calcul* un équivalent de la complexité temporelle du chiffrement d'un bloc au moyen d'un circuit classique (respectivement quantique). On considère généralement qu'une attaque est qualifiée par :

- le nombre $\mathtt{N_{op}}$ d'opérations de calcul hors-ligne ;[1]
- le nombre $\mathtt{N_{bloc}}$ de paires de blocs clair/chiffré utilisées ;
- la quantité $\mathtt{N_{mem}}$ de mémoire ;
- dans le cas d'une attaque quantique, la profondeur $\mathtt{N_{prof}}$ du calcul. Cette quantité est définie comme le nombre maximal d'opérations de calcul séquentielles effectuées lors de l'attaque.

Règle

RèglePrimChiffBloc

Une primitive de chiffrement par bloc ne doit être vulnérable à aucune attaque classique nécessitant un nombre d'opérations de calcul $\mathtt{N_{op}}$ inférieur à 2^{128}.

RègPQ

RèglePQPrimChiffBloc

Lorsqu'une sécurité post-quantique est visée, une primitive de chiffrement par bloc ne doit être vulnérable à aucune attaque quantique nécessitant un nombre d'opérations de calcul $\mathtt{N_{op}}$ inférieur à 2^{80} et de profondeur $\mathtt{N_{prof}}$ inférieure à 2^{48}.

RecoPQ

RecoPQPrimChiffBloc

Lorsqu'une sécurité post-quantique est visée, il est recommandé d'employer une primitive de chiffrement par bloc qui n'est vulnérable à aucune attaque quantique nécessitant un nombre d'opérations de calcul $\mathtt{N_{op}}$ inférieur à 2^{128} et de profondeur $\mathtt{N_{prof}}$ inférieure à 2^{64}.

Information

- Les règles et recommandations précédentes ne mentionnent ni $\mathtt{N_{bloc}}$ ni $\mathtt{N_{mem}}$. Ceci tient essentiellement à la volonté de ne pas trop compliquer leur énoncé.
- La règle **RèglePrimChiffBloc** ignore les attaques basées sur des stratégies de recherche exhaustive sur la clé complète.

Mécanismes conformes

- La primitive de chiffrement par bloc AES-128 est conforme aux règles **RèglePrimChiffBloc**, **RèglePQPrimChiffBloc**, mais elle ne respecte pas la recommandation **RecoPQPrimChiffBloc**.
- Les primitives de chiffrement par bloc AES-192 et AES-256 sont conformes aux règles **RèglePrimChiffBloc** et **RèglePQPrimChiffBloc**. Elles respectent

1. Dans le cas où une attaque quantique fait à la fois appel à des calculs quantiques et classiques, on conviendra, afin de simplifier les raisonnements, qu'une opération de calcul classique et quantique contribuent de la même manière à $\mathtt{N_{op}}$.

de plus la recommandation **RecoPQPrimChiffBloc**.

Mécanisme non conforme

La primitive de chiffrement par bloc Triple-DES trois clés est non conforme à la règle **RèglePrimChiffBloc** puisqu'elle est vulnérable à une attaque en 2^{112} opérations.

Modes opératoires de chiffrement. L'utilisation d'un mode opératoire de chiffrement permet d'assurer la confidentialité de messages de taille quelconque à partir d'une primitive de chiffrement par bloc. Comme expliqué dans l'annexe A.1.1.1, une simple primitive de chiffrement par bloc ne permet pas d'assurer une telle fonction, en particulier à cause de sa nature fondamentalement déterministe et de la taille imposée des blocs de données traités.

Le choix d'un mode opératoire de chiffrement est très dépendant de la nature des données traitées et du modèle de sécurité envisagé pour ce mécanisme. Les règles et recommandations se veulent malgré tout relativement génériques.

Règle

RègleModeChiff

Dans le modèle de sécurité pertinent pour l'usage du mode opératoire de chiffrement, il ne doit exister aucune attaque de complexité $\mathtt{N}_{\mathtt{op}}$ inférieure à celle de la recherche exhaustive exploitant un nombre $\mathtt{N}_{\mathtt{bloc}}$ de blocs chiffrés sous une même clé significativement inférieur à $2^{n/2}$ appels de la primitive de chiffrement par bloc sous-jacente, où n est la taille en bits du bloc.

Reco

RecoModeChiff

1. Il est recommandé d'employer un mode opératoire de chiffrement non déterministe.
2. Il est recommandé d'employer un mode opératoire de chiffrement disposant d'une preuve de sécurité dans un modèle d'adversaire pertinent vis-à-vis du contexte d'utilisation.
3. Il est recommandé de ne pas employer isolément un mode opératoire de chiffrement n'offrant pas d'intégrité.

Information

- De nombreux modes, tels que le mode CBC (voir annexe A.1.1.1), ne sont sûrs que si l'on traite significativement moins de $2^{n/2}$ blocs de messages avec la même clé, où n désigne la taille en bits du bloc.
- Pour garantir la confidentialité des informations, un mode opératoire de chiffrement ne doit pas être déterministe. Cela permet notamment d'éviter que le chiffrement d'un même message fournisse le même chiffré. L'emploi d'un vecteur d'initialisation (IV [2]) et d'un mode opératoire adapté

permet de résoudre ce problème. Le guide de sélection d'algorithmes cryptographiques [3] fournit des exemples de tels modes opératoires.

- Le besoin de confidentialité est souvent associé à un besoin d'intégrité, même si ce dernier semble parfois moins évident à première vue. En particulier, aucun des modes opératoires de chiffrement classiques (CBC, OFB, CFB, CTR) n'apporte la moindre protection en intégrité (voir annexe A.1.2). Il existe essentiellement deux manières d'obtenir conjointement des propriétés de confidentialité et d'intégrité : soit par une combinaison de mécanismes de chiffrement et d'intégrité utilisés avec des clés différentes, soit par un mode opératoire de chiffrement authentifié paramétré par une seule clé. [3]
- Il existe des modes garantissant une sécurité même lorsque $\mathtt{N_{bloc}}$ dépasse la limite de $2^{n/2}$: ces modes sont dit *beyond birthday-bound* car ils sont sûrs malgré un dépassement de la borne du paradoxe des anniversaires ($\mathtt{N_{bloc}} \geq 2^{n/2}$).

Mécanismes conformes

- Le mode opératoire de chiffrement CBC utilisant une primitive de chiffrement conforme au référentiel comme l'AES et des IV aléatoirement choisis pour chaque message et transmis en clair est un mécanisme de chiffrement par bloc conforme à la règle **RègleModeChiff** et aux recommandations **RecoModeChiff.1** et **RecoModeChiff.2**. Ce mécanisme est rappelé en annexe A.1.1.1. Il est particulièrement important de garantir que les IV sont générés dans le périmètre de sécurité du chiffrement – par exemple dans le composant sécurisé où le mode opératoire de chiffrement et la primitive sous-jacente sont implémentés et non hors de ce composant – et avec un générateur d'aléa sûr. Ces IV ne doivent en aucun cas pouvoir être contrôlés ou prédits par un attaquant.
- Le mécanisme de chiffrement authentifié AES-GCM est conforme à la règle et aux recommandations précédentes. Il est particulièrement important de garantir qu'un même IV n'est jamais réutilisé sous une même clé.

2.1.2.2 Chiffrement par flot

Les mécanismes de chiffrement par flot [4] constituent l'autre grande famille de mécanismes de chiffrement symétrique. Un mécanisme de chiffrement par flot associe à une clé et un IV une suite chiffrante de même longueur que la donnée en clair, additionnée bit par bit avec cette dernière pour produire le chiffré. Le modèle de sécurité généralement retenu pour l'évaluation d'un tel mécanisme est l'indistinguabilité pour l'attaquant des suites chiffrantes produites sous des IV choisis distincts de suites binaires parfaitement aléatoires indépendantes.

2. De l'anglais « Initialisation Vector ».
3. On peut indifféremment parler de « chiffrement authentifié » ou de « chiffrement intègre ».
4. « Stream cipher » en anglais.

Il convient de différencier les mécanismes de chiffrement par flot « dédiés » conçus spécialement pour cet usage de certains mécanismes de chiffrement par bloc employant des modes opératoires de chiffrement produisant une suite chiffrante (CTR, OFB, etc.). Les règles et recommandations suivantes ne concernent que les mécanismes de chiffrement par flot dédiés.

Règle

RègleChiffFlot

Un mécanisme de chiffrement par flot ne doit être vulnérable à aucune attaque classique nécessitant un nombre d'opérations $\mathtt{N}_{\mathtt{op}}$ inférieur à 2^{128} et au plus 2^{64} bits de flot de sortie.

RègPQ

RèglePQChiffFlot

Lorsqu'une sécurité post-quantique est visée, un mécanisme de chiffrement par flot doit employer un état interne d'au moins 256 bits.

Reco

RecoChiffFlot

1. Il est recommandé d'employer des mécanismes de chiffrement par flot utilisant un état interne d'au moins 256 bits.
2. Il est recommandé de compléter l'emploi d'un mécanisme de chiffrement par flot par un mécanisme d'intégrité.

Information

Les chiffrés produits par un mécanisme de chiffrement par flot sont facilement malléables, en ce sens qu'inverser un bit de chiffré induit une inversion de bit correspondant du clair. Sans mécanisme d'intégrité complémentaire, cette malléabilité peut être exploitée par un attaquant pour modifier le clair de manière contrôlée.

Mécanisme conforme

Le mécanisme de chiffrement par flot Chacha20 est conforme aux règles **RègleChiffFlot**, **RèglePQChiffFlot** et à la recommandation **RecoChiffFlot.1**, mais son emploi est non conforme à la recommandation **RecoChiffFlot.2** s'il n'est pas complété par l'utilisation d'un mécanisme d'intégrité.

2.1.3 Intégrité cryptographique

Un mécanisme d'intégrité de données, plus communément appelé code d'authentification de message (MAC [5]), permet de détecter d'éventuelles modifications de données (en transit ou stockées) à l'aide d'une clé secrète. Un tel mécanisme produit, à partir du

5. De l'anglais « Message Authentication Code ».

message et de la clé secrète, un motif d'intégrité difficile à calculer sans connaissance de la clé secrète.

La sécurité des MAC est évaluée quant à leur résistance aux attaques par contrefaçon, où l'objectif d'un attaquant est de produire un motif d'intégrité valide pour un nouveau message sans la connaissance de la clé secrète. Dans ce contexte, on caractérise une attaque contre un MAC avec une quantité supplémentaire, $\mathtt{N_{forge}}$, représentant la taille des requêtes liées aux tentatives de contrefaçons. [6] La définition de $\mathtt{N_{bloc}}$ est légèrement modifiée pour correspondre au nombre total de blocs des requêtes fournissant des couples de message et d'empreinte valides à l'attaquant.

Règle

RègleMAC

1. Si un mécanisme d'intégrité est basé sur une primitive sous-jacente, celle-ci doit être conforme au référentiel.
2. Il ne doit pas exister d'attaque classique sur le mécanisme d'intégrité nécessitant un nombre d'opérations $\mathtt{N_{op}}$ inférieur à 2^{128} et une taille de requêtes $\mathtt{N_{bloc}} + \mathtt{N_{forge}}$ inférieure à 2^{64}.
3. La taille du motif d'intégrité doit être d'au moins 96 bits si la sécurité du mécanisme d'intégrité est indépendante de la taille des messages, et d'au moins 128 bits dans le cas contraire.

Reco

RecoMAC

1. Si un mécanisme d'intégrité est basé sur une primitive sous-jacente, il est recommandé que celle-ci soit conforme aux recommandations afférentes.
2. Il est recommandé d'employer un mécanisme d'intégrité disposant d'une preuve de sécurité dans un modèle d'adversaire pertinent.
3. Il est recommandé d'employer un mécanisme d'intégrité produisant des motifs d'intégrité d'au moins 128 bits.

RègPQ

RèglePQMAC

Lorsqu'une sécurité post-quantique est visée et qu'un mécanisme d'intégrité est basé sur une primitive sous-jacente, celle-ci doit être conforme aux règles post-quantiques afférentes.

RecoPQ

RecoPQMAC

Lorsqu'une sécurité post-quantique est visée et qu'un mécanisme d'intégrité est basé sur une primitive sous-jacente, il est recommandé que celle-ci soit conforme aux recommandations post-quantiques afférentes.

6. Les requêtes comptabilisées dans $\mathtt{N_{forge}}$ ne donnent pas d'information à l'attaquant, outre le fait que l'attaque ait réussi ou non.

Information

- Contrairement au mode opératoire de chiffrement CBC, le mécanisme d'intégrité CBC-MAC fixe l'IV à zéro. Il est possible de construire des contrefaçons si CBC-MAC est utilisé comme le mode de chiffrement CBC, avec un IV variable.
- Il est important de prendre en considération les capacités d'un éventuel attaquant à observer des motifs d'intégrité mais également à en obtenir pour des messages de son choix.
- De nombreux mécanismes d'intégrité, tels que CMAC ou GMAC, ne sont sûrs que si l'on traite au plus de l'ordre de $2^{n/2}$ blocs de messages clairs, où n désigne la taille en bits du bloc.
- Dans certains mécanismes d'intégrité, tels que GMAC, la probabilité de forger l'empreinte d'un message augmente avec sa longueur. La règle **RègleMAC.3** requiert une empreinte d'au moins 128 bits pour ces mécanismes, afin de conserver une marge de sécurité convenable pour de longs messages.
- Un mécanisme d'intégrité vient souvent en complément d'un mécanisme assurant la confidentialité. La composition de deux mécanismes cryptographiques n'est jamais simple et doit être réalisée avec soin. À titre d'exemple, il est possible en associant un très bon chiffrement avec un très bon mécanisme d'intégrité de perdre la propriété de confidentialité.
- Le mécanisme d'intégrité HMAC utilise une fonction de hachage cryptographique pour laquelle une résistance en collision n'est pas requise. Lorsqu'une résistance post-quantique est visée, il est donc suffisant que cette fonction de hachage produise des empreintes de 256 bits pour satisfaire à la fois la règle **RèglePQMAC** et la recommandation **RecoPQMAC**.

Mécanismes conformes

- Le mécanisme d'intégrité CMAC utilisant l'AES-128 comme primitive de chiffrement par bloc sous-jacente est conforme aux règles **RègleMAC** et **RèglePQMAC** ainsi qu'à la recommandation **RecoMAC**, qu'il utilise des empreintes de 128 bits ou des empreintes tronquées à 96 bits. En revanche, ce mécanisme n'est pas conforme à la recommandation **RecoPQMAC** car l'AES-128 n'est pas conforme à la recommandation **RecoPQPrimChiffBloc**.
- Le mécanisme d'intégrité GMAC utilisant l'AES-128 comme primitive de chiffrement par bloc sous-jacente est conforme aux règles **RègleMAC** et **RèglePQMAC** ainsi qu'à la recommandation **RecoMAC** s'il utilise des empreintes de taille 128 bits. En revanche, ce mécanisme n'est pas conforme à la recommandation **RecoPQMAC** car l'AES-128 n'est pas conforme à la recommandation **RecoPQPrimChiffBloc**.
- Le mode d'intégrité HMAC utilisant SHA-256 comme fonction de hachage

sous-jacente est conforme aux règles et recommandations précédentes.

Mécanismes non conformes

- Le mécanisme d'intégrité CBC-MAC utilisé avec des messages de taille variable n'est pas conforme à la règle **RègleMAC.2** car il est vulnérable à des attaques par extension.
- Le mécanisme d'intégrité CBC-MAC « retail » [7] utilisé avec DES ou Triple-DES comme primitive de chiffrement par bloc sous-jacente n'est pas conforme à la règle **RègleMAC.1**.

2.1.4 Fonctions de hachage et fonctions à sortie extensible

Une fonction de hachage cryptographique est une fonction publique sans clé qui à un message de taille quelconque associe une sortie de taille fixe appelée empreinte. Une telle fonction doit satisfaire plusieurs propriétés de sécurité telles que la résistance à la recherche de préimages ou de collisions (voir annexe A.1.4). Des attaques dites génériques permettent de résoudre ces problèmes avec des complexités bien connues (voir annexe A.1.4), par exemple la recherche de collision basée sur le paradoxe des anniversaires. Les fonctions de hachage cryptographiques sont conçues pour qu'il n'existe aucune attaque plus efficace que l'attaque générique pour chacune de ces propriétés. Afin de contrer les attaques classiques ou quantiques fondées sur le paradoxe des anniversaires, une empreinte doit être plus longue qu'une clé symétrique pour atteindre le même niveau de robustesse.

En pratique, les fonctions de hachage sont construites de manière itérative autour de fonctions plus élémentaires, par exemple de fonctions de compression dans le cas de la famille SHA-2 ou de permutations publiques dans le cas de la famille SHA-3. On qualifie de *partielle* une attaque sur un de ces composants internes. L'existence d'une attaque partielle n'implique pas nécessairement de faiblesse de la fonction de hachage en elle-même, mais trahit des défauts de conception majeurs.

Règle

RègleHachage

1. La taille des empreintes produites par une fonction de hachage cryptographique doit être d'au moins 256 bits.
2. La meilleure attaque classique connue permettant de trouver des collisions doit nécessiter de l'ordre de $2^{h/2}$ calculs d'empreinte, où h désigne la taille en bits des empreintes.
3. La meilleure attaque classique connue permettant de trouver une préimage doit nécessiter de l'ordre de 2^{h} calculs d'empreinte, où h désigne la taille en bits des empreintes.

7. Voir annexe A.1.2.

Reco

RecoHachage

Il est déconseillé d'employer une fonction de hachage cryptographique pour laquelle une attaque partielle est connue.

RecoPQ

RecoPQHachage

Lorsqu'une sécurité post-quantique est visée, il est recommandé d'utiliser une fonction de hachage cryptographique dont les empreintes ont une taille d'au moins 384 bits.

Information

- Une taille d'empreinte d'au moins 256 bits garantit qu'il n'existe pas d'attaque générique en collision de complexité significativement inférieure à 2^{128} opérations.
- Une attaque en collision de complexité pratique contre une fonction de hachage cryptographique est particulièrement critique. En effet, des collisions peuvent être calculées une fois puis utilisées à plusieurs reprises par divers acteurs à des fins malveillantes. Dans le cas de SHA-1, une collision a été calculée en pratique en environ 2^{63} calculs d'empreinte et publiée en 2017 [33].
- Selon le type de mémoire quantique utilisé, les meilleures attaques quantiques de recherche d'une collision contre une fonction de hachage cryptographique produisant des empreintes de h bits ont des complexités de $2^{h/3}$ et $2^{2h/5}$ opérations. Pour une valeur de $h = 256$ bits, cela engendre des attaques demandant environ 2^{85} opérations quantiques dans un modèle de calcul extrêmement exigeant, ou 2^{102} opérations quantiques dans un modèle moins contraignant. Indépendamment du modèle, l'accélération quantique de la recherche de collisions est fortement limitée par la profondeur du circuit quantique disponible. Ainsi, l'emploi de fonctions de hachage avec 256 bits de sortie peut être présumé suffisant pour que les attaques quantiques de recherche de collisions soient inopérantes à assez long terme. Toutefois, la recommandation **RecoPQHachage** pourrait devenir une règle post-quantique en cas d'évolutions majeures des technologies de calcul quantique ou de l'algorithmique quantique.
- Pour une fonction de hachage cryptographique utilisée à l'intérieur de mécanismes de signature post-quantique fondés sur le hachage reconnus par la communauté académique et pour laquelle une résistance en collision n'est pas requise, la taille minimale des empreintes produites est de 128 bits pour résister aux attaques classiques. Lorsqu'une sécurité post-quantique est visée, il est recommandé que les empreintes aient une taille d'au moins 192 bits.

Mécanismes conformes

- Les fonctions de hachage SHA2-256 et SHA3-256 sont conformes aux règles **RègleHachage.1**, **RègleHachage.2**, **RègleHachage.3** et à la recommandation **RecoHachage**, mais ne sont pas conformes à la recommandation **RecoPQHachage**.
- La fonction de hachage SHA3-384 est conforme aux règles et recommandations précédentes.

