GUIDE DES MÉCANISMES CRYPTOGRAPHIQUES

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

ANSSI-PG-083
2020-01-01

PUBLIC VISÉ :

Développeur | Administrateur | RSSI | DSI | Utilisateur

Historique des versions

Date	Version	Notes
2004-11-19	1.02	2791/SGDN/DCSSI/SDS/Crypto Première version applicable.

2006-12-19	1.10	2741/SGDN/DCSSI/SDS/LCR
		Mise à jour planifiée.
		Principales modifications apportées :
		• prise en compte de la création du référentiel « gestion de clés » ;
		• indication du document décrivant les fournitures nécessaires à l'évaluation des mécanismes cryptographiques ;
		• prise en compte des évolutions de l'état de l'art dans les domaines suivants :
		  • algorithmes de chiffrement par flot ;
		  • modes opératoires de chiffrement ;
		  • problèmes mathématiques asymétriques ;
		  • génération d'aléa.
		• rajout d'une mention concernant le statut de SHA-1.

2010-01-26	1.20	Intégration dans le Référentiel général de sécurité (en tant qu'annexe B1). Mise à jour planifiée. Principales modifications apportées : • ajout ou modification de règles et de recommandations sur la primalité des ordres (RecommandationLogp-2, RecommandationECp-1, RecommandationEC2-1) ; • modification des règles et des recommandations concernant l'architecture et le retraitement pour la génération d'aléa cryptographique (section 2.4) ; • ajout de règles concernant les algorithmes de chiffrement symétriques à l'horizon 2020 .
2012-06-18	2.00	Mise à jour planifiée. Principales modifications apportées : • harmonisation du chapitre 1 avec la partie correspondante de l'annexe B2 ; • précisions concernant les mécanismes symétriques (section 2.1) et mention de HMAC-SHA-2 ; • précisions et exemples supplémentaires concernant les mécanismes asymétriques (section 2.3) ; • modification des règles et recommandations concernant la taille des clés en cryptographie asymétrique pour une utilisation à long terme (sous-section 2.3) ; • mention de la courbe elliptique FRP256v1 parue au journal officiel et du protocole ECKCDSA ; • ajouts relatifs à l'établissement de clé (sections 2.3.4 et A.2.3) ; • ajout d'une règle (RègleArchiGDA-4) concernant la génération d'aléa.

2012-10-17	2.01	Principale modification apportée : • précisions concernant l'utilisation d'une mémoire non volatile dans la génération d'aléa (introduction et figures de la section 2.4).
2013-03-29	2.02	Principales modifications apportées : • modification des règles et recommandations concernant le problème du logarithme discret (section 2.3.1) ; • mise à jour des records concernant le problème du logarithme discret (sections B.1.4 et B.1.5) ; • précisions concernant l'initialisation de l'état interne d'un générateur d'aléa (règle RègleArchiGDA-4).
2014-02-21	2.03	Principales modifications apportées : • précisions concernant l'architecture d'un générateur d'aléa et le retraitement algorithmique (RègleArchiGDA-3 et RecommandationArchiGDA-1) ; • mise à jour des records concernant le problème du logarithme discret (sections B.1.4 et B.1.5).
2020-01-01	2.04	• ajout d'un paragraphe sur les mécanismes asymétriques post-quantiques ; • modification des règles et recommandations concernant les tailles de clé à l'horizon 2020 ; • mise à jour des records concernant le problème de la factorisation (section B.1.2). logarithme discret (sections B.1.4 et B.1.5).

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**.

Suite aux développements majeurs qui ont eut lieu au cours des trois dernières décennies, la science cryptographique, bien qu'encore jeune, 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 ;
- les règles et recommandations concernant l'authentification ;
- la recommandation de mécanismes cryptographiques précis permettant d'atteindre les différents niveaux de robustesse cryptographique définis dans ce document, bien que certaines primitives très classiques soient mentionnées ;
- les aspects liés à l'implantation des mécanismes et en particulier au choix du support ainsi qu'à la sécurité de l'implantation face aux attaques par canaux auxiliaires (timing attack, simple power analysis [SPA], differential power analysis [DPA], higher order differential power analysis [HO-DPA], electromagnetic power analysis [EMA]…) ou par injection de faute (differential fault analysis [DFA]) ;
- 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 intitulé « Fournitures nécessaires à l'analyse de mécanismes cryptographiques » [1] ;
- les mécanismes non cryptographiques assurant cependant des fonctions de sécurité tels que l'emploi de mots de passe, l'usage de la biométrie… Ces mécanismes sont exclus du champ d'application de ce document car ils ne peuvent être analysés au moyen de méthodes cryptographiques usuelles. Ceci ne remet cependant pas en cause leur intérêt éventuel dans certaines applications.

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 généralement nécessaire à la reconnaissance du bon niveau de sécurité du mécanisme. 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 correspondent généralement à l'état de l'art d'un point de vue technique, ce qui assure un gain considérable en termes 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 du produit.

Note

Il importe de noter dès à présent que les règles et recommandations contenues dans ce document ne constituent pas un dogme imposé aux concepteurs de produits utilisant des mécanismes cryptographiques. L'objectif est de contribuer à une amélioration constante de la qualité des produits de sécurité. À ce titre, le suivi des règles énoncées dans ce document doit être considéré comme une démarche saine permettant de se prémunir contre de nombreuses erreurs de conception ainsi que contre d'éventuelles faiblesses non décelées lors de l'évaluation des mécanismes cryptographiques.

Dans un souci de transparence, les règles et recommandations contenues dans ce document sont le plus souvent accompagnées de justifications. Le but est de convaincre que les choix ne sont pas faits de manière arbitraire mais au contraire 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.

La définition des règles et des recommandations prend également en compte certaines hypothèses

1. Actuellement, version 1.2, nº 2336/SGDN/DCSSI/SDS du 6 novembre 2006.

classiques telles que la loi de Moore 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 par exemple 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 bien entendu 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…) ; cette approche est parfois indispensable mais ne peut être généralisée à cause des contraintes qu'elle impose. Par conséquent, une analyse qui semble valable à plus de 15 ans a été développée. Les résultats présentés doivent cependant être pris avec précaution. Il suffit pour s'en convaincre de comparer l'état de l'art actuel à celui d'il y a quelques dizaines d'années ; aucun mécanisme utilisé aujourd'hui n'a plus de cinquante ans, le plus ancien étant certainement le DES, standardisé en 1977.

1.5 Organisation du document

Ce document est organisé de la manière suivante :

- l'ensemble des règles et recommandations sont regroupées dans le chapitre 2 ; elles sont repérées selon la codification suivante : les premières lettres (Règle ou Recom) indiquent si l'on a affaire à une règle ou une recommandation, le domaine d'application est ensuite précisé et, finalement, un chiffre permet de distinguer les règles d'une même catégorie. Par exemple, RègleFact-3 désigne la règle numéro 3 concernant le problème de la factorisation ;
- 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 aucun tableau récapitulatif 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 serait une grave 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 maintenir à jour régulièrement. Une révision tous les deux ans

semble à la fois réaliste et suffisante. La collecte de commentaires et la diffusion des révisions sont effectuées par le laboratoire de cryptographie de l'ANSSI [2].

2. Pour toute correspondance, utiliser l'adresse courrier ou e-mail disponible sur le site http://www.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.

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, c'est-à-dire le nombre de bits réellement variables[1]. 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 on considère que les clés DES ont une taille de 56 bits.

Les tailles minimales définies ci-dessous n'ont de valeur que sous l'hypothèse que la meilleure attaque pratique 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 du mécanisme est en particulier indispensable.

RÈGLES ET RECOMMANDATIONS :

RègleCléSym

1. Pour les clés symétriques utilisées jusqu'à la fin de l'année 2025, la taille minimale est de 112 bits.
2. La taille minimale des clés symétriques devant être utilisées à partir de 2026 est de 128 bits.

RecommandationCléSym

La taille minimale recommandée des clés symétriques est de 128 bits.

1. Formellement, la taille de la clé est définie en fonction de l'espace des clés possibles et de la probabilité de choix de chacune des clés en utilisant le concept d'entropie. En pratique, une approche aussi complexe est généralement inutile, l'idée intuitive de « taille de clé effective » étant évidente.