Mécanismes non conformes

- La fonction de hachage SHA-1 n'est pas conforme au référentiel car elle ne respecte pas les règles **RègleHachage.1** et **RègleHachage.2**.
- La fonction de hachage SHA3-224 n'est pas conforme au référentiel car elle ne respecte pas la règle **RègleHachage.1**.
- La fonction de hachage définie par SHAKE-128 avec 256 bits de sortie n'est pas conforme au référentiel car elle ne respecte pas la règle **RègleHachage.3**. En effet, SHAKE-128 est basée sur une fonction éponge avec 256 bits de capacité, sur laquelle une attaque en préimage en 2^{128} opérations existe lorsque la taille de la sortie n'est pas trop grande.

Fonctions à sortie extensible. On appelle fonction à sortie extensible ou XOF[8] une variante d'une fonction de hachage capable de renvoyer une empreinte de longueur arbitraire. Une XOF prend en entrée un message de taille quelconque ainsi qu'une taille d'empreinte h et renvoie une empreinte de h bits. Deux appels à une XOF pour un même message et des tailles d'empreinte différentes renvoient deux empreintes dont l'une est un préfixe de l'autre.

Une telle fonction est généralement dérivée d'une fonction de hachage au moyen d'une construction itérative réutilisant un composant interne de la fonction de hachage.

Règle

RègleXOF

1. Pour des empreintes de h bits, la meilleure attaque classique connue permettant de trouver une collision doit nécessiter au minimum de l'ordre de $\min(2^{h/2}, 2^{128})$ calculs d'empreinte.
2. Pour des empreintes de h bits, la meilleure attaque classique connue permettant de trouver une préimage doit nécessiter au minimum de l'ordre de $\min(2^{h}, 2^{128})$ calculs d'empreinte.

RègPQ

RèglePQXOF

1. Lorsqu'une sécurité post-quantique est visée et pour des empreintes de h bits, il ne doit exister aucune attaque quantique permettant de

8. De l'anglais « eXtendable Output Function ».

trouver une collision nécessitant un nombre d'opérations $\mathtt{N_{op}}$ inférieur à $\min(2^{h/3}, 2^{80})$ calculs d'empreinte et une profondeur $\mathtt{N_{prof}}$ inférieure à 2^{48}.

2. Lorsqu'une sécurité post-quantique est visée et pour des empreintes de h bits, il ne doit exister aucune attaque quantique permettant de trouver une préimage nécessitant un nombre d'opérations $\mathtt{N_{op}}$ inférieur à $\min(2^{h/2}, 2^{80})$ calculs d'empreinte et une profondeur $\mathtt{N_{prof}}$ inférieure à 2^{48}.

Reco

RecoXOF

1. Pour des empreintes de h bits, il est recommandé que la meilleure attaque classique connue permettant de trouver une préimage nécessite au minimum de l'ordre de $\min(2^{h}, 2^{256})$ calculs d'empreinte.
2. Il est déconseillé d'employer une XOF pour laquelle une attaque partielle est connue.

RecoPQ

RecoPQXOF

1. Lorsqu'une sécurité post-quantique est visée et pour des empreintes de h bits, il est recommandé d'employer une XOF contre laquelle il n'existe aucune attaque quantique permettant de trouver une collision nécessitant un nombre d'opérations $\mathtt{N_{op}}$ inférieur à $\min(2^{h/3}, 2^{128})$ calculs d'empreinte et une profondeur $\mathtt{N_{prof}}$ inférieure à 2^{64}.
2. Lorsqu'une sécurité post-quantique est visée et pour des empreintes de h bits, il est recommandé d'employer une XOF contre laquelle il n'existe aucune attaque quantique permettant de trouver une préimage nécessitant un nombre d'opérations $\mathtt{N_{op}}$ inférieur à $\min(2^{h/2}, 2^{128})$ calculs d'empreinte et une profondeur $\mathtt{N_{prof}}$ inférieure à 2^{64}.

Information

- Les règles de sécurité des XOF et des fonctions de hachage ne coïncident pas. En effet, pour être conforme aux règles du référentiel, une fonction de hachage cryptographique doit avoir une résistance en préimage deux fois supérieure à la résistance en collision, alors qu'une XOF doit satisfaire le même niveau de sécurité pour la collision et le préimage. Cela est dû aux principes de conception de certaines familles de XOF comme SHAKE.
- Il est possible d'utiliser une XOF comme une fonction de hachage en fixant la taille de ses sorties. Il convient toutefois de faire attention aux points suivants. D'une part, de telles fonctions de hachage peuvent produire des sorties reliées, lorsque seule la taille d'empreinte varie. D'autre part, une XOF conforme au référentiel peut produire une fonction de hachage non conforme, comme détaillé dans la remarque précédente. Par exemple, SHAKE-128 avec 256 bits de sortie ne constitue pas une fonction de hachage conforme au référentiel.

Mécanismes conformes

- La fonction à sortie extensible SHAKE-256 est conforme aux règles et recommandations précédentes.
- La fonction à sortie extensible SHAKE-128 est conforme aux règles **RègleXOF.1**, **RègleXOF.2**, **RèglePQXOF.1**, **RèglePQXOF.2** et à la recommandation **RecoXOF.2** mais n'est pas conforme aux recommandations **RecoXOF.1**, **RecoPQXOF.1** et **RecoPQXOF.2**.

2.2 Cryptographie asymétrique

Les mécanismes de cryptographie asymétrique reposent tous sur l'utilisation d'une bi-clé, c'est-à-dire d'une paire (clé publique, clé privée) de clés calculatoirement reliées entre elles. La sécurité de ces mécanismes repose sur des problèmes mathématiques difficiles ou la sécurité de mécanismes symétriques. Certains problèmes, dits *classiques*, sont issus de la théorie des nombres : c'est le cas par exemple du problème de la factorisation ou de celui du logarithme discret. Ces deux problèmes sont vulnérables à un attaquant disposant d'un ordinateur quantique de taille suffisante. D'autres problèmes, dits *post-quantiques*, ont été proposés dans le milieu académique afin de résister à un attaquant disposant d'un ordinateur quantique. La sécurité de ces problèmes a été moins étudiée que celle des problèmes classiques. À l'heure actuelle, parmi les mécanismes asymétriques, la meilleure sécurité à long terme est donc obtenue par l'utilisation de mécanismes asymétriques dont la sécurité repose sur la difficulté à la fois d'un problème classique et d'un problème post-quantique, dans le sens où le mécanisme est sûr tant que l'un des deux problèmes est difficile. On parle alors de principe de non-régression de sécurité.

Parmi ces familles de problèmes dits *post-quantiques* figurent notamment des problèmes de géométrie des réseaux euclidiens, des problèmes de décodage de certains codes correcteurs d'erreur, la résolution de certaines équations polynomiales multivariées, et des problèmes de recherche de chemins dans certains graphes d'isogénies de courbes elliptiques.

Règle

RègleSécuAsym

La sécurité des mécanismes asymétriques contre un adversaire classique doit reposer sur au moins une des hypothèses suivantes :

- la difficulté d'un problème mathématique largement éprouvé et reconnu par la communauté académique,
- la sécurité d'un mécanisme symétrique conforme au référentiel.

RègPQ

RèglePQSécuAsym

Lorsqu'une sécurité post-quantique est visée, la sécurité des mécanismes asymétriques doit reposer sur au moins une des hypothèses suivantes :

- la difficulté d'un problème mathématique présumé résistant au calcul quantique,
- la sécurité d'un mécanisme symétrique conforme au référentiel.

On appellera *mécanisme post-quantique* tout mécanisme asymétrique satisfaisant la règle **RèglePQSécuAsym**.

Mécanismes conformes

- Les problèmes classiques de la factorisation et du logarithme discret (dans certains groupes multiplicatifs ou dans le groupe des points de certaines courbes elliptiques) ont été largement éprouvés. À condition de satisfaire notamment les conditions des sections 2.2.1.1 et 2.2.1.2, les mécanismes dont la sécurité repose sur ces problèmes sont conformes à la règle **RègleSécuAsym**.
- Les problèmes de réseaux euclidiens de l'apprentissage avec erreurs et de la solution entière courte définis dans la section 2.2.1.3 sont présumés résistants au calcul quantique. À condition de satisfaire notamment les conditions de la section 2.2.1.3, les mécanismes dont la sécurité repose sur ces problèmes sont conformes à la règle **RèglePQSécuAsym**.

Mécanismes non conformes

- Les problèmes classiques de la factorisation et du logarithme discret (dans certains groupes multiplicatifs ou dans le groupe des points de certaines courbes elliptiques) sont résolus en temps polynomial par un adversaire quantique utilisant l'algorithme de Shor [32]. Les mécanismes dont la sécurité repose sur ces problèmes uniquement ne respectent pas la règle **RèglePQSécuAsym**.
- Les mécanismes dont la sécurité repose uniquement sur les problèmes de réseaux euclidiens de l'apprentissage avec erreurs et de la solution entière courte définis dans la section 2.2.1.3 ne sont pas conformes à la règle **RègleSécuAsym**.

Information

- La plupart des problèmes post-quantiques ont attiré l'attention de la communauté académique de façon relativement récente, et il est donc plus difficile de disposer d'un recul important sur leur dimensionnement, leur conversion en primitives cryptographiques, ou sur les écueils à éviter lors de l'implémentation de celles-ci.
- Un produit visant une sécurité post-quantique doit satisfaire les règles **RègleSécuAsym** et **RèglePQSécuAsym**, afin d'assurer une non-régression de la sécurité. Pourtant, aucun problème mathématique ne peut être à ce jour considéré comme satisfaisant les deux règles simultanément. Il est cependant possible d'utiliser certains *modes d'hybridation*, qui combinent un mécanisme post-quantique avec un mécanisme classique (symétrique ou asymétrique conforme à la règle **RègleSécuAsym**) et produisent un mécanisme asymétrique satisfaisant les règles **RègleSécuAsym** et **RèglePQSécuAsym**. Le coût de la partie classique d'un mode d'hybridation (en particulier en termes de volume de communications) sera souvent marginal, alors que cette partie joue un rôle central dans l'assurance de sécurité

classique.

Modes d'hybridation conformes

Dans le cas de l'encapsulation de clés, la manière générale de satisfaire les règles **RègleSécuAsym** et **RèglePQSécuAsym** consiste à hybrider deux des trois catégories de mécanismes cryptographiques (symétrique, asymétrique classique, asymétrique post-quantique) :

- La méthode la plus souhaitable consiste à hybrider un mécanisme d'encapsulation de clé asymétrique classique avec un mécanisme d'encapsulation de clé asymétrique post-quantique. Les clés issues des décapsulations du mécanisme classique et du mécanisme post-quantique peuvent être combinées – éventuellement avec des données supplémentaires comme les chiffrés et les clés publiques de ces deux mécanismes – à l'aide d'une fonction de dérivation de clés bien choisie.

- Il est également possible d'hybrider un mécanisme d'encapsulation de clé asymétrique classique avec un mécanisme de chiffrement symétrique. La présence d'un secret partagé entre deux entités peut ajouter une couche de protection contre un adversaire quantique ; elle change en outre la nature du mécanisme, qui acquiert certaines caractéristiques d'un mécanisme symétrique. L'architecture de clés d'un tel mécanisme doit en particulier tenir compte des règles et recommandations du document [2]. Notamment, chaque secret symétrique pré-partagé doit ne l'être qu'entre deux entités. Ce mode d'hybridation peut cependant entraîner la perte de certaines propriétés de sécurité désirables comme la non-répudiation ou la confidentialité persistante (*Perfect Forward Secrecy*).

- Il est enfin envisageable d'associer un mécanisme symétrique dépendant d'un secret partagé entre deux entités avec un mécanisme asymétrique post-quantique. L'architecture de clés d'un tel mécanisme doit tenir compte des règles et recommandations citées ci-dessus. Ce type d'architecture sera cependant difficilement compatible avec la recommandation **RecoPQConfidentialitéPersistante** ci-dessous.

Les règles **RègleSécuAsym** et **RèglePQSécuAsym** peuvent par exemple être satisfaites des manières suivantes dans le cas des signatures :

- Il est possible d'hybrider par concaténation des mécanismes de signature classique et post-quantique. Une signature du mécanisme hybride est obtenue par concaténation d'une signature du mécanisme classique et d'une signature du mécanisme post-quantique. On considère la signature hybride valide si, et seulement si, la signature classique et la signature post-quantique sont toutes les deux valides.

- Certains mécanismes de signature post-quantiques basés sur une ou plusieurs primitives symétriques, par exemple une fonction de hachage, disposent d'une preuve de sécurité par réduction à la sécurité de la primitive symétrique sous-jacente. À condition d'être employés de manière cohérente avec ces preuves de sécurité, ces mécanismes sont donc conformes

aux règles **RègleSécuAsym** et **RèglePQSécuAsym**. Pour ces mécanismes, le recours à l'hybridation demeure conforme, mais optionnel.

2.2.1 Problèmes mathématiques asymétriques

La section qui suit est dédiée à la description de problèmes asymétriques répandus en cryptographie et à l'énumération de règles et recommandations associées.

2.2.1.1 Factorisation

On appelle « module RSA », ou plus simplement « module », un nombre entier obtenu de manière secrète par multiplication de deux nombres premiers, généralement de taille comparable. Le problème de la factorisation d'un module RSA consiste à retrouver la décomposition en facteurs premiers d'un module RSA donné. Dans la suite, on désignera ce problème par « problème de la factorisation ».

Le problème de la factorisation est principalement utilisé par le mécanisme RSA. Les calculs de chiffrement et de déchiffrement RSA font intervenir deux autres données que le module, appelées « exposant public » et « exposant secret ».

Règle

RègleFactorisation

1. La taille minimale du module est de 2048 bits, pour une utilisation ne devant pas dépasser la fin de l'année 2030.
2. Pour une utilisation à partir de l'année 2031, la taille minimale du module est de 3072 bits.
3. L'exposant secret doit être environ de même taille que le module.
4. Pour les applications de chiffrement, l'exposant public doit être strictement supérieur à $2^{16} = 65536$ et avoir une taille maximale de 256 bits.
5. Les nombres premiers constitutifs du module doivent être choisis aléatoirement uniformément, éventuellement sous des conditions supplémentaires satisfaites par un grand sous-ensemble de ces nombres.

Reco

RecoFactorisation

1. Il est recommandé d'employer des modules d'au moins 3072 bits, même pour une utilisation ne devant pas dépasser 2030.
2. Il est recommandé, pour toute application, d'employer des exposants publics strictement supérieurs à $2^{16} = 65536$ et de taille maximale 256 bits.
3. Il est recommandé que les nombres premiers constitutifs du module soient de même taille.

Information

- Le record public de factorisation (remontant à 2020) est de 829 bits. La taille de module de 1024 bits (encore occasionnellement rencontrée) est donc en deçà d'une marge de sécurité raisonnable. En revanche, compte tenu de la complexité des algorithmes connus pour factoriser de grands entiers, une taille de 3000 bits (en pratique, 3072 bits) donnerait une sécurité de même ordre de grandeur que celle d'une clé symétrique de 128 bits, et donc suffisante. Par conséquent, l'emploi de modules de 1024 bits est considéré comme une prise de risque incompatible avec des critères de sécurité raisonnables.
- L'annexe B.2 fournit des informations complémentaires sur les analyses liées au problème de la factorisation.
- L'emploi d'exposants publics très petits, tels que l'exposant 3, est également à proscrire dans le cas du chiffrement à cause des attaques existantes [13]. Plus généralement, pour toute application, l'emploi de tels exposants est déconseillé pour des raisons de sécurité.
- L'emploi d'exposants secrets particuliers (comme des exposants secrets petits par exemple) afin d'améliorer les performances est à proscrire. En revanche, l'emploi d'exposants publics particuliers, dans les limites de la remarque précédente, est possible : en pratique, l'exposant public généralement utilisé est $e = 65537$.
- L'emploi de nombres premiers p et q trop proches ou de tailles trop différentes peut compromettre la sécurité du système. Il faut également éviter des valeurs possédant des propriétés particulières comme l'absence d'un grand facteur premier dans la décomposition de $p \pm 1$ ou $q \pm 1$. Pour éviter ces problèmes, il faut choisir p et q aléatoirement uniformément parmi les nombres premiers de taille égale à la moitié de la taille du module, éventuellement sous la condition supplémentaire que $p \pm 1$ et $q \pm 1$ aient de grands diviseurs premiers.
- Le problème de la factorisation est résolu en temps polynomial par un adversaire quantique utilisant l'algorithme de Shor [32]. Il ne fournit aucune sécurité post-quantique.

2.2.1.2 Logarithme discret

Le problème dit « du logarithme discret » est fondé sur la difficulté d'inverser l'opération d'exponentiation dans un groupe. Ce problème peut être instancié dans différentes structures et nous donnons ici des règles et recommandations sur les choix de paramètres à utiliser pour trois d'entre elles :

- les corps finis $\mathrm{GF}(p)$ à p éléments où p est un nombre premier ;
- les groupes des points de courbes elliptiques définies sur $\mathrm{GF}(p)$ où p est un nombre premier ;
- les groupes des points de courbes elliptiques définies sur $\mathrm{GF}(2^n)$.

Bien qu'il soit possible d'instancier ce problème dans d'autres structures, nombre d'entre elles sont à proscrire : tel est notamment le cas des corps finis de petite caractéristique, et en particulier de $\mathrm{GF}(2^n)$. Certaines de ces autres structures ne présentent toutefois pas de faiblesses connues, mais leur sécurité doit être étudiée au cas par cas et leur emploi doit être soumis à l'avis de l'ANSSI.

Logarithme discret dans $\mathrm{GF}(p)$**.** Le problème dit « du logarithme discret dans $\mathrm{GF}(p)$ » est fondé sur des calculs effectués dans le corps fini à p éléments, où p est un nombre premier également appelé « module ».

Règle

RègleLogDiscretGFp

1. La taille minimale de modules premiers est de 2048 bits pour une utilisation ne devant pas dépasser la fin de l'année 2030.
2. Pour une utilisation à partir de l'année 2031, la taille minimale de modules premiers est de 3072 bits.
3. On emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 250 bits.

Reco

RecoLogDiscretGFp

1. Il est recommandé d'employer des modules premiers d'au moins 3072 bits, même pour une utilisation ne devant pas dépasser 2030.
2. Il est recommandé d'employer des sous-groupes dont l'ordre est premier.

Information

- Le problème du logarithme discret dans $\mathrm{GF}(p)$ semble avoir une complexité comparable à celle de la factorisation. Des méthodes similaires s'appliquent à la résolution des deux problèmes et il est raisonnable de penser qu'une avancée majeure dans la résolution du problème de la factorisation s'accompagnerait d'une avancée semblable dans celui du logarithme discret.

  Il est par conséquent naturel d'appliquer des règles identiques pour les deux problèmes.

- Le problème du logarithme discret semble cependant légèrement plus difficile en pratique, certaines phases de calcul étant plus délicates que pour la factorisation, les records de calcul (voir annexe B) mettent en évidence un décalage compris entre 100 et 200 bits mais on peut se demander si une telle différence ne provient pas en partie du plus grand prestige promis à un record en matière de factorisation.

- Si l'ordre d'un sous-groupe n'est pas premier mais possède un petit facteur

premier (dans le pire des cas, 2), le problème Diffie–Hellman décisionnel dans ce sous-groupe peut être résolu avec une probabilité non négligeable. Ceci peut être très problématique, notamment dans des mécanismes de chiffrement de type ElGamal.

- Le problème du logarithme discret dans $\mathrm{GF}(p)$ est résolu en temps polynomial par un adversaire quantique utilisant l'algorithme de Shor [32]. Il ne fournit aucune sécurité post-quantique.

Logarithme discret sur les courbes elliptiques définies sur $\mathrm{GF}(p)$**.** Il est également possible de définir un problème de logarithme discret dans des structures plus complexes pour lesquelles aucun algorithme classique plus efficace que les méthodes génériques de calcul de logarithme discret n'est connu. C'est en particulier aujourd'hui le cas des courbes elliptiques qui sont définies sur un corps de base pouvant être, en pratique, premier ($\mathrm{GF}(p)$) ou binaire ($\mathrm{GF}(2^n)$).

Règle

RègleCourbeElliptiqueGFp

1. On emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 250 bits.
2. En cas d'utilisation de courbes particulières faisant reposer la sécurité sur un problème mathématique plus facile que le problème générique de calcul de logarithme discret sur courbe elliptique définie sur $\mathrm{GF}(p)$, ce problème devra vérifier les règles correspondantes.

Reco

RecoCourbeElliptiqueGFp

Il est recommandé d'employer des sous-groupes dont l'ordre est premier.

Information

- Le problème du logarithme discret sur courbe elliptique permet d'obtenir, pour des tailles de paramètres réduites, une sécurité classique comparable à celle exigée pour des primitives symétriques.
- En cas d'utilisation de courbes généralement qualifiées de « particulières », la sécurité peut être sérieusement dégradée. Le problème mathématique sous-jacent peut notamment être ramené à un problème de calcul de logarithme discret dans un corps fini et non plus sur une courbe elliptique. Dans ce cas, les règles relatives à ce problème s'appliquent bien évidemment. Par exemple, dans le cadre d'un mécanisme utilisant les couplages – pour lequel l'utilisation de courbes « particulières » est nécessaire – le choix desdites courbes devra être fait avec soin. En particulier, les préconisations de la section 2.2.1.2 concernant la difficulté du logarithme discret devront être prises en compte pour le choix du groupe multiplicatif dans lequel le couplage prend ses valeurs.

- La raison de recommander un sous-groupe d'ordre premier est encore une fois que si l'ordre d'un sous-groupe n'est pas premier mais un petit multiple (dans le pire cas, le produit par 2) d'un grand premier, le problème Diffie–Hellman décisionnel dans ce sous-groupe peut être résolu avec une probabilité non négligeable.
- Le problème du logarithme discret sur les courbes elliptiques définies sur $\mathrm{GF}(p)$ est résolu en temps polynomial par un adversaire quantique utilisant l'algorithme de Shor [32]. Il ne fournit aucune sécurité post-quantique.

Mécanismes conformes

- La courbe FRP256v1 – définie dans le journal officiel n° 241 du 16/10/2011 et dont les paramètres, validés par l'ANSSI, peuvent librement être intégrés dans tous les produits de sécurité – est conforme au référentiel.
- Les courbes P-256, P-384 et P-521 définies dans le FIPS 186-5 de 2023, ainsi que des courbes `brainpoolP256r1`, `brainpoolP384r1` et `brainpoolP512r1` sont conformes au référentiel.

Logarithme discret sur les courbes elliptiques définies sur $\mathrm{GF}(2^n)$**.** Les règles énoncées précédemment se déclinent de manière similaire, en faisant attention au choix de n.

Règle

RègleCourbeElliptiqueGF2n

1. L'ordre du sous-groupe doit être multiple d'un nombre premier d'au moins 250 bits.
2. Le paramètre n doit être un nombre premier.
3. En cas d'utilisation de courbes particulières faisant reposer la sécurité sur un problème mathématique plus facile que le problème générique de calcul de logarithme discret sur courbe elliptique définie sur $\mathrm{GF}(2^n)$, ce problème devra vérifier les règles correspondantes.

Reco

RecoCourbeElliptiqueGF2n

Il est recommandé d'employer des sous-groupes dont l'ordre est premier.

Information

- L'emploi de n composé réduit considérablement la difficulté du calcul de logarithme discret et affaiblit donc le mécanisme correspondant.
- Les courbes elliptiques définies sur $\mathrm{GF}(p)$ ne sont pas différenciées de celles définies sur $\mathrm{GF}(2^n)$.
- Le problème du logarithme discret sur les courbes elliptiques définies

sur $\mathrm{GF}(2^n)$ est résolu en temps polynomial par un adversaire quantique utilisant l'algorithme de Shor [32]. Il ne fournit aucune sécurité post-quantique.

Mécanisme conforme

L'emploi des courbes B-283, B-409 et B-571 définies dans le FIPS 186-5 de 2023 est conforme au référentiel.

2.2.1.3 Réseaux euclidiens : apprentissage avec erreurs, solution entière courte

Les mécanismes fondés sur les réseaux euclidiens sont généralement construits à partir des problèmes échantillonnables de « l'apprentissage avec erreurs » (LWE, « Learning with Errors ») et de la « solution entière courte » (SIS, « Short Integer Solution »), ou de leurs variantes respectives, MLWE et MSIS [27].

On peut, pour certains choix de paramètres, faire la preuve que résoudre l'un de ces problèmes implique de résoudre des problèmes géométriques tels que le problème du plus court vecteur (SVP, « *Shortest Vector Problem* ») ou celui du plus proche vecteur (CVP, « *Closest Vector Problem* »). Ces problèmes sont présumés difficiles à résoudre, même avec l'aide d'un ordinateur quantique [30]. En pratique, les attaques de type « *block Korkine-Zolotarev* » (BKZ) [31] sont les plus efficaces pour les paramètres les plus usuels. Leur complexité est dominée par la résolution de SVP dans une dimension réduite, appelée taille de bloc.

RègPQ

RèglePQRéseauEuclidien

1. Les paramètres de (M)LWE doivent être tels que la complexité de la meilleure attaque quantique connue soit au moins aussi élevée que celle de la résolution de SVP dans un réseau générique de dimension 400.
2. Les paramètres de (M)SIS doivent être tels que la complexité de la meilleure attaque quantique connue soit au moins aussi élevée que celle de la résolution de SVP dans un réseau générique de dimension 400.

RecoPQ

RecoPQRéseauEuclidien

1. Il est recommandé que les paramètres de (M)LWE soient tels que la complexité de la meilleure attaque quantique connue soit au moins aussi élevée que celle de la résolution de SVP dans un réseau générique de dimension 600.
2. Il est recommandé que les paramètres de (M)SIS soient tels que la complexité de la meilleure attaque quantique connue soit au moins aussi élevée que celle de la résolution de SVP dans un réseau générique de dimension 600.

Information

- Les problèmes de réseaux euclidiens ont un espace de paramètres bien plus grand que les problèmes fondés sur la théorie des nombres. En effet, pour paramétrer le problème LWE, il faut choisir : les deux dimensions de la matrice définissant le système, le module, la distribution du secret ainsi que la distribution du bruit ajouté aux équations. L'interaction entre ces paramètres est complexe, et les jeux de paramètres proposés par les standards prennent en général en compte toutes les attaques connues, même celles fonctionnant dans des régimes spécifiques.
- Les algorithmes fondés sur les réseaux euclidiens ont pour but d'apporter une sécurité post-quantique. À ce titre, il est crucial de prendre en compte leur sécurité contre un adversaire quantique, contrairement aux problèmes fondés sur la théorie des nombres qui n'apportent qu'une sécurité pré-quantique.
- Compte tenu des meilleures attaques quantiques connues pour résoudre SVP [12] combinées avec des techniques de réduction de la dimension du problème [18], un problème SVP en dimension 400, respectivement 600, requiert un calcul quantique de complexité en temps estimé à au moins 2^{94}, respectivement 2^{143}. Cependant, cette estimation néglige les facteurs polynomiaux de l'attaque, qui sont difficiles à calculer précisément, et sous-estime donc largement sa complexité.
- La résolution de SVP est une étape répétée à de nombreuses reprises lors de l'exécution de l'algorithme BKZ, et donc au cours de la résolution des problèmes (M)LWE et (M)SIS. Cependant, il reste de nombreux points d'ombre sur les heuristiques utilisées pendant ces résolutions, notamment dans le cas de l'attaque dite "duale" contre LWE [11, 19, 28]. De plus, l'optimisation de l'algorithme BKZ est toujours à l'étude dans la communauté académique [1]. De manière conservatrice, la règle considère que la complexité de ces attaques est équivalente à celle d'une seule résolution de SVP.

2.2.2 Encapsulation de clé et chiffrement asymétrique

Un mécanisme de chiffrement asymétrique est composé de trois algorithmes : un algorithme de génération de bi-clés, un algorithme de chiffrement, prenant en entrée un message et la clé publique et renvoyant un chiffré, et un algorithme de déchiffrement, prenant en entrée un chiffré et la clé privée et renvoyant le message correspondant.

Un mécanisme d'encapsulation de clé (KEM[9]) est également composé de trois algorithmes : un algorithme de génération de bi-clés, un algorithme d'encapsulation, prenant en entrée la clé publique et renvoyant une clé de session et son chiffré, et un algorithme de décapsulation, prenant en entrée la clé privée et un chiffré et renvoyant la clé de session correspondante.

9. « Key Encapsulation Mechanism » en anglais.

Tout mécanisme d'encapsulation de clé ou de chiffrement asymétrique doit *a minima* être sémantiquement sûr (voir annexes A.2.1 et A.2.2).

Ces deux mécanismes s'appuient sur un problème difficile de base. Ce dernier doit donc être en accord avec le niveau de robustesse recherché et respecter les règles correspondantes. De plus, il est possible pour la plupart de ces mécanismes de faire la preuve que la sécurité est équivalente à la difficulté du problème de base, au lieu de simplement relier le mécanisme au problème de manière heuristique.

Il existe différentes modes d'hybridation, parfois entre mécanismes de chiffrement asymétrique et mécanismes d'encapsulation de clé, quitte à transformer implicitement l'un en l'autre via des transformations génériques.

Reco

RecoConfidentialitéAsym

1. Il est recommandé d'employer des mécanismes disposant d'une preuve de sécurité dans un modèle d'adversaire pertinent.
2. En cas d'hybridation entre deux mécanismes, il est recommandé qu'ils disposent tous deux d'une preuve de sécurité dans le même modèle d'adversaire. Il est recommandé que le mode d'hybridation fournisse une sécurité contre le type d'adversaire le plus fort que les deux mécanismes supportent.

Information

- L'existence d'une preuve de sécurité apporte des garanties importantes sur la résistance du mécanisme.
- L'emploi d'un mode d'hybridation cohérent avec les mécanismes sous-jacents permet d'éviter l'emploi du mécanisme par mégarde dans un contexte où des attaques seraient possibles en raison d'un mode d'hybridation dégradant la sécurité.

Mécanismes conformes

- Le mécanisme d'encapsulation de clé ECIES-KEM, tel que défini dans l'ISO18033-2 par exemple, est conforme au référentiel. Pour suivre la règle post-quantique **RèglePQSécuAsym**, il faut hybrider ce mécanisme avec un KEM post-quantique.
- Le mécanisme d'encapsulation de clé ML-KEM-512 défini dans le FIPS 203 est conforme au référentiel à condition de l'hybrider avec un mécanisme classique pour respecter la règle **RègleSécuAsym**. Il est toutefois préférable d'utiliser le niveau de sécurité supérieur, ML-KEM-768.
- Le mécanisme d'encapsulation de clé Frodo-KEM-640 est conforme au référentiel à condition de l'hybrider avec un mécanisme classique pour res-

pecter la règle **RègleSécuAsym**. Il est toutefois préférable d'utiliser le niveau de sécurité supérieur, Frodo-KEM-976.

- Le mécanisme de chiffrement asymétrique RSAES-OAEP défini dans le document PKCS#1 v2.1 est conforme au référentiel à condition de respecter les règles **RègleFactorisation.1**, **RègleFactorisation.2**, **RègleFactorisation.3** et **RègleFactorisation.4**. Pour respecter la règle post-quantique **RèglePQ-SécuAsym**, il faut l'hybrider avec un mécanisme post-quantique.

Mécanismes non conformes

- Utilisé sans hybridation et quel que soit le jeu de paramètres utilisé, le mécanisme d'encapsulation de clés ML-KEM ne respecte pas la règle **RègleSécuAsym**.
- Le mécanisme de chiffrement asymétrique RSAES mis en œuvre selon le document PKCS#1 v1.5 n'est pas conforme au référentiel dans un contexte où il est possible d'invoquer un oracle de vérification de padding. En effet, Bleichenbacher a mis en évidence en 1998 une attaque (attaque à chiffrés choisis) exploitant judicieusement un tel oracle pour retrouver le message clair correspondant à un chiffré donné [8].

2.2.3 Signature numérique

Un mécanisme de signature numérique est composé de trois algorithmes : un algorithme de génération de bi-clés, un algorithme de signature, prenant en entrée un message et la clé privée et renvoyant une signature, et un algorithme de vérification, prenant en entrée un message, une signature et la clé publique et acceptant ou non la signature pour ce message.

Tout mécanisme de signature doit *a minima* être robuste face aux attaques en contrefaçon, mais d'autres propriétés additionnelles peuvent être attendues (voir annexe A.2.3).

La sécurité des mécanismes de signature numérique s'appuie sur un problème mathématique difficile ou sur la sécurité d'un mécanisme symétrique. Ce dernier doit donc être en accord avec le niveau de robustesse recherché. De plus, il est possible pour la plupart des mécanismes de signature de faire la preuve que la sécurité est équivalente à celle du problème de base et pas uniquement reliée de manière heuristique.

Reco

RecoSignature

1. Il est recommandé d'employer des mécanismes de signature numérique disposant d'une preuve de sécurité dans un modèle d'adversaire pertinent.
2. En cas d'hybridation entre deux mécanismes de signature numérique, il est recommandé qu'ils disposent tous deux d'une preuve de sécurité dans

le même modèle d'adversaire. Il est recommandé que le mode d'hybridation possède les mêmes propriétés additionnelles que les deux mécanismes.

Information

- L'existence d'une preuve de sécurité apporte des garanties importantes sur la résistance du mécanisme.
- Il est courant de hacher un message avant de le signer. Dans ce cas, le niveau de robustesse de la fonction de hachage utilisée (voir section 2.1.4) doit être en accord avec le niveau de robustesse souhaité pour le mécanisme de signature.
- L'emploi d'un mode d'hybridation cohérent avec les mécanismes sous-jacents permet d'éviter l'apparition d'attaques qui seraient dues à une absence de propriétés additionnelles de ce dernier.

Mécanismes conformes

- Le mécanisme de signature numérique RSA-SSA-PSS[10] défini dans le document PKCS#1 v2.1 est conforme au référentiel à condition de respecter les règles **RègleFactorisation.1**, **RègleFactorisation.2**, **RègleFactorisation.3** et **RègleFactorisation.4**.
- Le mécanisme de signature numérique ECDSA défini dans le FIPS 186-5, ainsi que le mécanisme de signature numérique ECKCDSA, sont conformes au référentiel lorsqu'ils utilisent la courbe FRP256v1 – définie dans le journal officiel nº 241 du 16/10/2011 et dont les paramètres, validés par l'ANSSI, peuvent librement être intégrés dans tous les produits de sécurité – ou lorsqu'ils utilisent l'une des courbes P-256, P-384, P-521, B-283, B-409 et B-571 définies dans le FIPS 186-5. Pour hybrider l'un des mécanismes de signature précédents avec une signature post-quantique conformément à la règle post-quantique **RèglePQSécuAsym**, on peut par exemple signer un message avec ML-DSA tel que défini dans le FIPS 204, puis utiliser ce mécanisme classique pour signer le couple (message, signature).
- Le mécanisme de signature numérique SLH-DSA défini dans le FIPS 205 est conforme aux règles **RègleSécuAsym** et **RèglePQSécuAsym** car il repose sur la sécurité de la fonction de hachage sous-jacente, qui est conforme au référentiel, à la fois en tant que sécurité classique et post-quantique. Il peut donc être utilisé tel quel après 2030.

Mécanismes non conformes

- Le mécanisme de signature numérique RSA-SSA, mis en œuvre selon le document PKCS#1 v1.5 n'est pas conforme au référentiel lorsque l'exposant

10. RSA-SSA-PSS : "RSA Signature Scheme with Appendix – Provably Secure encoding method for digital Signatures".

public e est petit et pour un mauvais choix d'implémentation des vérifications liées au padding. En effet, Bleichenbacher a mis en évidence en 2006 une attaque permettant de créer des signatures valides dans ce cas [9].

- Utilisé sans hybridation et quel que soit le jeu de paramètres utilisé, le mécanisme de signature numérique ML-DSA ne respecte pas la règle **RègleSécuAsym**.

Attention

Certains contextes et protocoles utilisent des signatures de telle manière que la simple résistance à la contrefaçon ne suffit pas à garantir la sécurité du système. Celle-ci peut cependant être garantie si la signature vérifie une ou plusieurs propriétés additionnelles telles que la sécurité forte en contrefaçon, la propriété exclusive, la signature liée au message ou la non-resignabilité. Ces propriété additionnelles sont rappelées en annexe A.2.3.

Information

- L'utilisation du mécanisme de signature numérique ECDSA dans le protocole Bitcoin crée une vulnérabilité permettant à un attaquant d'utiliser deux fois le même jeton, en raison de l'absence de sécurité forte en contrefaçon de la signature [26]. Des contre-mesures spécifiques doivent être implémentées dans le protocole pour corriger cette vulnérabilité.
- Le protocole SSH requiert que les signatures satisfassent la sécurité forte en contrefaçon [7]. L'utilisation du mécanisme de signature numérique ECDSA n'est pas compatible avec ce protocole.
- Une version préliminaire du protocole ACME, utilisé dans Let's Encrypt, un service de fournisseurs de certificats X.509, présentait une vulnérabilité [5] liée à l'utilisation d'une signature (RSA) résistante en contrefaçon, mais ne satisfaisant pas la notion de propriété exclusive. La vulnérabilité a été corrigée en changeant la signature utilisée.
- Le protocole DRKey est un protocole d'établissement de clés. Il s'avère que ce protocole présente une vulnérabilité liée à l'utilisation de signatures ne satisfaisant pas la propriété de non-resignabilité [25], bien que celles-ci soient résistantes aux attaques en contrefaçon.

2.2.4 Authentification d'entités et établissement de clé

Les mécanismes interactifs d'authentification d'entités et d'établissement de clé assurent l'authenticité du possesseur d'une clé publique ainsi que la confidentialité des échanges subséquents avec cette même entité. Pour de nombreux mécanismes d'établissement de clé, la propriété cryptographique appelée *confidentialité persistante* (« *Perfect Forward Secrecy* », ou PFS) peut être garantie. Elle assure qu'une compromission de clés ou d'autres données long terme ne remet pas en cause la confidentialité des échanges passés.

Reco

RecoConfidentialitéPersistante

1. Il est recommandé que la compromission à un instant donné d'éventuels secrets de long terme n'affecte pas la confidentialité des sessions passées.
2. Afin de ne pas perdre la propriété de PFS, il est recommandé que tout secret temporaire contribuant à la protection en confidentialité d'un canal sécurisé (secrets éphémères, clé de session, etc.) soit effacé au plus tard à la fin de chaque session de communication.

Information

- On peut définir informellement la propriété de PFS comme l'impossibilité d'obtenir quelque information que ce soit sur une clé de session pour un attaquant capable (1) d'observer et/ou perturber les échanges d'établissement de clé au cours de laquelle cette clé de session a été établie entre deux entités et (2) d'accéder, postérieurement à la période de validité de cette clé de session, à l'ensemble des secrets d'une de ces deux entités. Une condition nécessaire pour assurer cette propriété de PFS est de recourir, pour l'établissement des clés symétriques temporaires, à un mécanisme reposant sur l'emploi par les deux entités de secrets éphémères effacés après utilisation. Les clés symétriques temporaires doivent elles-mêmes être effacées par les deux entités à l'échéance de leur période de validité (fin de session).
- Comme il est rappelé dans l'annexe A.2.4, les mécanismes interactifs d'authentification d'entités et d'établissement de clé reposent en général, au moins partiellement, sur des mécanismes de génération d'aléa, de hachage et de chiffrement ou de signature à clé publique. Les règles et recommandations énoncées dans les sections 2.1.4, 2.2.2, 2.2.3 et 2.3 s'appliquent alors directement. Ces mécanismes peuvent également faire appel à des primitives asymétriques spécifiques, telles que les schémas d'authentification « à divulgation nulle de connaissance » ou l'échange de clés de Diffie-Hellman ; les règles et recommandations énoncées dans la section 2.2.1 s'appliquent alors aux problèmes mathématiques sur lesquels ces primitives reposent. L'évaluation du niveau de robustesse global du mécanisme doit être effectuée avec soin, même si des primitives conformes au référentiel sont employées. Une attention particulière devra notamment être portée à la résistance des mécanismes d'établissement de clé aux attaques par le milieu. L'utilisation conjointe de mécanismes d'établissement de clé et d'authentification d'entité convenablement liés l'un à l'autre peut permettre de se prémunir contre des attaques de ce type.