Justification

- L'estimation de la capacité de calcul que peut rassembler une organisation motivée fait l'objet de beaucoup de controverses. De nombreux indices (voir A.1.1, B.1.1 et B.2.1) indiquent cependant que l'emploi de clé de moins de 112 bits semble risqué. Les clés de 56 bits sont clairement insuffisantes et la capacité actuelle à attaquer des clés de 64 bits est aujourd'hui admise, même si un tel calcul n'est pas à la portée de n'importe qui. De telles attaques ont cependant déjà été menées concrètement dans le milieu public (voir B.1.1).
- Une recherche exhaustive sur des clés de 112 bits demeure difficilement concevable avant plusieurs dizaines d'années. L'emploi de clés de moins de 128 bits devrait cependant tendre à disparaître avec l'emploi d'algorithmes modernes tels que l'AES.

Remarque

- L'impact en termes de performances de l'emploi de clés d'au moins 128 bits est souvent faible, comme le montre l'exemple de l'AES.
- L'emploi de clés de 128 bits permet de s'assurer que les attaques génériques par recherche exhaustive seront inopérantes, y compris à assez long terme. Ceci ne veut bien entendu pas dire que tout mécanisme utilisant de telles clés est cryptographiquement sûr.
- L'emploi de clés de 112 bits, comme dans le cas du triple DES, ne pose pas de problème pratique de sécurité vis-à-vis d'attaques par recherche exhaustive. L'utilisation du triple DES peut cependant être déconseillée pour d'autres raisons, en particulier liées à la taille du bloc (64 bits) insuffisante pour assurer une sécurité pratique avec certains modes opératoires classiques.

2.1.2 Chiffrement symétrique

2.1.2.1 Chiffrement par bloc

Les deux caractéristiques les plus simples d'un mécanisme de chiffrement par bloc sont la taille effective de la clé ainsi que la taille des blocs traités (voir A.1.1). Les règles et recommandations concernant la taille effective de la clé ont été présentées au paragraphe précédent.

Taille de bloc.

RÈGLES ET RECOMMANDATIONS :

RègleBlocSym

1. Pour une utilisation jusqu'à la fin de l'année 2025, la taille minimale des blocs des mécanismes de chiffrement par bloc est de 64 bits.
2. À partir de l'année 2026, la taille minimale des blocs des mécanismes de chiffrement par bloc est de 128 bits.

3. Le nombre maximal de blocs chiffrés sous une même clé ne doit pas dépasser $2^{n/2-5}$, où n est la taille d'un bloc (en bits).

RecommandationBlocSym

1. La taille recommandée des blocs de mécanismes de chiffrement par bloc est de 128 bits.

Justification

- L'emploi de blocs de taille trop petite rend des attaques élémentaires comme la constitution de dictionnaires plus efficaces en pratique que la recherche de la clé secrète. Il est communément admis que la taille d'un bloc doit être d'au moins 64 bits.
- Les mécanismes de chiffrement par bloc sont utilisés via des modes opératoires permettant de chiffrer des messages de taille quelconque ou bien de calculer des codes d'authentification de message. La taille du bloc intervient alors dans l'estimation de la sécurité de ces mécanismes. La principale menace est la découverte, fréquente, d'attaques dites « en racine carrée », fondées sur le paradoxe des anniversaires (voir A.1) ; ceci signifie que certaines attaques ont 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. Dans le cas de blocs de 64 bits, cette limite de sécurité est donc seulement de l'ordre de la centaine de millions de blocs, ce qui peut être très rapidement atteint pour certaines applications. Une manière simple de se prémunir en pratique contre de telles attaques est d'utiliser des blocs de 128 bits.
- L'emploi de blocs de moins de 128 bits devrait tendre à disparaître avec l'emploi d'algorithmes modernes tels que l'AES.

Choix de l'algorithme. Le choix d'un algorithme 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 bien entendu prendre en compte la sécurité intrinsèque apportée par le mécanisme face à des attaques plus évoluées que la simple recherche exhaustive sur la clé (cryptanalyse linéaire, différentielle…).

Considérons une attaque sur un algorithme de chiffrement par bloc. Une telle attaque a un but et nécessite des moyens. Le but peut être de retrouver la clé ou, plus modestement, de distinguer le chiffrement par bloc d'une permutation aléatoire. Les moyens consistent par exemple à permettre à l'attaquant d'observer des chiffrés, les clairs correspondants étant connus ou pas, ou bien de lui permettre de faire chiffrer des messages de son choix, voire même de faire déchiffrer des chiffrés qu'il choisit.