RecoPQ

RecoPQConfidentialitéPersistante

Lorsqu'une sécurité post-quantique est visée, afin de conserver la propriété de PFS en présence d'adversaires quantiques, il est recommandé que la compromission à un instant donné d'éventuels secrets de long terme par un adversaire doté de moyens de calcul quantique n'affecte pas la confidentialité des sessions passées.

Mécanisme non conforme

L'hybridation de l'échange de clé Diffie-Hellman avec un mécanisme symétrique au moyen d'une clé pré-partagée ne respecte pas la recommandation **RecoPQConfidentialitéPersistante**. En cas de compromission de la clé pré-partagée, un adversaire disposant d'un enregistrement des communications passées pourra les déchiffrer immédiatement s'il dispose de capacités quantiques et ultérieurement – au moment où il disposera de capacités quantiques – dans le cas contraire.

Cas particulier des secrets de faible entropie. Certains mécanismes d'authentification reposent sur des sources de faible entropie, telles que des données biométriques ou des mots de passe [4]. À titre d'exemple, des mots de passe de 8 caractères alphanumériques (chiffres et lettres majuscules ou minuscules) ont une entropie au mieux de 47 bits, sous l'hypothèse très optimiste que ces mots de passe sont choisis aléatoirement.

Règle

RègleSecretFaibleEntropie

Si un mécanisme d'authentification utilise un secret de faible entropie, il ne doit exister aucun moyen pour un attaquant actif ou passif de procéder à une recherche exhaustive hors-ligne sur ce secret.

Information

- Il convient, lorsqu'on étudie la sécurité d'un mécanisme d'authentification, de distinguer les calculs qui peuvent être effectués hors-ligne, c'est-à-dire sans accès à une ressource telle qu'un serveur, et les opérations qui doivent être réalisées en ligne. Il faut également prendre en compte le nombre de ressources susceptibles d'être attaquées.
- Notons enfin que des attaques fondées sur le paradoxe des anniversaires peuvent également s'appliquer, par exemple si une liste d'empreintes de mots de passe d'accès à un système est disponible.

2.3 Génération d'aléa cryptographique

Comme expliqué en annexe A.3, la qualité de l'aléa est un élément crucial pour la sécurité d'un système, que ce soit pour la génération des clés ou pour le bon fonctionnement des mécanismes cryptographiques.

Dans cette partie sont énoncées les règles et les recommandations applicables à un générateur d'aléa destiné à alimenter un système cryptographique de manière durable.

Un tel générateur consiste généralement en la combinaison de différentes sources d'aléa et d'une couche de retraitement algorithmique.

Plus précisément, une **source d'aléa** désignera un dispositif susceptible de fournir en entrée du retraitement des éléments au moins partiellement aléatoires. Une source d'aléa est :

- **physique** s'il s'agit d'un **générateur physique d'aléa**, c'est-à-dire d'un dispositif physique spécialement conçu pour produire des bits aléatoires en quantité (théoriquement) illimitée. Un tel générateur élabore ses sorties à partir de phénomènes physiques présentant un caractère imprédictible (par exemple le bruit thermique électronique, des phénomènes quantiques, etc.);
- **systémique** si elle correspond à une accumulation d'événements partiellement imprévisibles provenant du système (par exemple le procédé d'accumulation d'aléa de `/dev/random` sous Linux);
- **importée** s'il s'agit de données secrètes parfaitement aléatoires spécialement fournies par le reste du système d'information;
- **manuelle** s'il s'agit de données aléatoires secrètes obtenues par action intentionnelle d'un utilisateur (par exemple : frappes au clavier, mouvements de la souris, etc.).

On note que selon les cas les sources d'aléa peuvent être disponibles de manière régulière ou, au contraire, ponctuelle.

Un **retraitement algorithmique** est un mécanisme de nature cryptographique destiné à combiner différentes sources d'aléa et à produire des sorties indistinguables de sorties produites par une source d'aléa idéale, ne présentant ni biais ni dépendances indentifiables. En particulier, la connaissance d'une partie des sorties ne fournit aucune information sur les autres sorties.

Un **état interne** est une donnée secrète dédiée au retraitement, destinée généralement à accumuler l'entropie des sources d'aléa. Les éventuelles clés secrètes utilisées par les mécanismes cryptographiques employés par le retraitement font également partie de l'état interne.

En l'absence de source d'aléa régulière, c'est-à-dire si les sources d'aléa disponibles sont ponctuelles, on note que le retraitement s'apparente à un **générateur pseudo-aléatoire**. Il possède nécessairement un état interne et une source d'aléa ponctuelle pour l'initialiser.

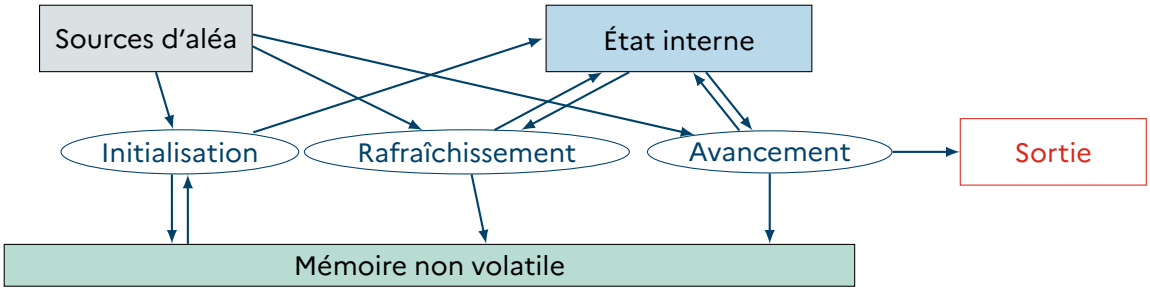


Figure 1 – Architecture générique pour la génération d'aléa cryptographique.

Dans le cas des systèmes pouvant être mis hors tension (cas considéré par défaut dans ce qui suit), un élément de sécurité important consiste en l'utilisation d'une **mémoire non volatile**, protégée en confidentialité et en intégrité, pour stocker des données qui seront utilisées lors de l'initialisation suivante. Ces données peuvent par exemple être mises à jour par les fonctions d'initialisation et/ou d'avancement. En particulier, de telles dispositions permettent de se protéger des attaques par rejeu et de fournir suffisamment d'entropie à l'algorithme de retraitement lors de l'initialisation.

Finalement, les différents éléments constituant un générateur d'aléa cryptographique peuvent être représentés comme sur la figure 1, étant entendu que tous les composants ne sont pas forcément nécessaires et que le détail des fonctionnalités représentées peut varier d'un système à un autre.

Information

Les règles et recommandations applicables aux générateurs d'aléa se fondent sur le constat qu'il est aujourd'hui difficile de fournir une preuve convaincante concernant la qualité de l'aléa issu d'un générateur physique, alors qu'il est relativement aisé de se convaincre de la qualité d'un bon retraitement. Ces règles sont cependant susceptibles d'être revues si des avancées théoriques importantes sont effectuées dans le domaine des générateurs physiques d'aléa.

2.3.1 Architecture d'un générateur d'aléa

Règle

RègleArchiGénAléa

1. Un retraitement algorithmique disposant d'un état interne doit être employé.
2. En l'absence de générateur physique d'aléa, le retraitement algorithmique doit disposer d'une mémoire non volatile.
3. La taille de l'état interne doit être d'au moins 192 bits.
4. La qualité des sources d'aléa ponctuelles ou régulières utilisées pour initialiser l'état interne doit être suffisante pour assurer à la valeur initiale de cet

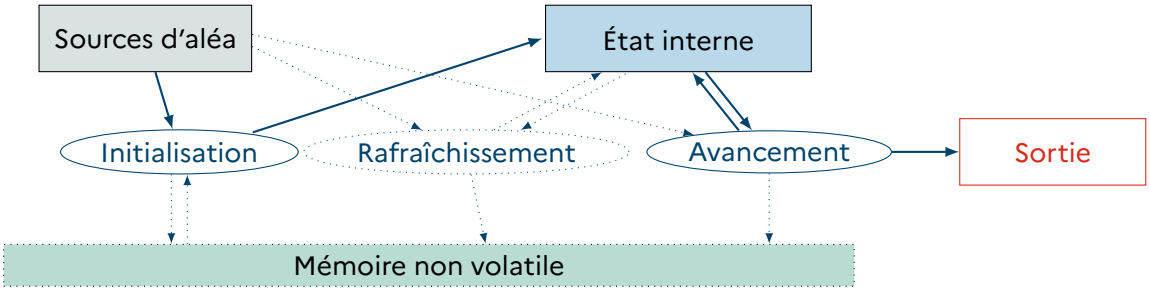


Figure 2 – Architecture minimale pour la génération d'aléa.
(Les pointillés figurent les éléments recommandés.)

état une entropie voisine de sa longueur, ou tout au moins supérieure au seuil défini dans la règle **RègleArchiGénAléa.3** si un raisonnement permet d'établir qu'aucune faiblesse n'en résulte.

Reco

RecoArchiGénAléa

Il est recommandé d'utiliser un retraitement avec un état interne d'au moins 256 bits, une mémoire non volatile *et* une source d'aléa rafraîchissant régulièrement l'état interne du générateur.

Information

- L'emploi d'un générateur physique non retraité est exclu. De même, l'emploi de retraitements élémentaires (« lissages »), ou dont l'état interne est trop petit est jugé insuffisant.
- Un retraitement algorithmique est, par nature, très différent d'un générateur physique d'aléa. En effet, le retraitement est un algorithme déterministe qui ne « génère » pas d'aléa mais uniquement des suites de bits indistinguables de suites réellement aléatoires en partant d'une graine aléatoire de taille suffisante. A l'instar d'autres mécanismes cryptographiques, la sécurité d'un retraitement algorithmique peut être étudiée sous l'hypothèse d'une bonne initialisation et de la robustesse des primitives cryptographique qu'il met en œuvre.
- Le bon fonctionnement du retraitement algorithmique peut facilement être contrôlé, à l'instar de tout autre mécanisme déterministe cryptographique. Il y a là une différence majeure avec les générateurs physiques d'aléa pour lesquels il est tout juste possible de contrôler qu'ils ne sont pas dans un état de panne évident au moyen de tests statistiques. En particulier, utiliser des tests statistiques de panne à la sortie du retraitement algorithmique est non seulement inutile mais peut même s'avérer dangereux pour la sécurité. Par ailleurs, il convient d'appliquer les mêmes règles d'implémentation aux algorithmes de retraitement que pour tout autre mécanisme cryptographique.
- Par **rafraîchissement** de l'état interne, on entend le calcul d'un nouvel état interne combinant l'ancien état et des données aléatoires externes. (Il ne

s'agit donc pas d'une simple réinitialisation.) En cas de source continue d'aléa, le rafraîchissement est souvent combiné à l'avancement.

- Dans le cas où le retraitement utilise des clés secrètes, celles-ci sont considérées comme faisant partie de l'état interne, en particulier pour le calcul de sa taille.
- La méthodologie d'évaluation des générateurs d'aléas physiques AIS31 définie par le BSI [10] est basée sur l'établissement d'un modèle stochastique décrivant le fonctionnement du générateur physique, et exige des tests statistiques en phase d'évaluation et des tests de panne en cours de fonctionnement. Un générateur physique évalué selon cette méthodologie est présumé conforme aux règles et à la recommandation énoncées ci-dessus.

2.3.2 Générateur physique d'aléa

Le générateur physique d'aléa est conçu pour générer de l'aléa tout au long de la vie du système. Il importe donc de garantir autant que possible la qualité et la fiabilité de cet aléa.

Règle

RègleGénPhysAléa

1. Le générateur physique d'aléa doit disposer d'une description fonctionnelle. Celle-ci doit notamment indiquer les principes concourant à la génération des sorties qu'il produit.
2. Des tests statistiques en sortie du générateur physique ne doivent pas faire apparaître de défauts significatifs dans l'aléa généré.

Reco

RecoGénPhysAléa

Il est recommandé qu'un raisonnement permette de justifier la qualité de l'aléa produit par le générateur physique.

Information

- La conception d'un générateur physique ne repose pas simplement sur la juxtaposition de composants physiques, en espérant que l'assemblage obtenu fournisse un aléa satisfaisant. Une certaine réflexion doit guider le concepteur dans ses choix. Ces choix de conception et la réflexion qui les a guidés doivent donc apparaître dans un document. Il s'agit, entre autres, de décrire les sources physiques utilisées et le traitement appliqué à ces sources.
- Il est souvent recommandé de surveiller la qualité de l'aléa issu d'une source physique au moyen de tests statistiques élémentaires afin de détecter d'éventuels blocages. Ces tests doivent être réalisés avant tout

retraitement. Il convient également de spécifier très précisément la conduite à tenir en cas de détection de panne par ces tests.

- L'étape suivante, qui est une simple recommandation, est de justifier les choix faits par un raisonnement, qu'il soit heuristique ou rigoureux, qualitatif ou quantifié. La forme et le type de raisonnement sont laissés libres. Son but est de convaincre à la fois les concepteurs et les utilisateurs de la qualité de l'aléa que le générateur produit.
- Force est de constater aujourd'hui qu'il est très difficile de fournir une preuve convaincante concernant la qualité de l'aléa issu d'un générateur physique. Il est donc nécessaire de pratiquer des tests statistiques sur un échantillon représentatif issu directement du générateur physique. Ces tests servent de validation a posteriori des choix de conception. On pourra par exemple utiliser les tests préconisés par le NIST (FIPS 140-2 et SP 800-22), mais tout test paraissant pertinent peut être utilisé.
- Les tests statistiques concernés par la règle **RègleGénPhysAléa.2** sont des tests d'usine ou des tests ponctuels. Il ne s'agit pas d'éventuels tests de panne pouvant être réalisés en fonctionnement, au cours de l'utilisation du générateur physique.

2.3.3 Retraitement algorithmique

Les propriétés attendues du retraitement algorithmique pour la génération d'aléa sont maintenant énoncées.

Règle

RègleGénAléaRetraitement

1. Les primitives cryptographiques employées par le retraitement algorithmique doivent être conformes au référentiel.
2. Dans l'hypothèse où *l'état interne* est fiable, même en cas de défaillance des sources d'aléa présentes, les sorties successives du retraitement doivent être parfaitement aléatoires du point de vue de l'attaquant. De plus, la connaissance de ces sorties ne doit pas mettre en danger la confidentialité des états internes.
3. Une compromission affectant l'état interne et/ou les sources d'aléa éventuellement présentes ne doit pas permettre à l'attaquant de déduire de nouvelles informations sur les sorties passées.

Information

Le but du retraitement est de garantir la qualité de l'aléa généré en s'appuyant sur la robustesse de primitives cryptographiques, sous réserve d'une bonne initialisation. De plus, comme la détection des pannes physiques est délicate, on souhaite limiter leur impact sur la sécurité du générateur final.

Mécanisme conforme

Les mécanismes de retraitement d'aléa suivants définis dans le standard NIST SP 800-90Ar1 [29] sont conformes au référentiel : HMAC_DRBG et Hash_DRBG mettant en œuvre la fonction de hachage SHA-384, et le mécanisme CTR_DRBG mettant en œuvre la primitive de chiffrement par bloc AES-256. La règle **RègleGénAléaRetraitement.3** est satisaite en ce sens que la transition d'état effectuée après le traitement de chaque requête de génération d'aléa est irréversible.

Annexe A
Définitions et concepts

Objectif

L'objet de cette annexe est de rappeler certaines définitions et certains concepts essentiels en cryptographie dans le but de faciliter la compréhension des règles et des recommandations de ce document. Ces rappels couvrent le strict minimum et sont énoncés volontairement de façon non mathématique.

Des compléments permettant de préciser certaines notions mais pouvant être passés en première lecture sont proposés en petits caractères, dans le style de ce paragraphe.

La **cryptologie**, discipline à la frontière entre mathématiques et informatique, est traditionnellement définie comme la « science du secret ». Longtemps concentrée sur la problématique de la confidentialité, à des fins essentiellement militaires ou diplomatiques, la cryptologie a bénéficié d'un essor scientifique important suite au développement de la société de l'information jusqu'à devenir un outil incontournable pour sécuriser les systèmes d'information.

La cryptologie traite de la conception, de la sécurité et de l'emploi de mécanismes cryptographiques, le terme générique de **mécanisme** englobant ici à la fois les **primitives** (fonction de hachage, chiffrement par bloc, etc.), les **modes opératoires** et les **protocoles** cryptographiques.

On divise traditionnellement la cryptologie en deux branches selon que l'on se place du point de vue du concepteur ou de celui de l'attaquant. La **cryptographie** étudie la conception de mécanismes permettant d'assurer des propriétés de sécurité variées comme la confidentialité, l'intégrité ou l'authenticité de l'information. La **cryptanalyse** s'intéresse à ces mêmes mécanismes en tentant d'analyser leur sécurité, voire de la mettre en défaut. Il va de soi qu'il n'y a pas de cryptographie sans cryptanalyse et inversement.

Par ailleurs, sur un plan technique, on distingue habituellement la cryptographie **symétrique**, ou encore cryptographie **à clé secrète**, de la cryptographie **asymétrique**, ou encore cryptographie **à clé publique**. Cette séparation est issue de l'article fondateur de W. Diffie et M. Hellman [15] qui, en 1976, a donné naissance à la cryptographie asymétrique en suggérant que pour chiffrer un message à l'attention d'un destinataire donné il n'est pas nécessaire de partager au préalable un secret avec lui. Dans la suite de ce chapitre, les mécanismes symétriques et asymétriques sont abordés de manière séparée.

Table 1 – Mécanismes cryptographiques et propriétés de sécurité

Propriété de sécurité	Cryptographie symétrique	Cryptographie asymétrique
Confidentialité	Chiffrement symétrique (A.1.1)	Chiffrement à clé publique (A.2.1)
		Encapsulation de clé (A.2.2)
		Établissement de clé (A.2.4)
Intégrité	Code d'authentification de message (A.1.2)	Signature numérique (A.2.3)
Non-répudiation	*Aucune primitive*	

Une seconde dimension de classification des primitives cryptographiques concerne l'objectif de sécurité visé. Traditionnellement, on considère trois besoins de sécurité de la donnée : la confidentialité, l'intégrité et la non-répudiation. À ceux-ci s'ajoute les besoins d'authentification d'entités. D'autres propriétés plus avancées sont envisageables mais celles qui viennent d'être citées sont, de loin, les principales traitées par des moyens cryptographiques.

Le but de la **confidentialité** est de s'assurer que des informations transmises ou stockées ne sont accessibles qu'aux personnes autorisées à en prendre connaissance. Cet objectif de sécurité est classiquement assuré par le chiffrement mais peut également l'être par tout autre moyen approprié, à commencer par des mesures organisationnelles non cryptographiques.

Assurer l'**intégrité** de données consiste à détecter toute altération illégitime de celles-ci. L'altération couvre toute modification, suppression partielle ou insertion d'information réalisées par un attaquant ou même de manière accidentelle. L'intégrité des données est classiquement assurée en cryptographie par des codes d'authentification de message ou des signatures numériques. [11]

La **non-répudiation** vise à éviter qu'une entité puisse nier avoir effectué une certaine action. Elle garantit le lien entre une donnée et son auteur. Le principal mécanisme de non-répudiation est la signature à clé publique. En effet, une signature sur une donnée lie celle-ci à son signataire.

L'**authentification d'entités** vise pour sa part à s'assurer qu'un correspondant est bien celui qu'il prétend être. Elle peut être réalisée de diverses manières, symétriques ou asymétriques.

Les principaux mécanismes offerts par la cryptographie moderne peuvent maintenant être abordés. La table 1 permet de les répartir en fonction de leur nature symétrique ou asymétrique et de leur objectif de sécurité. D'autres mécanismes très intéressants peuvent être cités mais ils ne s'inscrivent pas naturellement dans une telle table. La suite de ce chapitre décrit sommairement ces mécanismes et rappelle les points essentiels, nécessaires à la compréhension du reste de ce document.

11. Il est à noter que les mécanismes de type checksum ne relèvent pas de la cryptographie car ils ne protègent pas contre un attaquant actif mais permettent uniquement de détecter des erreurs accidentelles.

A.1 Cryptographie symétrique

Les mécanismes cryptographiques symétriques se caractérisent par le fait qu'ils utilisent des **clés secrètes** partagées par plusieurs entités, généralement deux. La sécurité d'un mécanisme symétrique repose notamment sur la connaissance exclusive de ces secrets par les entités légitimes.

Une clé secrète est généralement une suite aléatoire de bits, c'est-à-dire de 0 et de 1, de taille fixée. Cette taille de clé, notée *k*, est déterminante pour la sécurité du système car il existe exactement 2^k valeurs de clé différentes de longueur *k*. Les cryptosystèmes symétriques sont en général susceptibles d'être attaqués de manière générique au moyen d'une énumération de toutes les clés possibles. Une telle attaque, dite par « recherche exhaustive », ne peut cependant aboutir que si le nombre de calculs est réalisable par des moyens informatiques raisonnables. Les cryptosystèmes symétriques robustes utilisent donc une taille de clé *k* telle que 2^k opérations soient impossibles à effectuer en un temps raisonnable.

> Afin de fixer les idées sur les ordres de grandeur manipulés et sur les capacités de calcul disponibles de nos jours, quelques exemples numériques concrets sont rassemblés dans la table 2.
>
> Ces chiffres montrent que la fonction 2^k croît extrêmement rapidement avec *k* et que la capacité de réaliser de l'ordre de 2^{128} opérations de calcul en temps raisonnable semble très peu plausible de nos jours, même en disposant de moyens très importants. En outre, il apparaît en tout état de cause qu'effectuer 2^{256} calculs est, et restera, parfaitement hors de portée. Ce constat permet d'écarter toute menace de recherche exhaustive sur des systèmes bien dimensionnés et forme la base de toute sécurité en cryptographie. Ceci va en particulier à l'encontre de l'idée reçue selon laquelle tout mécanisme peut nécessairement être cassé par recherche exhaustive, à condition d'y mettre les moyens.

A.1.1 Chiffrement symétrique

Les mécanismes de cryptographie symétrique les plus connus sont les algorithmes de chiffrement [12]. Ceux-ci garantissent la confidentialité des données échangées entre deux possesseurs d'une clé secrète, même si le canal de communication employé est écouté. Le chiffrement symétrique permet également de sécuriser le stockage statique d'informations, sans intention de les transmettre.

Les problèmes majeurs que sont la génération et l'éventuel échange initial de la clé secrète entre les correspondants doivent être traités en complément du chiffrement.

De plus, l'intégrité de la donnée n'est pas garantie par le chiffrement donc rien ne permet d'exclure des possibilités d'attaques actives visant à modifier les informations transmises ou stockées. On peut ainsi concevoir des scénarios d'attaques au cours desquels un attaquant peut modifier des communications chiffrées et impacter le clair correspondant sans être détecté. Un exemple de cette attaque est détaillé en annexe A.1.2.

Les méthodes de chiffrement symétrique ou à clé secrète se divisent naturellement en deux familles, le **chiffrement par bloc** (« block cipher ») et le **chiffrement par flot** (« stream cipher »), décrites ci-dessous.

12. Le seul terme admis en français est celui de chiffrement ; les termes « cryptage » et « chiffrage » sont incorrects. L'opération inverse du chiffrement est le déchiffrement.

Table 2 – Ordre de grandeur de la valeur de 2^k pour le calcul

2^k	Ordre de grandeur
2^{15}	Opérations élémentaires nécessaires pour l'implémentation d'une primitive symétrique.
2^{32}	Opérations par seconde par cœur de processeur 4 GHz.
2^{46}	Opérations effectuables par seconde sur processeur graphique (GPU).
2^{57}	Opérations par an par cœur de processeur.
2^{60}	Opérations par seconde effectuables par les meilleurs supercalculateurs connus.
2^{70}	Empreintes SHA-256 calculées chaque seconde dans le monde pour pour la blockchain Bitcoin.
2^{71}	Opérations effectuables par an sur GPU.
2^{76}	Opérations par jour effectuables par les meilleurs supercalculateurs connus.
2^{91}	Opérations effectuables en un siècle par les meilleurs supercalculateurs connus.
2^{96}	Empreintes SHA-256 calculées pour la blockchain Bitcoin entre sa création en 2009 et 2025.
2^{119}	Opérations effectuables en 13.8 milliards d'années par les meilleurs supercalculateurs connus.
2^{128}	Opérations effectuables en 13.8 milliards d'années par l'ensemble des processeurs mondiaux.
2^{256}	Électrons dans l'univers.

A.1.1.1 Chiffrement par bloc

Un mécanisme de chiffrement par bloc est la combinaison d'une primitive de chiffrement par bloc et d'un mode opératoire de chiffrement. Il permet de traiter les données à chiffrer, de taille quelconque, par blocs de taille fixe. On notera n le nombre de bits de ces blocs de données ; typiquement, cette taille vaut 128 bits en pratique.

Une primitive de chiffrement par bloc E est une fonction déterministe qui prend en entrée un bloc m de n bits et une clé K de k bits et renvoie un bloc $c = E_K(m)$ de n bits. À clé K fixée, la fonction E_K est une permutation facilement inversible. Selon le contexte, le terme *chiffrement par bloc* peut désigner le mécanisme de chiffrement par bloc visant à assurer la confidentialité d'une donnée de taille arbitraire ou la primitive sous-jacente. [13]

> L'un des exemples les plus connus de primitives de chiffrement par bloc est l'AES (Advanced Encryption Standard), sélectionné par le NIST au terme d'une compétition internationale lancée en 1997. L'AES est conçu pour traiter des blocs de 128 bits au moyen de clés de 128, 192 ou 256 bits ce qui le met hors de portée des attaques génériques.

Une primitive de chiffrement par bloc seule, ne prenant en entrée qu'un bloc de n bits, ne suffit pas à définir un mécanisme de chiffrement par bloc. En effet, un **mode opératoire** est nécessaire pour préciser la manière dont est instrumentée la primitive de chiffrement par bloc sous-jacente pour traiter un message de taille arbitraire. La taille du message n'étant pas toujours un multiple de la taille de bloc n, il convient d'utiliser une méthode

13. La primitive de chiffrement par bloc est une brique élémentaire pouvant être utilisée dans d'autres constructions cryptographiques, comme des fonctions de hachage ou des MAC.

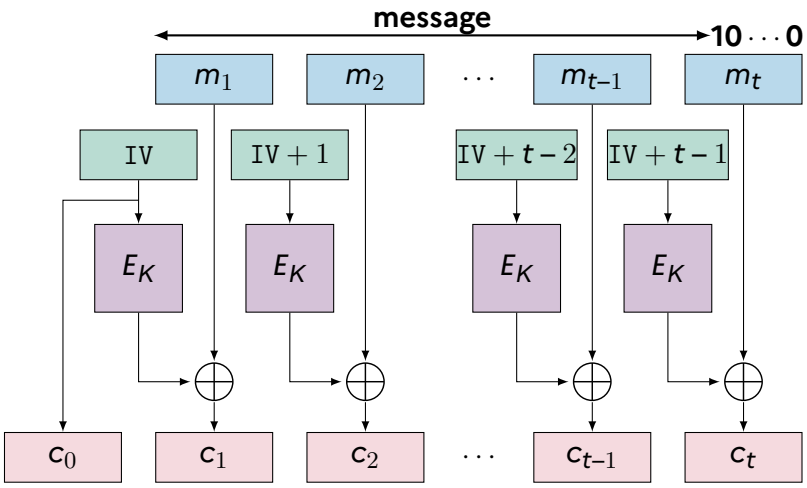


Figure 3 – Mode opératoire CTR.

de bourrage (plus communément appelé *padding* en anglais) pour remplir le dernier bloc de message.

> Un padding usuel consiste à ajouter en suffixe du message un bit valant 1 et autant de 0 que nécessaire afin d'obtenir un nombre total de bits multiple de la taille du bloc.

Par ailleurs, afin de pallier le caractère fondamentalement déterministe de la primitive de chiffrement par bloc, il est possible et recommandé d'employer un mode opératoire probabiliste utilisant un **vecteur d'initialisation** (IV, pour « Initialisation Vector ») public et différent à chaque chiffrement. Cela empêche un attaquant de détecter une répétition du message en observant les chiffrés. Ce vecteur d'initialisation est généré aléatoirement ou de manière déterministiquement contrôlée lors du chiffrement et est transmis avec le chiffré pour permettre le déchiffrement.

> Les deux modes opératoires de chiffrement les plus utilisés sont le mode compteur noté CTR et le mode chaîné noté CBC pour « cipher-block chaining »en anglais. Notons m_1, m_2, ..., m_t les t blocs de message clair obtenus après padding, $\mathtt{IV}$ le vecteur d'initialisation et $c_0, c_1, c_2, \ldots, c_t$ les blocs de chiffrés où par convention $c_0 = \mathtt{IV}$.
>
> Pour CTR, le chiffré est obtenu en calculant $c_i = E_K(\mathtt{IV} + i - 1) \oplus m_i$ pour i allant de 1 à t, comme représenté figure 3. Le déchiffrement se fait de manière similaire : $m_i = E_K(\mathtt{IV}+i-1) \oplus c_i$ pour i allant de 1 à t, avec $\mathtt{IV} = c_0$. Pour CBC, le chiffré est obtenu en calculant séquentiellement $c_i = \mathrm{E}_K(c_{i-1} \oplus m_i)$ pour i allant de 1 à t, comme représenté figure 4. Le déchiffrement se fait en calcule $m_i = c_{i-1} \oplus \mathrm{E}_K^{-1}(c_i)$ pour i allant de 1 à t.
>
> Pour le mode CBC, la réutilisation de l'IV implique que les chiffrés de deux messages dont les k premiers blocs sont identiques auront les mêmes k premiers blocs de chiffré (en omettant c_0 qui représente l'IV), ce qui est détectable par l'attaquant. L'IV doit être aléatoire pour ne pas induire de fuite d'information. Pour le mode CTR, les valeurs des compteurs $\mathtt{IV} + i$ pour i allant de 0 à $t-1$ ne doivent pas être répétées pour des messages différents. Dans le cas contraire, cela induirait une répétition de la suite chiffrante, donc une perte de la confidentialité des données. Pour s'assurer de cela, on peut par exemple limiter les tailles de message d'entrée à maximum $2^{32} - 1$ blocs et générer déterministiquement des vecteurs d'initialisation toujours différents dont les 32 bits de poids faible sont nuls.

Les modes opératoires de chiffrement disposent généralement d'une preuve de sécurité qui réduit la sécurité du mode à la sécurité de la primitive de chiffrement par bloc sous-jacente. Cette preuve de sécurité exige que certaines hypothèses, par exemple sur l'IV, soient satisfaites et donne une borne de sécurité, à savoir une estimation du nombre

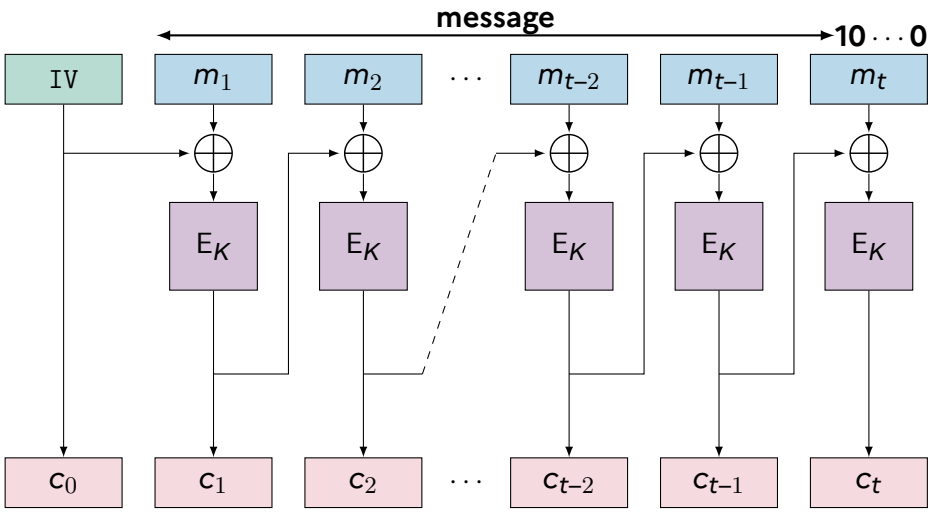


Figure 4 – Mode opératoire CBC.

maximal d'appel à la primitive E_K avant qu'une fuite d'information se produise. Les modes CTR et CBC sont ainsi prouvés jusqu'à la borne des anniversaires sous l'hypothèse de la génération aléatoire de l'IV pour CBC et de la non-répétition du compteur pour CTR. Les preuves de modes opératoires de chiffrement considèrent un attaquant passif (attaquant CPA). L'annexe A.1.3 présente brièvement les modèles de sécurité les plus utilisés en cryptographie symétrique.

A.1.1.2 Chiffrement par flot

Les mécanismes de chiffrement par flot utilisent une approche différente du chiffrement par bloc dans le sens où elles considèrent généralement le message à chiffrer comme une suite de bits combinés de manière simple (généralement un « ou-exclusif bit à bit ») avec une séquence de bits dérivée de la clé secrète (et dans la plupart des cas également d'un vecteur d'initialisation).

Les algorithmes de chiffrement par flot s'inspirent du chiffrement de Vernam, ou *one-time-pad*, qui est à la fois très simple, très sûr et inutilisable en pratique. Si l'on considère un message représenté sous la forme d'une suite de bits m_1, m_2, m_3, … ainsi qu'une clé également vue comme une suite de bits k_1, k_2, k_3, …, le chiffré du message est alors obtenu au moyen de l'opération de « ou-exclusif bit à bit », ou XOR : $c_i = m_i \oplus k_i$. Afin de déchiffrer, il suffit d'appliquer la même opération de XOR du chiffré avec la clé secrète car $c_i \oplus k_i = m_i$.

Le one-time-pad est parfaitement sûr, en terme de confidentialité, si la clé est au moins de même taille que le message à chiffrer et n'est utilisée qu'une seule fois. Ceci restreint considérablement les applications envisageables à cause de la taille de la clé à partager entre émetteur et destinataire ; dans la plupart des cas, cette mise en accord de clé secrète pose un problème similaire à la transmission sécurisée du message lui-même.

L'idée générale du chiffrement par flot est d'utiliser des clés de petite taille, typiquement de l'ordre de 128 bits, et d'en dériver de manière déterministe une suite chiffrante d'allure aléatoire de même longueur que le message. Cette suite chiffrante est ensuite additionnée bit à bit avec le message pour produire le chiffré, similairement au one-time-pad. Le

déchiffrement agit de la même manière, en générant la même suite chiffrante à partir de la clé secrète.

La réutilisation d'une même suite chiffrante pour deux messages différents implique une perte immédiate de confidentialité. En effet, le XOR des deux chiffrés serait alors égal au XOR des deux messages clairs. Pour pallier ce problème, l'introduction d'un vecteur d'initialisation différent à chaque chiffrement est indispensable. Ce vecteur d'initialisation est produit, aléatoirement ou déterministiquement à l'aide d'un compteur, lors du chiffrement et est transmis avec le chiffré.

> Certains chiffrements par flot peuvent être construits à partir de primitives de chiffrement par bloc. Le mode CTR (Counter) et le mode OFB (Output Feedback) en sont deux exemples caractéristiques. Lorsqu'ils sont instanciés avec une primitive de chiffrement par bloc, ils peuvent être considérés à la fois comme des mécanismes de chiffrement par bloc et par flot.

A.1.2 Intégrité cryptographique

Les chiffrements vus dans les sections précédentes ne fournissent que la propriété de confidentialité des données. Cela n'empêche pas un attaquant d'altérer les données chiffrées et d'induire ainsi une modification du clair sans être détecté. Des mécanismes cryptographiques complémentaires sont donc nécessaires pour garantir la détection d'une telle altération. Ces mécanismes fournissent la propriété d'intégrité de la donnée et peuvent être utilisés indépendamment de l'emploi d'un chiffrement.

> Pour illustrer la nécessité de l'intégrité en complément du chiffrement, nous prenons l'exemple de données chiffrées avec un chiffrement par flot. Si un attaquant désire inverser un bit de message clair, il lui suffit d'inverser le bit de chiffré correspondant, et ce sans connaître la suite chiffrante. À titre d'exemple, imaginons le chiffrement du montant d'une transaction financière ; le positionnement du montant à payer dans le message se situe en général à une position fixe et connue. L'attaquant peut alors inverser un bit de cette somme et ainsi transformer un faible montant en une somme très importante, comme si l'on transformait, en notation décimale, 0000017 euros en 1000017 euros.

Plus précisément, la principale technique permettant d'assurer l'intégrité des données consiste à calculer un **code d'authentification de message** (appelé MAC pour « Message Authentication Code »[14]), aussi appelé **motif d'intégrité**, à partir des données dont on veut garantir l'intégrité et d'une clé secrète symétrique. Ce motif d'intégrité est ensuite concaténé à la donnée transmise. Le récepteur calcule de son côté le MAC de la donnée reçue à l'aide de la clé symétrique et le compare au MAC reçu. L'égalité de ces MAC garantit l'intégrité de la donnée puisqu'un attaquant, ignorant la clé symétrique, est incapable de produire un MAC valide de la donnée altérée. Pour cela, il est nécessaire que le MAC soit de longueur suffisante, typiquement d'au moins 96 bits.

Un MAC peut être construit à partir d'un mode opératoire d'intégrité reposant sur la sécurité d'une primitive de chiffrement par bloc sous-jacente, et éventuellement à partir d'un problème mathématique théorique, par exemple une évaluation polynomiale dans un corps fini.

14. Le terme MAC peut faire référence à l'algorithme permettant de calculer le motif d'intégrité et au motif d'intégrité lui-même.

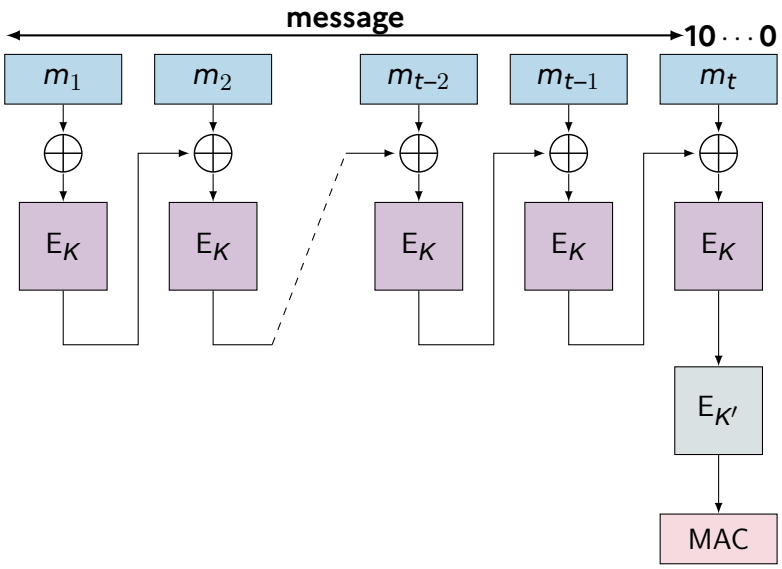


Figure 5 – Mode opératoire CBC-MAC surchiffré

Un exemple de MAC est le mode CBC-MAC surchiffré. Le MAC est alors calculé en appliquant le mode de chiffrement CBC (décrit figure 4) au message, sans utiliser de vecteur d'initialisation (IV), et en ne conservant comme valeur de MAC que le dernier bloc de chiffré, surchiffré avec une clé K', différente de celle utilisée dans la chaîne CBC. Graphiquement, on obtient la représentation symbolique de la figure 5. Toute modification, même minime, du message à protéger engendre un résultat totalement différent du MAC.