Afin de simplifier, on considère généralement qu'une attaque est qualifiée par :

- le nombre $\mathtt{N}_{\mathtt{op}}$ d'opérations de calcul nécessaires à l'attaque, une opération étant équivalente au chiffrement d'un bloc ;

- le nombre $\mathtt{N_{bloc}}$ de blocs à faire chiffrer ou déchiffrer afin de réaliser l'attaque ;
- la quantité $\mathtt{N_{mem}}$ de mémoire nécessaire, par exemple pour stocker des précalculs.

Règles et recommandations :

RègleAlgoBloc

1. Pour un algorithme de chiffrement par bloc ne devant pas être utilisé après la fin de l'année 2025, aucune attaque nécessitant moins de $\mathtt{N_{op}} = 2^{100}$ opérations de calcul ne doit être connue.
2. Pour un algorithme de chiffrement par bloc dont l'utilisation au-delà du 1er janvier 2026 est envisagée, aucune attaque nécessitant moins de $\mathtt{N_{op}} = 2^{125}$ opérations de calcul ne doit être connue.

RecommandationAlgoBloc

1. Il est recommandé d'employer des algorithmes de chiffrement par bloc largement éprouvés dans le milieu académique.

Remarque

- Les règles ne font pas mention du nombre de blocs $\mathtt{N_{bloc}}$ à faire chiffrer ou déchiffrer afin de réaliser l'attaque, ni de la quantité de mémoire $\mathtt{N_{mem}}$ nécessaire. Ceci tient essentiellement à la volonté de ne pas trop compliquer l'énoncé de ces règles. Il conviendra, au cas par cas, de juger si l'un de ces deux paramètres est suffisamment important afin de justifier qu'un mécanisme de chiffrement par bloc est sûr même s'il ne vérifie pas les règles RègleAlgoBloc-1 et/ou RègleAlgoBloc-2.

Justification

- Les règles tentent de définir les attaques que l'on qualifie habituellement de pratiques, opérationnelles ou réalistes, bien que ces termes prennent souvent des sens très différents selon qui les emploie.
- L'aspect pratique des attaques est privilégié. Par contre, il va de soi que l'existence d'attaques plus « théoriques », même si elles ne mènent pas directement à des attaques opérationnelles, sont une preuve d'existence de faiblesses intrinsèques au mécanisme.
- L'emploi d'algorithmes largement étudiés par la communauté académique offre un gage de qualité très important.

Mécanisme conforme

- L'AES, tel qu'il est spécifié dans le FIPS 197, est un mécanisme de chiffrement par bloc conforme au référentiel.

Remarque

- Le triple DES, c'est-à-dire l'utilisation du DES avec deux clés K_1 et K_2 en chiffrant avec K_1, déchiffrant avec K_2 et chiffrant de nouveau avec K_1, est un algorithme de chiffrement par bloc utilisant des clés de 112 bits et des blocs de 64 bits. Le triple DES respecte donc les règles RègleCléSym-1 et RègleBlocSym-1 imposant des tailles de clés et de bloc minimales lorsqu'une utilisation à partir de 2026 n'est pas envisagée. Cependant, ce mécanisme utilise une taille de bloc inférieure à la taille préconisée dans la recommandation RecommandationBlocSym-1. Il convient donc d'être extrêmement prudent lorsqu'on utilise ce mécanisme avec un mode opératoire de chiffrement, d'intégrité ou bien dans des protocoles de transport de clé par exemple, en particulier à cause de la faible taille du bloc. De plus, le triple DES avec deux clés est vulnérable à une attaque à clair connu. La complexité de cette attaque est de 2^{80} pour 2^{40} couples clairs-chiffrés connus et de 2^{100} pour 2^{20} couples clairs-chiffrés connus. Selon le cadre d'emploi, l'utilisation du triple DES avec deux clés peut ne pas être conforme au référentiel. En particulier, le contexte d'emploi ne doit pas permettre le chiffrement avec une même clé de plus de 2^{20} blocs de message connus d'un attaquant.

Mode opératoire pour le chiffrement. Le mode opératoire pour le chiffrement permet d'assurer la confidentialité de messages de taille quelconque à partir d'une primitive de chiffrement par bloc. Comme expliqué en A.1.1, un simple mécanisme 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.

RÈGLES ET RECOMMANDATIONS :

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.

RègleModeChiff

1. Au sein du modèle de sécurité correspondant à l'usage du mode de chiffrement, il ne doit exister aucune attaque de complexité inférieure à $2^{n/2}$ appels de la primitive où n est la taille en bits du bloc.