Un autre MAC très utilisé est GMAC (Galois MAC) opérant sur des corps finis de grande taille (128 bits). GMAC interprète le message comme un polynôme et l'évalue en la clé, puis chiffre ce résultat à l'aide d'une primitive de chiffrement par bloc pour obtenir le motif d'intégrité. Il est prouvé que la sécurité de GMAC repose sur la robustesse de la primitive de chiffrement par bloc.

Notons enfin qu'un MAC valide ne permet que de garantir l'intégrité d'un message. Si l'on souhaite assurer l'intégrité d'un ensemble de messages formant une communication, il convient de prendre en compte des mesures contre le rejeu ou la suppression de certains messages accompagnés de MAC valides.

L'utilisation conjointe d'un algorithme de chiffrement et d'un mécanisme d'intégrité est appelée chiffrement intègre ou plus communément chiffrement authentifié, depuis l'anglais *Authenticated Encryption*. Pour ce faire, il est possible d'utiliser deux mécanismes indépendants ou un seul mécanisme intégrant les deux propriétés de sécurité. Dans ce second cas, les mécanismes utilisés sont souvent basés sur un mode opératoire de chiffrement authentifié instancié avec une primitive de chiffrement par bloc.

Les mécanismes de chiffrement authentifié les plus utilisés sont AES-GCM(AES-Galois Counter Mode) et AES-CCM(AES-Counter with CBC-MAC), standardisés par le NIST.

A.1.3 Modèles de sécurité

Les mécanismes cryptographiques sont souvent complexes et leur sécurité est difficile à appréhender a priori. Les propriétés de confidentialité et d'intégrité sont générales et recouvrent un éventail nuancé de notions de sécurité dépendantes des capacités estimées de l'attaquant. La cryptographie moderne formalise ces notions à travers différents **modèles de sécurité** définissant la portée d'une attaque et les capacités de l'attaquant

considéré. Il devient alors possible de prouver la sécurité des mécanismes dans un certain modèle à partir de certaines propriétés, plus simples, des primitives cryptographiques sous-jacentes. Il convient néanmoins de vérifier la pertinence du modèle de sécurité avec le contexte envisagé.

> Une propriété naturellement attendue d'un algorithme de chiffrement est que le chiffré ne donne aucune information sur le clair à un attaquant qui ne dispose pas de la clé secrète. Néanmoins, la taille du clair est une information particulièrement complexe à protéger tout en gardant une efficacité pratique. La plupart des modèles de sécurité accepte donc que l'attaquant puisse déduire la taille du clair à partir de celle du chiffré.

Un modèle de sécurité est défini par un **problème de décision** et un **oracle** auquel a accès l'attaquant. Le problème de décision le plus commun est le problème d'indistinguabilité (IND) défini comme suit. L'attaquant propose deux messages différents M_0 et M_1 de son choix (de même taille) et obtient le chiffré C de l'un des deux. Il doit alors déterminer lequel des deux messages a été chiffré avec probabilité significativement supérieure à $1/2$. S'il n'en est pas capable, cela indique que la vue d'un chiffré ne contient pas d'information exploitable sur le clair associé.

L'oracle définit formellement la capacité de l'attaquant en lui octroyant la capacité de chiffrer et déchiffrer des messages avec plus ou moins de liberté. Par exemple, dans le modèle IND-CPA, l'oracle CPA autorise uniquement à l'attaquant de chiffrer un ensemble de messages de son choix. Certaines variantes de cet oracle autorisent le chiffrement de messages qui peuvent dépendre de précédentes réponses de l'oracle, on parle alors d'oracle adaptatif. Le modèle IND-CCA définit quant à lui un oracle qui permet aussi de déchiffrer des données (différentes de C). Le modèle utilisant sa version adaptative, IND-CCA2, est considéré comme offrant le niveau de sécurité le plus élevé parmi les chiffrements. Certains modèles de sécurité permettent aussi de formaliser les propriétés de sécurité liées à l'intégrité (par exemple INT-CTXT), ce qui permet de formaliser la sécurité de chiffrements authentifiés.

> Une propriété des chiffrements déterministes (par exemple des modes opératoires de chiffrement sans IV) est que deux chiffrements du même message génèrent des chiffrés identiques, dévoilant potentiellement de l'information à l'attaquant. Dans le modèle IND-CPA, l'attaquant peut demander à l'oracle le chiffré de M_0 ou M_1, et vérifier lequel des deux est égal à C, répondant ainsi correctement au problème décisionnel de l'indistinguabilité. Ainsi, un tel chiffrement déterministe ne peut pas être prouvé dans le modèle IND-CPA, et nécessite un autre modèle plus faible, apportant une sécurité moindre.

Les modèles de sécurité utilisés en cryptographie moderne les plus conservateurs, tels que IND-CCA2 ou des conjonctions de modèles de confidentialité et d'intégrité, permettent de capturer l'ensemble des moyens d'attaquants quelque soit le contexte d'utilisation des mécanismes cryptographiques. Évaluer la sécurité de mécanismes dans de tels modèles permet de minimiser les risques de faille de sécurité lors de leur emplois au sein de constructions plus complexes et plus généralement de s'assurer qu'aucune attaque liée à la conception de ce mécanisme n'existe. Enfin, la cryptographie moderne propose des mécanismes cryptographiques permettant d'atteindre de tels niveaux de sécurité sans réduire fortement les performances ; il serait par conséquent dommage de se priver de l'emploi de ces mécanismes.

A.1.4 Fonctions de hachage et fonctions à sortie extensible

Certaines primitives cryptographiques ne sont pas paramétrées par des clés. Ces primitives sont néanmoins souvent classées parmi les primitives symétriques dans la mesure où leurs caractéristiques principales empruntent beaucoup aux primitives à clé secrète.

La plus importante de ces primitives est la fonction de hachage dont le but est de transformer, de manière déterministe, une suite de bits de longueur quelconque, en une **empreinte**, ou **haché**, de taille fixe. Une fonction de hachage cryptographique doit être non inversible, c'est-à-dire qu'il ne soit pas possible étant donné une empreinte de trouver un message dont l'image par la fonction de hachage est égale à celle-ci : on dit que la fonction de hachage est **résistante aux préimages**. En outre, on demande souvent qu'il ne soit pas possible de trouver deux messages distincts ayant une même empreinte. On parle alors de **résistance aux collisions**.

> Il est à noter qu'une fonction de hachage étant une fonction d'un ensemble de taille potentiellement infinie vers un ensemble de taille finie, il existe une infinité de telles collisions. On demande cependant qu'il ne soit pas possible de calculer pratiquement, en temps raisonnable, une telle collision.
>
> L'attaque générique pour trouver des collisions exploite le paradoxe des anniversaires : si l'on note n le nombre de bits de l'empreinte, pour trouver une collision, il suffit de calculer de l'ordre de $2^{n/2}$ hachés de messages aléatoires avant que deux de ces messages ne fournissent la même empreinte. Les fonctions de hachage sont essentiellement dimensionnées en prenant en compte cette attaque générique.
>
> Il existe d'autres propriétés de sécurité des fonctions de hachage. Par exemple, à partir d'un message et de son empreinte, il doit être impossible de trouver en pratique un autre message qui donne la même empreinte : on parle alors de **résistante aux secondes préimages**. De telles propriétés découlent souvent de la résistance aux préimages ou de la résistance aux collisions.

Une des nombreuses applications des fonctions de hachage apparaît dans les mécanismes de signature numérique afin de réduire le message de longueur quelconque à une simple empreinte de petite taille fixée. Elles sont également utilisées comme primitive de base dans certains MAC (par exemple HMAC).

On appelle fonction à sortie extensible ou **XOF** (de l'anglais « eXtendable Output Function ») une fonction sans clé XH qui à un message binaire M de taille arbitraire et à un entier $\ell \geq 0$ associe une empreinte $XH(M, \ell)$ de longueur ℓ bits. Lorsque $\ell < \ell'$, $XH(M, \ell)$ doit être égal pour tout M au préfixe de $XH(M, \ell')$ constitué de ses ℓ premiers bits.

Généralement, une XOF est définie à niveau de sécurité fixé λ. Définir des propriétés de sécurité exhaustives et précises de XOF est une tâche ardue, mais il est tout de même possible de définir certaines propriétés requises par les XOF. Pour tout intervalle d'entiers de taille λ, la XOF restreinte aux bits de sortie à position dans cet intervalle doit se comporter comme une fonction de hachage avec taille de sortie λ pour laquelle il n'existe pas de meilleure attaque que les attaques génériques. De plus, à message inconnu d'entropie suffisamment grande, la XOF se comporte comme un mécanisme de chiffrement par flot instancié avec une clé de taille suffisamment grande. Cela revient à dire que la sortie de la XOF doit ressembler à une séquence aléatoire et ne doit posséder aucune structure exploitable.

A.2 Cryptographie asymétrique

L'idée majeure de la cryptographie asymétrique est que les opérations publiques (le chiffrement, la vérification de signature, etc.) n'ont pas nécessairement besoin d'utiliser les mêmes clés que les opérations privées (le déchiffrement, la signature, etc.). Ainsi, la cryptographie asymétrique, telle que décrite par W. Diffie et M. Hellman [15], utilise des « clés privées », que seul un utilisateur possède et peut utiliser pour les opérations privées, et des « clés publiques », que tout le monde connaît et peut utiliser pour les opérations publiques. [15] Les clés publiques et privées sont reliées par des équations mathématiques, ces équations étant la base de problèmes que l'on croit difficiles à résoudre. Pour illustrer la notion de difficulté en pratique, une image simple est la suivante : à partir d'un pot de peinture jaune et d'un pot de peinture bleue, il est facile par simple mélange d'obtenir un pot de peinture verte. Par contre, l'opération inverse consistant à séparer les pigments jaunes des pigments bleus, sans être théoriquement infaisable, l'est en pratique. Dans cet exemple, le vert fait office d'élément public, et le bleu et le jaune sont privés : tout le monde sait que le but est de diviser le vert en bleu et jaune pour retrouver la clé privée, mais l'opération est considérée comme très difficile. De même, dans le monde mathématique, il existe de telles opérations inversibles mais asymétriques dans leur difficulté. On les appelle souvent problèmes difficiles ou asymétriques.

> La plus connue de ces opérations asymétriques est la multiplication de grands nombres premiers. L'opération inverse, consistant à retrouver ces nombres à partir du résultat, est appelée factorisation ou encore décomposition en produit de facteurs premiers. Choisir deux nombres premiers de taille fixée et les multiplier est une opération facile mais, dès que la taille des entiers manipulés est suffisamment grande, aucune méthode efficace permettant de retrouver ces facteurs premiers à partir de la simple valeur de leur produit n'est connue à ce jour. Insistons sur le fait qu'une telle factorisation n'est pas impossible ; elle est mathématiquement parfaitement définie et l'on connaît des algorithmes simples permettant de la retrouver. Le problème réside dans la complexité temporelle, c'est-à-dire dans le temps de calcul nécessaire à ces méthodes pour trouver le résultat. De plus, rien ne permet de dire qu'il n'existe pas de méthode efficace. On peut tout juste affirmer qu'aucune méthode efficace de factorisation utilisant une technologie disponible n'a été rendue publique à ce jour.

Aucun détail nécessitant de lourds rappels mathématiques ne sera donné ici. Il suffira de noter simplement que ces problèmes ne deviennent réellement difficiles, et donc cryptographiquement intéressants, que pour des tailles suffisantes de certains paramètres. Ce sont ces tailles qui déterminent la sécurité intrinsèquement apportée par le problème de base à l'ensemble du mécanisme. La sécurité d'un système asymétrique s'évalue donc en fonction de la difficulté à résoudre numériquement un certain problème mathématique et non pas en fonction de la taille de l'espace des clés. Ainsi, par exemple, lorsque l'on parle de RSA-2048, ceci signifie que l'on utilise RSA avec un paramètre, appelé module, long de 2048 bits et obtenu par multiplication de deux nombres premiers de 1024 bits chacun. Il ne faut par conséquent pas s'étonner de voir apparaître en cryptographie asymétrique des paramètres ou des clés de 2048 bits ou plus alors qu'en cryptographie symétrique les clés dépassent rarement 256 bits. Les tailles de clés symétriques et asymétriques ne sont donc pas comparables. [16]

15. L'usage veut que l'on réserve le terme de clé secrète aux applications symétriques et le terme de clé privée aux applications asymétriques.

16. Ainsi, si 256 bits de clés sont largement suffisants pour la cryptographie symétrique, un module RSA de 256 bits peut être factorisé sur un simple ordinateur en moins d'une heure.

A.2.1 Chiffrement asymétrique

Un mécanisme de **chiffrement asymétrique** est spécifié par la donnée de trois algorithmes. Le premier permet de générer une bi-clé. Le second, utilisé par n'importe qui, prend en entrée un message et une clé publique, et renvoie un chiffré de ce message sous la clé publique. Enfin, le troisième algorithme est utilisé par le possesseur de la clé privée, prend en entrée cette clé ainsi qu'un chiffré et renvoie le message correspondant.

> Une image classique consiste à imaginer une boîte aux lettres (physique). L'équivalent de la clé publique est la boîte à l'adresse du destinataire, adresse consultable par tous, dans un annuaire par exemple. L'équivalent de la clé privée est la clé de la boîte dont ne dispose, normalement, que le propriétaire de la boîte. Afin de transmettre un document, il « suffit » de prendre connaissance de l'adresse du destinataire et de le lui envoyer sous pli scellé. Lors de la réception, seul le destinataire peut prendre connaissance du document à la condition que lui seul possède la clé. Cette image permet en outre de comprendre le problème posé par l'authentification de la boîte du destinataire puisqu'il faut savoir relier de manière fiable le destinataire à sa boîte pour s'assurer que la bonne personne recevra le document. Pour le destinataire des documents si aucun mécanisme n'est prévu pour authentifier l'émetteur, il ne peut davantage s'assurer de l'intégrité de ce qu'il reçoit.

Il est important de noter que, contrairement au cas des mécanismes symétriques, il n'est pas ici nécessaire que les deux correspondants partagent, au préalable, une clé secrète. Ceci permet théoriquement de résoudre très élégamment les problèmes de mise en accord de clé inhérents à la cryptographie symétrique. Il se pose cependant le problème de la certification des clés publiques visant à s'assurer qu'une clé publique utilisée pour chiffrer appartient bien au correspondant à qui l'on destine le message.

Le problème apparaît plus clairement formalisé sous le diptyque confidentialité/authentification. En effet s'il est facile de comprendre que l'échange d'un secret nécessite de la confidentialité, on aurait tendance à oublier que l'authentification de l'origine du secret (c'est-à-dire à la fois la garantie de son intégrité et de l'émetteur) est cruciale. Si le chiffrement asymétrique lève le problème de la confidentialité, le problème de l'authentification reste quant à lui entier. Il est résolu par des mécanismes de certification comme cela est indiqué plus loin.

Comme dans le cas de mécanismes de chiffrement par bloc, l'utilisation de chiffrement à clé publique avec des messages de taille quelconque doit être très précisément spécifiée. Ceci implique d'être capable de formater et de compléter les messages afin d'appliquer l'algorithme de chiffrement ; on parle de « padding » ou « bourrage ». L'utilisation de données aléatoires est également nécessaire afin de rendre le chiffrement probabiliste et par conséquent d'éviter que le chiffrement du même message à deux reprises produise des chiffrés identiques. Ceci est fondamental pour des applications où l'espace des messages est restreint à un petit sous-ensemble des messages possibles.

> Notons enfin que pour des raisons d'efficacité il est inutile de chiffrer de grands messages au moyen d'un mécanisme asymétrique. Une méthode bien plus efficace, désignée sous le terme de « chiffrement hybride », consiste à choisir une clé de session pour un mécanisme de chiffrement symétrique et à ne chiffrer avec le mécanisme à clé publique que cette clé de session, le message étant lui chiffré en symétrique avec la clé de session. On peut ainsi transmettre la clé de session de manière sécurisée à son interlocuteur, et un long message peut être chiffré de manière conventionnelle et très efficace.

Notions de sécurité. Il existe principalement deux notions de sécurité pour le chiffrement asymétrique.

La première est la sécurité sémantique [22]. Goldwasser et Micali la définissent de la manière suivante. Un adversaire reçoit une clé publique, et choisit deux messages différents m_0 et m_1. On lance une pièce, qui donne un résultat $b = 0$ ou 1, et on donne à l'adversaire un chiffré c^* de m_b. L'adversaire renvoie b' et gagne si $b = b'$. Le mécanisme de chiffrement est sémantiquement sûr si aucun adversaire n'a une probabilité de gagner non-négligeablement supérieure à $1/2$, soit la probabilité de gagner en répondant au hasard. On trouve cette notion dans la littérature scientifique sous la dénomination *IND-CPA*. En particulier, cette notion implique la confidentialité des données chiffrées, un adversaire ne sachant pas quelle donnée peut se trouver sous un chiffré.

La seconde est la résistance aux attaques par chiffrés choisis et est définie similairement à ce qui précède. Le seul changement est que l'adversaire est autorisé à demander le déchiffrement de chiffrés de son choix, sauf c^*. On trouve cette notion dans la littérature scientifique sous la dénomination *IND-CCA* ou *IND-CCA2*. En particulier, cette notion implique l'intégrité des données chiffrées. En effet, si un adversaire savait manipuler des chiffrés pour en créer de nouveaux, il pourrait gagner le jeu avec haute probabilité, en modifiant c^* et en demandant le déchiffrement du nouveau chiffré.

D'autres notions, plus faibles ou intermédiaires, existent dans la littérature et sont principalement utilisés pour obtenir des réductions plus fines lorsque le chiffrement asymétrique est utilisé dans un autre mécanisme.

A.2.2 Encapsulation de clé

Un **mécanisme d'encapsulation de clé** permet de générer et de protéger en confidentialité (*encapsuler*) une clé de session symétrique à l'aide d'une clé publique à destination du propriétaire de la clé privée associée. Celui-ci peut la récupérer en effectuant l'opération de *décapsulation* à l'aide de sa clé privée. La clé de session est choisie au sein de l'algorithme d'encapsulation, qui ne prend donc en entrée que la clé publique, et est renvoyée en plus du chiffré pour permettre son utilisation par la personne appelant l'algorithme d'encapsulation.

Le lien entre ce mécanisme et le chiffrement asymétrique est fort. Un mécanisme de chiffrement asymétrique est transformé en KEM par la transformation de Fujisaki-Okamoto [20, 21], tandis qu'un KEM et un chiffrement symétrique donnent un mécanisme de chiffrement asymétrique, par la méthode du chiffrement dit *hybride*, à ne pas confondre avec l'hybridation dans le contexte du post-quantique.

Notions de sécurité. Il existe principalement deux notions de sécurité pour les mécanismes d'encapsulation de clé, définies similairement à celles du chiffrement asymétrique.

La première est la sécurité sémantique. On utilise la procédure d'encapsulation pour obtenir un couple (chiffré, clé) (c, K_0) et on tire uniformément aléatoirement une suite de bits K_1 de même longueur que K_0. On lance une pièce, qui donne un résultat $b = 0$ ou 1,

et on donne à l'adversaire la clé publique pk, le chiffré c ainsi que la clé K_b. L'adversaire renvoie b' et gagne si $b = b'$. Le mécanisme d'encapsulation de clé est sémantiquement sûr si aucun adversaire n'a une probabilité de gagner non-négligeablement supérieure à $1/2$, soit la probabilité de gagner en répondant au hasard. On trouve cette notion dans la littérature scientifique sous la dénomination *IND-CPA*. En particulier, cette notion implique la confidentialité de la clé de session symétrique, un adversaire ne sachant pas extraire d'information sur la clé à partir d'un chiffré.

La seconde est la résistance aux attaques par chiffrés choisis et est définie similairement à ce qui précède. Le seul changement est que l'adversaire est autorisé à demander la décapsulation de chiffrés de son choix, sauf c^*. On trouve cette notion dans la littérature scientifique sous la dénomination *IND-CCA* ou *IND-CCA2*. En particulier, cette notion implique l'intégrité des données chiffrées. En effet, si un adversaire savait manipuler des chiffrés pour en créer de nouveaux, il pourrait gagner le jeu avec haute probabilité, en modifiant c^* et en demandant le déchiffrement du nouveau chiffré.

A.2.3 Signature numérique

La **signature numérique** permet de garantir l'intégrité d'un message sans utiliser de clé secrète partagée entre l'émetteur et le destinataire. Elle permet également d'assurer une authentification forte de l'émetteur du message en empêchant ce dernier de nier par la suite avoir envoyé le message ; on parle de propriété de non-répudiation.

La signature est un mécanisme asymétrique qui peut donner l'impression d'avoir de nombreuses similitudes avec le chiffrement à clé publique. Signature et chiffrement ne doivent cependant surtout pas être confondus. Le principal point commun est l'emploi de bi-clés formées d'une clé privée et d'une clé publique associée. La clé privée permet de générer la signature d'un message sous la forme d'une donnée qui lui est accolée[17], à la manière des MAC vus dans l'annexe A.1.2. Afin de vérifier la validité d'une signature, il suffit de disposer de la clé publique. Par conséquent, tout le monde peut potentiellement s'assurer de l'authenticité d'un message puisque la clé publique peut être librement rendue disponible. Ceci réalise donc une version électronique de la signature manuscrite classique, avec des garanties de sécurité plus fortes encore. Notons cependant qu'ici encore la certification de la clé publique est un problème crucial qui est abordé rapidement ci-dessous dans l'annexe A.4 consacrée à la gestion de clé.

Comme pour le chiffrement, la mise en forme à appliquer au message avant signature est très importante en termes de sécurité. Elle utilise souvent une fonction de hachage transformant un message de longueur quelconque en une empreinte de taille fixe et petite.

> Il a pu arriver par le passé qu'on présente la signature numérique comme une sorte de réciproque du chiffrement asymétrique. Ceci provient du fait que dans le cas de RSA, souvent présenté comme illustration, chiffrement et signature sont en effet très proches, la signature d'un message s'apparentant fortement à l'opération de déchiffrement. Ce cas est cependant exceptionnel. La grande majorité des mécanismes de signature ne peuvent être utilisés à des fins de chiffrement.

17. Plus exactement, le mécanisme consistant à calculer une signature et à l'ajouter au message est appelé « signature avec appendice ». D'autres techniques permettant d'économiser un peu de place sont envisageables avec des mécanismes tel que RSA utilisant une permutation à trappe. Ceci n'a cependant que peu d'importance en première approche.

Notions de sécurité. Les mécanismes de signature admettent plusieurs notions de sécurité pertinentes. La notion de base est la *sécurité en contrefaçon*, qui garantit qu'aucun adversaire ne peut produire une signature pour un nouveau message de son choix, en ayant accès à des couples (message, signature) de son choix ainsi qu'à la clé publique correspondante. Il existe des variantes, mais toutes restreignent l'adversaire, par exemple en lui imposant un message pour lequel contrefaire une signature, et sont par conséquent moins intéressantes.

On peut au contraire relâcher la condition précédente en autorisant un adversaire à produire une signature valide différente pour un message déjà signé. Dans ce cas, on parle de sécurité *forte* en contrefaçon. Cette notion est nécessaire pour éviter la rejouabilité dans de nombreux protocoles.

Une ligne de recherche récente [14, 16, 17] considère trois notions supplémentaires et orthogonales aux notions précédentes.

La notion de *propriété exclusive* garantit qu'aucun adversaire ne peut produire deux clés publiques différentes, une signature et un message, de telle sorte que la signature soit vérifiée pour le message sous les deux clés publiques. Cette notion participe à la non-répudiabilité de la signature, en garantissant qu'un signataire ne puisse pas blâmer quelqu'un d'autre pour une signature qu'il a produite.

La notion de *signature liée au message* garantit qu'aucun adversaire ne peut produire une clé publique, une signature et deux messages, de telle sorte que la signature soit vérifiée sous la clé publique pour les deux messages. Cette notion participe aussi à la non-répudiabilité de la signature, en garantissant qu'un signataire malhonnête ne peut pas interchanger le message qu'il signe avec un autre.

La notion de *non-resignabilité* garantit qu'aucun adversaire ne peut produire une paire (clé publique, signature) vérifiant la condition de victoire dans la situation suivante. Un message est choisi aléatoirement, puis signé. L'adversaire reçoit alors la clé secrète, la signature, mais pas le message, seulement certaines informations dessus. L'adversaire gagne si la signature qu'il produit est vérifiée pour le message caché sous la clé publique qu'il produit. Cette notion permet l'authentification du signataire d'un message, lorsque celui-ci est assimilable à un secret partagé entre deux entités.

A.2.4 Authentification asymétrique d'entités et établissement de clé

Les mécanismes de cryptographie asymétrique présentés dans les sections précédentes peuvent être utilisés dans le cadre d'authentification d'entités. Afin d'authentifier une personne dont on connaît avec certitude la clé publique, il suffit par exemple de chiffrer à son attention un message quelconque suffisamment long et de lui demander de retourner ce message clair. De même, on peut lui demander de signer un tel message et ensuite vérifier la validité de cette signature.

Il existe cependant des mécanismes spécifiquement conçus pour le cadre interactif. Ces primitives dites « à divulgation nulle de connaissance » ou « zero-knowledge », bien qu'encore peu utilisées en pratique, sont à

la source de la plupart des mécanismes de signature numérique. On citera par exemple le standard de signature américain DSA (« digital signature algorithm »), qui présente une certaine parenté avec le protocole d'authentification de Schnorr.

Par ailleurs, notons qu'une variante symétrique de ce principe existe, dans lequel une première entité partageant une clé symétrique avec une deuxième entité génère un message aléatoire, l'envoie à cette dernière et lui demande de fournir un MAC valide du message sous la clé secrète.

La cryptographie asymétrique permet également de résoudre le problème de la confidentialité dans l'établissement de clé, qui consiste pour deux entités ne partageant initialement aucun secret commun à se mettre d'accord sur une valeur de clé secrète. [18] La sécurité de l'établissement de clé est un problème délicat, car celui-ci doit se faire à l'aide d'un canal non sécurisé, c'est-à-dire potentiellement écouté voire entièrement contrôlé par un attaquant. L'article fondateur de W. Diffie et M. Hellman [15] décrivait d'ailleurs principalement une solution au problème d'établissement de clé ; le protocole résultant est ce que l'on appelle encore aujourd'hui l'« échange de clé de Diffie-Hellman ».

Techniquement, la remarque fondamentale de W. Diffie et M. Hellman est que, lorsque l'on utilise une structure mathématique dans laquelle le calcul de logarithme discret est difficile, on peut facilement choisir un grand entier secret a et publier x^a sans que a ne puisse être retrouvé par quiconque. Ainsi, afin que deux interlocuteurs notés A et B se mettent d'accord sur un secret commun K, il suffit que A choisisse un entier a et transmette $y = x^a$, que B choisisse un entier b et transmette $z = x^b$. A peut alors calculer $K = z^a = (x^b)^a = x^{ab}$ et B peut faire de même en calculant la même valeur $K = y^b$. Par contre, un attaquant qui écoute passivement la communication ne peut apprendre que y et z ; à ce jour on ne connaît pas de méthode plus efficace que le calcul de logarithme discret pour retrouver les secrets a et b afin d'en déduire le secret partagé K.

Notons cependant que ce protocole élémentaire ne permet pas de garantir la sécurité face à des attaquants actifs pouvant modifier les communications. Il ne garantit également aucune forme d'authentification des interlocuteurs. Une attaque connue sous le nom d'attaque par le milieu permet en effet à un attaquant actif de mettre en défaut la sécurité de l'échange de clé de Diffie-Hellman.

En pratique, les mécanismes d'authentification et d'établissement de clé sont généralement utilisés de concert en pratique car, d'une part, établir une clé entre deux personnes ignorant l'identité de leur interlocuteur a peu d'intérêt et, d'autre part, une simple authentification apporte peu. Le lien entre authentification et établissement de clé doit cependant être réalisé avec soin : il est nécessaire que les deux mécanismes soient imbriqués si l'on souhaite éviter des scénarios tels que l'attaque par le milieu. Un mécanisme d'authentification totalement décorrélé de l'établissement de clé ne protégerait pas le système contre les attaques par le milieu, que l'authentification soit mise en œuvre avant l'établissement de clé ou bien après l'établissement de clé — sur un canal chiffré à l'aide des clés établies au moyen de ce dernier. [19]

A.2.5 Sécurité des primitives asymétriques

La cryptographie moderne a développé des techniques de preuve de nature mathématique afin de tenter de prouver la sécurité des primitives, notamment asymétriques. Ces preuves n'ont pas un caractère absolu mais reposent avant tout sur un modèle de sécurité formalisant les propriétés attendues de la primitive ainsi que les capacités supposées de l'attaquant contre lequel on veut se prémunir.

18. Cette clé secrète pouvant par exemple par la suite être utilisée pour chiffrer des messages.

19. Dans le premier cas, il suffirait en effet à un attaquant d'attendre le succès de la phase d'authentification pour mettre en œuvre une attaque par le milieu. Dans le second cas, il lui suffirait de réaliser une attaque par le milieu contre le mécanisme d'établissement de clé, puis de relayer sur les canaux chiffrés ainsi établis les messages du protocole d'authentification entre ces entités.

Ces preuves sont dites « par réduction » au sens où elles vont ramener la sécurité globale d'une primitive à une hypothèse bien identifiée telle que « factoriser des modules RSA de 2048 bits n'est pas possible [20] ». Les preuves en clé publique n'assurent ainsi jamais une « sécurité inconditionnelle » — c'est-à-dire face à un attaquant de puissance infinie. [21] Elles offrent au contraire le plus souvent une « sécurité calculatoire », c'est-à-dire l'assurance qu'il est impossible d'attaquer une certaine propriété du mécanisme sans résoudre un certain problème mathématique.

> Afin d'illustrer l'importance mais également la difficulté de définition d'un modèle de sécurité, prenons la signature comme exemple. On peut tout d'abord considérer divers types d'attaques, selon les capacités de l'attaquant, qui peuvent aller de la simple connaissance de la clé publique de vérification de signature à la capacité d'obtenir la signature de n'importe quel message de son choix. On peut également s'interroger sur la définition même de ce que l'on entend par succès d'une attaque. Cela peut aller de la simple « contrefaçon existentielle », consistant à réussir à générer une signature valide d'un message non maîtrisé, à un « cassage total », consistant à retrouver la clé privée de signature.
>
> Il existe deux types de sécurité, comme mis en avant la première fois par M. Bellare et Ph. Rogaway [6] : la sécurité « asymptotique » et la sécurité « exacte ». Quand la première se contente d'assurer que le mécanisme est sûr pour des paramètres « assez grands », la seconde énonce clairement des estimations de la difficulté de l'attaque en fonction de la difficulté du problème. Ainsi, il est très important de prendre en compte les résultats qu'apportent la preuve, et de ne pas s'arrêter au fait que le mécanisme est sûr pour des paramètres assez grands. Pour être plus précis, la sécurité exacte conduit à ce que l'on appelle des réductions fines (dans lesquels la difficulté du cassage du mécanisme est de l'ordre de la difficulté du problème mathématiques) et des réductions lâches (dans lesquels il est plus facile — d'un ordre de grandeur non négligeable — de casser le mécanisme que de résoudre le problème). On voit ainsi que les mécanismes à préférer sont ceux apportant les réductions les plus fines possibles, car les autres mécanismes nécessitent de plus grandes clés et paramètres pour une même garantie de sécurité. De même, s'il n'est pas possible de sélectionner un mécanisme avec une réduction fine, il faut dans ce cas utiliser les coefficients de finesse donnés par la preuve pour en déduire les tailles de paramètres et de clés adéquats. Ceci permet de conserver une signification à la garantie de sécurité offerte par la preuve.

Il est clair qu'une primitive asymétrique prouvée sûre, relativement à une hypothèse bien identifiée et raisonnable, est d'autant plus intéressante que l'on a pris en compte des attaquants puissants et des définitions de succès d'attaque modestes dans la preuve. Tout comme pour la sécurité du chiffrement symétrique, vue en annexe A.1.3, les modèles de sécurité utilisés en cryptographie moderne peuvent paraître excessifs mais l'expérience montre que c'est loin d'être le cas et qu'il est important de se placer dans ce cadre contraignant afin d'évaluer correctement les primitives.

A.3 Génération d'aléa cryptographique

La cryptographie fait un usage intensif de données aléatoires, typiquement afin de générer des clés mais également pour bien d'autres applications comme dans les opérations de formatage de messages avant chiffrement ou signature. La qualité des données aléatoires utilisées est parfois critique en termes de sécurité et nécessite de disposer de données aléatoires « de qualité ».

> À titre d'exemple particulièrement frappant de la nécessité de disposer de bon aléa pour certaines applications, citons le standard de signature américain DSA. En utilisant cet algorithme, la signature d'un message nécessite l'emploi d'un nombre aléatoire de 160 bits, gardé secret par le signataire. Pour chaque signature, il est nécessaire de disposer d'un nouveau nombre aléatoire indépendant des précédents et donc à usage unique.

20. En un temps raisonnable.
21. La puissance étant ici la mémoire et la puissance de calcul.

Sans que cela ne remette en cause la sécurité de DSA, on connaît aujourd'hui une attaque qui permet de retrouver la clé privée à condition de disposer de signatures pour lesquelles seulement 2 bits du nombre aléatoire à usage unique sont connus. Cette attaque fonctionne donc très efficacement même si les 158 bits restants sont parfaitement aléatoires et inconnus de l'attaquant. Cet exemple montre que, pour certaines applications, il est impossible de se contenter de nombres partiellement aléatoires.

La génération de bits réellement aléatoires, c'est-à-dire valant 0 ou 1 avec même probabilité et, surtout, indépendants les uns des autres, est particulièrement délicate sur une plate-forme fondamentalement déterministe comme un ordinateur ou un microprocesseur de carte à puce. Il existe cependant des dispositifs physiques spécifiques tirant parti de phénomènes supposés imprévisibles comme le bruit thermique ou le temps s'écoulant entre deux désintégrations d'une source radioactive. On parle alors de générateur d'**aléa physique**.

Il est également possible de générer du **pseudo-aléa**, c'est-à-dire des suites de bits indistinguables de suites réellement aléatoires mais issues d'un mécanisme déterministe initialisé avec un germe ou graine, de petite taille mais réellement aléatoire pour sa part. La plupart des mécanismes de chiffrement par flot sont d'ailleurs construits autour de tels générateurs pseudo-aléatoires.

Notons enfin que, par définition, il est en pratique impossible de distinguer du pseudo-aléa correctement généré de véritables bits aléatoires. Malgré l'existence de notions théoriques bien définies issues de la théorie de l'information, comme celle d'entropie, la seule manière de tester des bits ainsi générés est d'appliquer des tests statistiques visant à détecter des biais statistiques dont la probabilité d'apparition est négligeable dans des séquences de bits issues de sources d'aléa idéales. L'efficacité de ces tests, aussi élaborés soient-ils, demeure cependant très limitée en pratique. Une suite déterministe aussi simple que la succession des décimales de π suffit en effet à tromper la plupart des tests statistiques.

A.4 Gestion de clés

A.4.1 Clés secrètes symétriques

La gestion des clés peut être plus ou moins simple selon les applications. Dans le contexte de mécanismes symétriques, la principale difficulté réside dans la distribution, ou mise en accord, des clés afin de permettre aux correspondants de partager les mêmes secrets initiaux sans que des attaquants potentiels ne les aient interceptés. Ceci peut être réalisé au moyen de techniques asymétriques modernes mais peut également l'être via des méthodes non cryptographiques de nature organisationnelle.

En outre, une durée de vie maximale, appelée **crypto-période**, est en général associée à chaque clé. Une telle durée de vie peut être représentée par une date limite d'emploi ou par un compteur du nombre d'utilisations qui ne doit pas dépasser une certaine limite. Une telle limitation de l'usage des clés vise en général à réduire l'effet d'une éventuelle compromission des clés. Elle peut cependant également s'avérer nécessaire si les primitives cryptographiques sont sous-dimensionnées par rapport au niveau de sécurité visé.

Il est cependant important de bien comprendre que dans un système cryptographiquement bien conçu il ne doit pas y avoir de phénomène « d'usure » des clés limitant leur emploi.

Afin de protéger les clés lors de leur stockage, celles-ci peuvent être elles-mêmes chiffrées avec une autre clé qui n'a généralement pas à être partagée. On désigne en général sous le terme de **clé noire** une clé ainsi chiffrée, par opposition aux **clés rouges** qui sont en clair. Il va de soi que l'ensemble des clés d'un système en fonctionnement ne peuvent toutes être noires simultanément.

Notons enfin un cas particulier d'architecture, encore assez courant, utilisant un secret largement partagé entre un grand nombre d'utilisateurs. La divulgation de telles clés a en général des conséquences dramatiques en termes de sécurité, ce qui est contradictoire avec leur large diffusion. Dans certaines applications, l'usage exclusif de primitives symétriques rend nécessaire l'emploi de telles architectures ; ceci milite fortement en faveur d'une utilisation d'architectures asymétriques permettant de s'en passer.

> À titre d'exemple, imaginons un groupe important de n individus souhaitant pouvoir s'authentifier mutuellement. En utilisant des techniques symétriques, on peut soit prévoir une clé secrète par paire d'individu, ce qui implique que chacun mémorise au moins $n-1$ clés, soit donner la même clé à tout le monde. Si l'on souhaite de plus pouvoir ajouter de nouveaux membres facilement, cette dernière solution devient la seule possible. Cependant, quelle confiance peut-on avoir dans un tel système, même si la clé est stockée dans une enceinte protégée telle une carte à puce ?
>
> Une manière simple de résoudre ce problème avec une technique asymétrique est de faire choisir à chaque membre du groupe une bi-clé dont la clé publique est certifiée par une autorité. Chaque membre doit donc uniquement mémoriser sa bi-clé et la clé publique de l'autorité. On peut ensuite utiliser une des techniques d'identification évoquées en annexe A.2.1.

A.4.2 Bi-clés asymétriques

La gestion des bi-clés en cryptographie asymétrique est à la fois plus simple et plus complexe que dans le cas symétrique. Plus simple, et également plus sûre, car il n'y a plus besoin de partager des secrets à plusieurs. Ainsi, la clé privée n'a besoin d'être connue que de son seul détenteur et certainement pas divulguée à d'autres. Par conséquent, il n'y a en théorie nul besoin de faire générer de telles clés par un tiers. On peut par exemple tout à fait concevoir qu'une clé privée soit générée par une carte à puce et qu'à aucun moment de la vie du système cette clé n'ait à quitter l'enceinte supposée sécurisée de la carte.