RecommandationModeChiff

1. L'emploi d'un mode opératoire de chiffrement non déterministe est recommandé.
2. L'utilisation d'un mode opératoire de chiffrement se fera de préférence conjointement à l'utilisation d'un mécanisme d'intégrité. Un tel mécanisme pourra être indépendant du mode de chiffrement.
3. On utilisera de préférence des modes opératoires disposant d'une preuve de sécurité.

Justification

- De nombreux modes, tels que le CBC (voir A.1.1), 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. Pour un mécanisme de chiffrement utilisant des blocs de 64 bits, cette limite peut être rapidement atteinte (32 Go).
- L'aspect pratique des attaques est privilégié. Par contre, il va de soi que l'existence d'attaques plus « théoriques », même si elles ne conduisent pas directement à des attaques opérationnelles, sont une preuve d'existence de faiblesses intrinsèques au mécanisme.
- 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 de « valeurs initiales » et d'un mode opératoire adapté (voir A.1.1) permet de résoudre ce problème.
- Il est important de prendre en considération les capacités d'un éventuel attaquant à observer des messages chiffrés mais également à obtenir les messages clairs correspondants, à faire chiffrer ou déchiffrer des messages de son choix… Le modèle de sécurité est ici fondamental ; se restreindre au scénario où l'attaquant peut seulement avoir connaissance de messages chiffrés est une grave erreur qui peut avoir de réelles conséquences pratiques sur la sécurité du mécanisme.
- Le besoin de confidentialité est souvent associé à un besoin d'intégrité, même si ce dernier semble parfois moins évident à première vue. Il est fondamental de prendre conscience du fait qu'un mécanisme de chiffrement peut apporter une protection de très haut niveau en termes de confidentialité sans pour autant garantir la moindre intégrité ! En particulier, aucun des modes opératoires classiques (ECB, CBC, OFB, CFB, CTR) n'apporte la moindre protection en intégrité.

Mécanisme conforme

- Le mode de chiffrement CBC utilisant une primitive de chiffrement conforme au référentiel comme l'AES et des valeurs initiales aléatoirement choisies pour chaque message et transmises en clair est un mécanisme de chiffrement symé-

trique conforme au référentiel. Ce mécanisme est rappelé en section A.1.1. Il est particulièrement important de garantir que les valeurs initiales sont générées dans le périmètre de sécurité du chiffrement – par exemple dans le composant sécurisé où le mode de chiffrement et la primitive sous-jacente sont implantés et non hors de ce composant – et avec un générateur d'aléa sûr. Elles ne doivent en aucun cas pouvoir être contrôlées ou prédites par un attaquant.

2.1.2.2 Chiffrement par flot

Les algorithmes de chiffrement par flot [2] constituent l'autre grande famille de mécanismes de chiffrement symétrique (voir A.1.1).

L'algorithme dit de « one-time pad », qui se résume à une addition bit à bit du message à chiffrer avec une clé de même taille, est à part dans la classification des algorithmes de chiffrement. Il ne peut en particulier pas être considéré comme un chiffrement par flot même si ces derniers en dérivent souvent. Cet algorithme dispose d'une sécurité parfaite. Ses contraintes d'emploi sont cependant telles que son utilisation est en pratique impossible, sauf dans des cas très particuliers. Rappelons que ce mécanisme nécessite en particulier d'employer une clé à usage unique aussi longue que le message à protéger, sans aucune possibilité d'une quelconque réutilisation de cette clé. En général, l'emploi du « one-time pad » repousse simplement le problème du chiffrement au niveau de la mise en accord de clé.

Choix de l'algorithme. Avant de définir des règles liées au choix d'algorithmes de chiffrement par flot, il convient de rappeler que ces derniers ne garantissent en général aucune forme d'intégrité des messages transmis (voir remarque ci-dessus pour les modes de chiffrement par bloc). Cependant, pour un algorithme de chiffrement par flot sans rebouclage, le déchiffrement d'un message chiffré non généré par l'algorithme de chiffrement n'apporte aucune information supplémentaire susceptible de favoriser une attaque. Le modèle de sécurité généralement retenu pour l'évaluation de tels mécanismes est la connaissance par l'adversaire du flot de sortie de l'algorithme.

Il convient par ailleurs de préciser qu'il est exclusivement question des algorithmes de chiffrement par flot « dédiés », en ce sens qu'ils ont été conçus spécialement pour cet usage. Les règles ne concernent pas les autres types de générateurs pseudo-aléatoires déterministes (mode flot de chiffrement par bloc, générateurs fondés sur des problèmes difficiles, générateurs informatiquement sûrs).

Note

D'autre part, les règles et recommandations en termes de chiffrement par flot se fondent sur le constat qu'à ce jour la recherche académique dans ce domaine ne semble pas posséder la même maturité que dans le domaine des primitives de chiffrement par bloc. Ces règles sont cependant susceptibles d'être revues si des avancées théoriques importantes sont effectuées dans le domaine du chiffrement par flot.

2. « Stream cipher » en anglais.

Règles et recommandations :

RègleChiffFlot

1. Pour un algorithme de chiffrement par flot ne devant pas être utilisé après la fin de l'année 2025, aucune attaque nécessitant moins de 2^{100} opérations de calcul ne doit être connue.
2. Pour un algorithme de chiffrement par flot devant être utilisé à partir de 2026, aucune attaque nécessitant moins de 2^{125} opérations de calcul ne doit être connue.

RecommandationChiffFlot

1. Il est recommandé d'employer des primitives de chiffrement par bloc et non des algorithmes de chiffrement par flot dédiés. Il est ainsi possible, si les propriétés du chiffrement par flot sont requises, d'utiliser un mode opératoire par flot de chiffrement par bloc conforme au référentiel et simulant un chiffrement par flot.
2. En cas d'utilisation d'un algorithme de chiffrement par flot, il est recommandé d'employer des algorithmes de chiffrement par flot largement éprouvés dans le milieu académique.

Remarque

- Les règles ne font pas mention de la quantité de données à chiffrer ou à déchiffrer afin de réaliser l'attaque, ni de la quantité de mémoire nécessaire. Ceci tient essentiellement à la volonté de ne pas trop compliquer l'énoncé de ces règles. Il conviendra, au cas par cas, de juger si l'un de ces deux paramètres est suffisamment important afin de justifier qu'un mécanisme de chiffrement par flot est sûr même s'il ne vérifie pas les règles RègleChiffFlot-1 et/ou RègleChiffFlot-2.

Justification

- L'aspect pratique des attaques est privilégié. Par contre, il va de soi que l'existence d'attaques plus « théoriques », même si elles ne conduisent pas directement à des attaques opérationnelles, sont une preuve d'existence de faiblesses intrinsèques au mécanisme.
- L'emploi d'algorithmes largement étudiés par la communauté académique offre un gage de qualité très important.
- Les algorithmes de chiffrement par flot dédiés sont parfois préférés aux algorithmes de chiffrement par bloc pour leur efficacité. L'expérience montre cependant que la conception d'algorithmes de chiffrement par flot dédié est très difficile. De nombreuses propositions ont été cryptanalysées et le savoir-faire ne semble pas avoir atteint une maturité comparable à celle observée dans

le domaine du chiffrement par bloc. Par conséquent, l'emploi de chiffrement par bloc, en combinaison avec des modes opératoires satisfaisants, est recommandé, de préférence à des algorithmes de chiffrement par flot.

- Lorsque les propriétés du chiffrement par flot sont néanmoins requises, il est recommandé, si la dégradation en termes de performance le permet, d'utiliser un mode opératoire par flot de chiffrement par bloc, tel que les modes classiques OFB, CFB ou CTR. Il est ainsi possible de tirer parti du savoir-faire acquis en chiffrement par bloc tout en bénéficiant d'un mécanisme de chiffrement par flot.

2.1.3 Authentification et intégrité de messages

Les règles sur les méthodes d'authentification et d'intégrité de messages sont très dépendantes du mécanisme choisi. Certaines règles générales peuvent cependant être émises.

RÈGLES ET RECOMMANDATIONS :

RègleIntegSym

1. Les méthodes symétriques d'intégrité les plus classiques se basent sur des mécanismes de chiffrement par bloc ou de hachage. De telles primitives doivent être conformes au référentiel.
2. Il ne doit pas exister d'attaque sur le mécanisme d'intégrité utilisant moins de $2^{n/2}$ appels à la primitive sous-jacente où n est la taille de sortie de cette primitive.

RecommandationIntegSym

1. On utilisera de préférence des mécanismes disposant d'une preuve de sécurité.

Remarque

- Par confusion avec les modes opératoires de chiffrement, l'emploi de « valeurs initiales » est parfois constaté pour des mécanismes d'intégrité tels que le CBC-MAC [a] ; de graves failles de sécurité peuvent en découler.
- Il est important de prendre en considération les capacités d'un éventuel attaquant à observer des éléments d'intégrité mais également à en obtenir pour des messages de son choix, par exemple.
- De nombreux modes, tels que le CBC-MAC, 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. Pour un mécanisme de chiffrement utilisant des blocs de $n = 64$ bits, cette limite peut être rapidement atteinte.
- **L'emploi de clés de taille importante ne garantit pas nécessairement une**

sécurité en rapport avec cette taille. La plupart des variantes du CBC-MAC construites afin d'obtenir une sécurité comparable à celle du triple DES ont ainsi été cryptanalysées au sens où leur sécurité est plus comparable à celle du DES que du triple DES (c'est le cas de l'exemple de la section A.1.1 si l'on utilise le DES comme algorithme de chiffrement par bloc, même si la taille de la clé est au total de 112 bits).

- 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 algorithme d'intégrité d'obtenir un mécanisme n'assurant plus le service de confidentialité.

Mécanisme conforme

- Le mode d'intégrité CBC-MAC « retail » utilisant l'AES comme mécanisme de chiffrement par bloc et deux clés distinctes (une pour la chaîne CBC et l'autre pour le surchiffrement dit « retail ») est conforme au référentiel (à condition, bien entendu, de ne pas utiliser de valeur initiale). Ce mécanisme est rappelé section A.1.1. Il est à noter que le mode CBC-MAC sans surchiffrement n'est sûr que lorsqu'il est utilisé pour des messages de taille fixe.
- Le mode d'intégrité HMAC utilisant SHA-2 comme fonction de hachage est conforme au référentiel.

Mécanisme non conforme

- Le mode d'intégrité CBC-MAC « retail » recommandé ci-dessus n'est pas conforme au référentiel s'il est utilisé avec le DES comme mécanisme de chiffrement par bloc, et ce même s'il emploie deux clés distinctes. En effet, bien qu'utilisant alors 112 bits de clé, l'observation de 2^{32} MACs valides permet ensuite de retrouver ces 112 bits de clé en effectuant « seulement » de l'ordre de 2^{56} calculs de DES.

2.2 Fonctions de hachage

Les fonctions de hachage cryptographiques doivent avoir plusieurs propriétés telles que la résistance à la recherche de « collisions » (voir A.3). De telles collisions peuvent cependant toujours être trouvées au moyen d'attaques génériques fondées sur le « paradoxe des anniversaires ». Un des buts lors de la conception d'une fonction de hachage est par conséquent de faire en sorte qu'il n'existe pas de meilleure attaque. En pratique, afin de contrer les attaques fondées sur le paradoxe des anniversaires, une empreinte doit être deux fois plus longue qu'une clé symétrique pour atteindre le même niveau de robustesse.

a. Pour des raisons de simplification, CBC-MAC désigne le mode avec surchiffrement également connu sous le nom de CBC-MAC « retail ». (voir section A.1.1)

D'autre part, les fonctions de hachage itératives sont construites autour de fonctions plus élémentaires appelées fonctions de compression. L'existence d'attaques sur ces constituants, attaques qualifiées de « partielles » ci-dessous, n'implique pas nécessairement la possibilité d'attaquer la fonction de hachage en elle-même mais trahit des défauts de conception majeurs.

RÈGLES ET RECOMMANDATIONS :

RègleHash

1. La taille minimale des empreintes produites par une fonction de hachage est de 256 bits.
2. La meilleure attaque connue permettant de trouver des collisions doit nécessiter de l'ordre de $2^{h/2}$ calculs d'empreintes, où h désigne la taille en bits des empreintes.

RecommandationHash

1. L'emploi de fonctions de hachage pour lesquelles des « attaques partielles » sont connues est déconseillé.

Justification

- Les règles en termes de taille d'empreinte sont directement déduites de celles appliquées en cryptographie symétrique.
- Il faut insister sur le fait que l'existence d'attaques partielles, même si elles ne conduisent pas à une attaque sur la fonction de hachage, trahit de graves défauts de conception.
- Le critère de sécurité employé pour juger de la qualité d'une fonction de hachage est l'absence de collisions connues. Dans certaines applications, cette propriété semble trop forte car seul le calcul de « pré-image » doit être irréalisable. Il