Le problème majeur qui se pose réside cependant dans la nécessité d'associer une clé publique à l'identité de son détenteur légitime. Une telle certification de clé publique peut être effectuée au moyen de la signature d'un certificat par une autorité qui certifie de ce fait que telle clé publique appartient bien à tel individu ou entité. Il se pose alors le problème de la vérification de cette signature qui va à son tour nécessiter la connaissance de la clé publique de l'autorité. Afin de certifier cette clé, on peut concevoir qu'une autorité supérieure génère un nouveau certificat, et ainsi de suite. On construit ainsi un chemin de confiance menant à une clé racine en laquelle il faut bien finir par avoir confiance, sans que cette confiance soit garantie par un mécanisme cryptographique. De telles constructions sont désignées sous le terme d'**infrastructure à clé publique** (**ICP** ou **PKI** pour « *public key infrastructure* »). La notion d'**Infrastructure de gestion de clé** (**IGC** ou

KMI pour « *key management infrastructure* ») recouvre quant à elle toutes les opérations d'enregistrement ou d'affectation d'une clé dans un système en y incluant aussi la vérification de l'identité de son possesseur, la gestion de ses équipements, des révocations, etc.

Notons enfin que dans de nombreuses applications pratiques, il est nécessaire de disposer d'une sorte de voie de secours permettant par exemple d'accéder à des données chiffrées sans être pour autant destinataire de ces informations. Les motivations de tels **mécanismes de recouvrement** peuvent être multiples mais il est important d'insister sur le fait qu'elles peuvent être parfaitement légales et légitimes. La méthode la plus simple est le séquestre de clés consistant à mettre sous scellés les clés privées ou secrètes tout en contrôlant les conditions d'accès à ces informations.

Des travaux cryptographiques modernes proposent cependant de nombreuses autres solutions bien plus souples, sûres et efficaces.

Annexe B
Éléments académiques de dimensionnement cryptographique

Le dimensionnement des mécanismes cryptographiques se fait de manière empirique, en se basant sur les capacités actuelles de calculs et sur les records de calculs de problèmes mathématiques difficiles. À partir de ceux-ci, une marge de sécurité est prise en estimant leur progression à moyen et long termes. Cette section a pour vocation de mettre en lumière les différents faits et records ayant servi à établir les dimensionnements de ce guide.

B.1 Coût des calculs et attaques pratiques en cryptographie symétrique

En cryptographie symétrique, il est maintenant communément admis qu'une clé de 64 bits ou moins ne fournit aucune sécurité contre un adversaire motivé. L'estimation du nombre de bits de clé garantissant une sécurité acceptable est faîte en s'appuyant sur les coûts actuels des calculs. On considère que ceux-ci tiennent essentiellement de l'énergie nécessaire au calcul, que l'on peut extrapoler à partir de données existantes. Dans cette optique, nous fournissons les exemples suivant :

- En 2026, retrouver une clé DES de 56 bits par force brute coûte de l'ordre de 30 euros et prend de l'ordre de 26 heures sur une machine dédiée d'une valeur d'environ 100 k€.
- En 2026, une préimage partielle de SHA-256, i.e., dont le haché SHA-256 commence par 79 bits à zéro, est trouvée toutes les 10 minutes au sein de la blockchain Bitcoin. Elle rapporte environ 200 k€ à celui qui la trouve et on estime que la dépense énergétique nécessaire au calcul est du même ordre de grandeur.

Ces deux exemples mettent en valeur des coûts par opération de calcul qui diffèrent d'un facteur de l'ordre de 2^{10}, dû en partie aux disparités des coûts de l'énergie à travers le monde. En se basant sur le coût par opération du minage, le plus conservateur des deux, une recherche exhaustive sur 128 bits ou équivalent coûterait de l'ordre de 100 milliards de milliards d'euros. De plus, même si la quantité d'énergie était rassemblée pour effectuer ces calculs, les équipements disponibles dans le monde ne suffiraient très probablement pas à les mener à bout en temps raisonnable. On considère pour ces raisons qu'une attaque par force brute sur une clé de 128 bits est hors de portée de quiconque.

Dans un second registre, pour illustrer la praticabilité de certaines attaques contre diverses faiblesses cryptographiques, nous mentionnons les attaques suivantes :

- En 2016, Bhargavan et Leurent ont montré qu'il était possible pour un attaquant de retrouver des données secrètes dans divers protocoles lorsqu'une primitive de chiffrement par bloc avec des bloc de 64 bits est utilisée avec le mode CBC. Cela nécessite la capture d'environ 785 Go de données.
- En 2017, Stevens et al. ont exhibé et implémenté une attaque par collision sur SHA-1, dont le temps de calcul est estimé à $2^{63.1}$ appels à SHA-1. Leur estimation du coût de l'attaque sur GPU loués pour l'occasion est de l'ordre de 150 k€.
- En 2020, Leurent et Peyrin ont exhibé et implémenté une attaque par collision à préfixe choisi sur SHA-1 dont le temps de calcul est estimé à $2^{63.5}$ appels à SHA-1. Leur estimation du coût de l'attaque sur GPU loués pour l'occasion est de l'ordre de 50 k€.

Ces attaques soulignent d'une part que les attaques sur les modes utilisant des chiffrements par bloc de 64 bits ne sont pas que théoriques, et d'autre part,qu'une entité disposant d'importants moyens de calculs pourrait générer des collisions personnalisées sur SHA-1.

B.2 Records de calculs de factorisation

Les principaux records successifs en termes de calcul de factorisation de modules produits de deux nombres premiers de taille comparable sont listés dans la table 3.

Les deux premiers records ont utilisé l'algorithme du crible quadratique et les suivants l'algorithme du crible algébrique, le plus efficace connu à ce jour pour factoriser de grands entiers quelconques. La plupart de ces challenges ont été proposés par la société RSA. [22]

B.2.1 Factorisation par des machines dédiées

De même qu'il est possible de concevoir des machines dédiées conçues exclusivement à des fins de calculs exhaustifs sur des clés de chiffrement symétrique, il est aujourd'hui sérieusement envisagé de concevoir de telles machines afin de factoriser de grands modules RSA. Le projet le plus abouti a été présenté par Adi Shamir et Eran Tromer en août 2003. Les estimations de coût indiquent qu'il semblerait possible de réaliser pour quelques dizaines de millions d'euros une machine capable de factoriser des modules de 1024 bits en moins d'un an. À ce jour aucune annonce de conception concrète n'a cependant été faite.

B.2.2 Autres records de factorisation

Notons enfin que dans certains cas il est possible d'employer des algorithmes de factorisation particuliers, plus efficaces que les algorithmes généraux mais ne s'appliquant pas à tous les entiers (et en particulier, à ce jour, ne s'appliquant pas aux modules RSA).

22. Les challenges de la société RSA étaient disponibles sur http://www.rsasecurity.com/rsalabs/challenges, mais ont été retirés en 2007.

Table 3 – Records de factorisation d'entiers de type RSA.

Date	Bits	Chiffres	Auteurs
1993-06-12	**397**	120	Denny, Dodson, Lenstra et Manasse
1994-04-02	**426**	129	Atkins, Graff, Lenstra, Leyland
1996-04-10	**432**	130	Lenstra et al.
1999-02-02	**466**	140	te Riele et al.
1999-08-22	**512**	155	te Riele et al.
2002-01-18	**524**	158	Bahr, Franke, Kleinjung
2003-04-01	**530**	160	Bahr, Franke, Kleinjung, Lochter et Boehm
2003-12-03	**576**	174	Franke et Kleinjung
2005-05-09	**663**	200	Bahr, Boehm, Franke et Kleinjung
2009-12-12	**768**	232	Kleingjung et al.
2019-12-02	**795**	240	Thomé et al.
2020-02-28	**829**	250	Thomé et al.

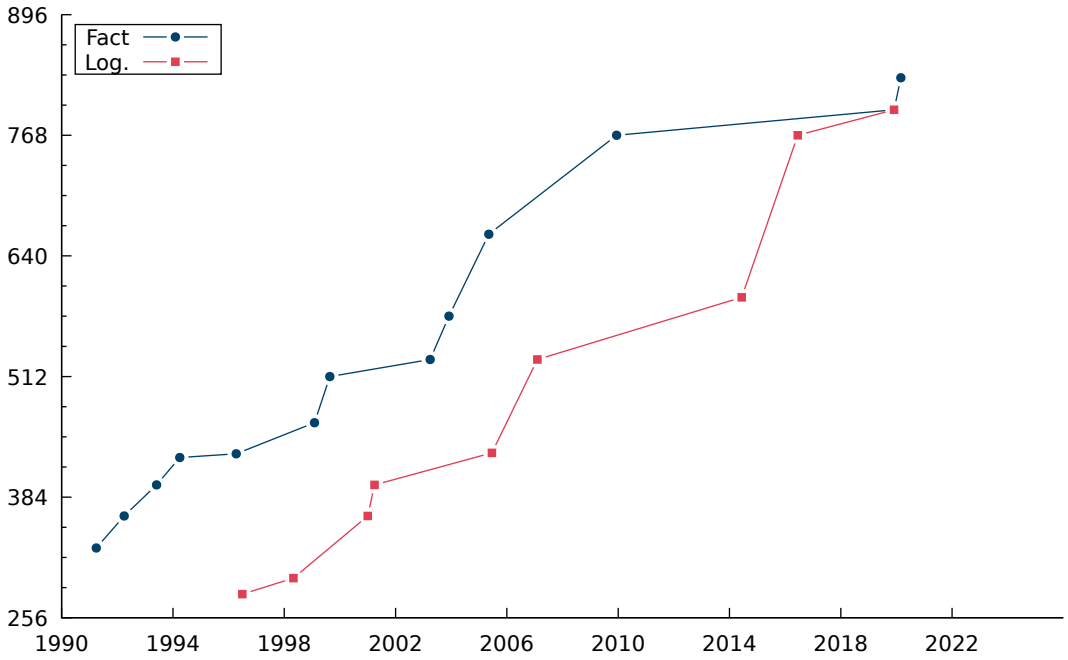


Figure 6 – Records de factorisation d'entiers de type RSA et de logarithme discret (modulo p). L'axe des abscisses donne les années des records, et l'axe des ordonnées les tailles des records en bits.

L'algorithme SNFS (crible algébrique dit « spécial ») a ainsi permis de factoriser le nombre de Mersenne $2^{809} - 1$ de **809** bits (244 chiffres décimaux) début 2003, puis le nombre de Mersenne $2^{1039} - 1$ de **1039** bits (313 chiffres décimaux) en 2007, un nombre qui est donc plus grand qu'un module RSA de 1024 bits.

B.3 Records de calcul de logarithme discret dans GF(p)

Les principaux records successifs de calcul de logarithme discret dans un corps fini premier à p éléments $\mathrm{GF}(p)$ sont résumés dans la table 4.

Table 4 – Records de logarithme discret multiplicatif modulo p

Date	Bits	Chiffres	Auteurs
1996-11-25	**281**	85	Weber, Denny et Zayer
1998-05-26	**298**	90	Joux et Lercier
1999-11	**331**	100	Joux et Lercier
2001-01-19	**364**	110	Joux et Lercier
2001-04-17	**397**	120	Joux et Lercier
2005-06-18	**431**	130	Joux et Lercier
2007-02-05	**530**	160	Bahr, Franke et Kleinjung
2010-05-08	**596**	180	Thomé et al.
2016-06-16	**768**	232	Kleinjung et al.
2019-12-02	**795**	240	Thomé et al.

B.4 Calcul de logarithme discret sur courbe elliptique

La société Certicom a publié le 6 novembre 1997 une liste de challenges [23] concernant le problème du logarithme discret sur courbe elliptique. Les challenges sont de trois types : courbe elliptique « quelconque » définie sur $\mathrm{GF}(p)$, courbe elliptique « quelconque » définie sur $\mathrm{GF}(2^n)$, et courbe de Koblitz également définie sur $\mathrm{GF}(2^n)$. Ces challenges sont respectivement désignés par les codes ECCp-*x*, ECC2-*x* et ECC2K-*x*, où *x* désigne la taille en bits de l'ordre premier du sous-groupe dans lequel sont définies les opérations.

La table 5 reproduit les résultats annoncés.

Table 5 – Records de logarithme discret sur courbes elliptiques.

Nom	Date	Auteurs
ECCp-79	1997-12-06	Harley et Baisley
ECCp-89	1998-01-12	Harley et al.
ECCp-97	1998-03-18	Harley et al.
ECCp-109	2002-11-06	Monico et al.
secp112r1	2009-07	Bos, Kaihara, Kleinjung, Lenstra, Montgomery
ECCp-131		*Non résolu*
ECC2-79	1997-12-16	Harley et al.
ECC2-89	1998-02-09	Harley et al.
ECC2-97	1999-09-22	Harley et al.
ECC2-109	2004-04-15	Monico et al.
sect113r1	2015-02-27	Wenger et Wolfger
ECC2k-95	1998-05-21	Harley et al.
ECC2k-108	2000-04-04	Harley et al.
Koblitz, 113 bits	2014-05-27	Wenger et Wolfger

23. Voir http://www.certicom.com.

B.5 Records de calcul de vecteurs courts dans un réseau aléatoire

L'université TU Darmstadt maintient depuis 2010 une liste de challenges [24] concernant les problèmes de réseaux euclidiens, en particulier le problème du plus court vecteur (SVP). Certains résultats récents sont résumés dans la table 6. Cette liste de records contient des réseaux générés par une graine et une procédure imposées par l'université et des vecteurs dont la norme est inférieure à $1.05\times$ la norme du plus court vecteur du réseau, celle-ci étant estimée heuristiquement. La date correspond à celle du dernier record dans cette dimension, c'est à dire que le vecteur trouvé à cette date est plus court que celui du précédent record.

Table 6 – Records de réduction de réseaux.

Dimension	Date	Auteurs	Norme
186	2023-07-25	Wang et al.	3484
187	2026-01-20	Bai et al.	3582
188	2025-03-12	Wang et al.	3582
190	2024-07-21	Sun et Chang	3613
200	2025-03-04	Zhao et Ding	3723
210	2026-01-01	Ding et Zhao	3808

24. Voir https://www.latticechallenge.org/svp-challenge.

Annexe C
Bibliographie

[1] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens.
The General Sieve Kernel and New Records in Lattice Reduction.
In Yuval Ishai and Vincent Rijmen, editors, EUROCRYPT 2019, Part II, volume 11477 of LNCS, pages 717–746. Springer, Cham, mai 2019.

[2] ANSSI.
Référentiel Général de Sécurité, Annexe B2, Gestion des clés cryptographiques, 2012.

[3] ANSSI.
Guide de sélection d'algorithmes cryptographiques.
ANSSI, 1.0 edition, 2021.

[4] ANSSI.
Recommandations relatives à l'authentification multifacteur et aux mots de passe.
ANSSI, 2.0 edition, 2021.

[5] Andrew Ayer.
Duplicate Signature Key Selection Attack in Let's Encrypt, 2015.

[6] Mihir Bellare and Philipp Rogaway.
Optimal Asymmetric Encryption.
In Proceedings of Eurocrypt 1994, volume 839 of LNCS, pages 92–111. Springer-Verlag, 1994.

[7] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila.
Multi-Ciphersuite Security of the Secure Shell (SSH) Protocol.
In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, ACM CCS 2014, pages 369–381. ACM Press, novembre 2014.

[8] Daniel Bleichenbacher.
Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1.
In Proceedings of Crypto 1998, volume 1462 of LNCS, pages 1–12. Springer, 1998.

[9] Daniel Bleichenbacher.
Crypto Rump Session, 2006.

[10] Functionality Classes and Evaluation Methodology for Physical Random Number Generators.
BSI.
Technical Report AIS31 Version 4, 2025.

[11] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich.
Reduction from Sparse LPN to LPN, Dual Attack 3.0.
In Marc Joye and Gregor Leander, editors, EUROCRYPT 2024, Part VII, volume 14657 of LNCS, pages 286–315. Springer, Cham, mai 2024.

[12] André Chailloux and Johanna Loyer.
Lattice Sieving via Quantum Random Walks.
In Mehdi Tibouchi and Huaxiong Wang, editors, ASIACRYPT 2021, Part IV, volume 13093 of LNCS, pages 63–91. Springer, Cham, décembre 2021.

[13] Don Coppersmith.
Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities.
Journal of Cryptology, 10(4) :233–260, septembre 1997.

[14] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson.
BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures.
In 2021 IEEE Symposium on Security and Privacy, pages 1696–1714. IEEE Computer Society Press, mai 2021.

[15] Whitfield Diffie and Martin Hellman.
New directions in cryptography.
IEEE Transactions on Information Theory, 22(6) :644–654, 1976.

[16] Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck.
Hide-and-Seek and the Non-resignability of the BUFF Transform.
In Elette Boyle and Mohammad Mahmoody, editors, TCC 2024, Part III, volume 15366 of LNCS, pages 347–370. Springer, Cham, décembre 2024.

[17] Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck.
On the (In)Security of the BUFF Transform.
In Leonid Reyzin and Douglas Stebila, editors, CRYPTO 2024, Part I, volume 14920 of LNCS, pages 246–275. Springer, Cham, août 2024.

[18] Léo Ducas.
Shortest Vector from Lattice Sieving : A Few Dimensions for Free.
In Jesper Buus Nielsen and Vincent Rijmen, editors, EUROCRYPT 2018, Part I, volume 10820 of LNCS, pages 125–145. Springer, Cham, avril / mai 2018.

[19] Léo Ducas and Ludo N. Pulles.
Does the Dual-Sieve Attack on Learning with Errors Even Work ?
In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part III, volume 14083 of LNCS, pages 37–69. Springer, Cham, août 2023.

[20] Eiichiro Fujisaki and Tatsuaki Okamoto.
Secure Integration of Asymmetric and Symmetric Encryption Schemes.
In Michael J. Wiener, editor, CRYPTO'99, volume 1666 of LNCS, pages 537–554. Springer, Berlin, Heidelberg, août 1999.

[21] Eiichiro Fujisaki and Tatsuaki Okamoto.
Secure Integration of Asymmetric and Symmetric Encryption Schemes.
Journal of Cryptology, 26(1) :80–101, janvier 2013.

[22] Shafi Goldwasser and Silvio Micali.
Probabilistic Encryption.
Journal of Computer and System Sciences, 28(2) :270–299, 1984.

[23] Lov K. Grover.
A Fast Quantum Mechanical Algorithm for Database Search.
In 28th ACM STOC, pages 212–219. ACM Press, mai 1996.

[24] Secrétariat général de la défense nationale.
Fournitures nécessaires à l'analyse de mécanismes cryptographiques, 2006.

[25] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse.
Seems Legit : Automated Analysis of Subtle Attacks on Protocols that Use Signatures.
In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, ACM CCS 2019, pages 2165–2180. ACM Press, novembre 2019.

[26] Evan Klitzke.
Bitcoin Transaction Malleability, 2017.

[27] Adeline Langlois and Damien Stehlé.
Worst-case to average-case reductions for module lattices.
DCC, 75(3) :565–599, 2015.

[28] MATZOV.
Report on the Security of LWE : Improved Dual Lattice Attack, avril 2022.

[29] Recommandation for Random Number Generation Using Deterministic Random Bit Generators.
NIST.
Technical Report Special Publication 800-90A, Revision 1, U.S. Department of Commerce, Washington, D.C., 2015.

[30] Oded Regev.
On lattices, learning with errors, random linear codes, and cryptography.
In Harold N. Gabow and Ronald Fagin, editors, 37th ACM STOC, pages 84–93. ACM Press, mai 2005.

[31] C.P. Schnorr.
A hierarchy of polynomial time lattice basis reduction algorithms.
Theoretical Computer Science, 53(2), 1987.

[32] Peter W. Shor.
Algorithms for Quantum Computation : Discrete Logarithms and Factoring.
In 35th FOCS, pages 124–134. IEEE Computer Society Press, novembre 1994.

[33] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov.
The First Collision for Full SHA-1.
In Jonathan Katz and Hovav Shacham, editors, CRYPTO 2017, Part I, volume 10401 of LNCS, pages 570–596. Springer, Cham, août 2017.

Liste des règles

Liste des recommandations

Liste des tables

Liste des figures

Version 3.00 - 2026-03-20 - ANSSI-PG-083

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

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

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité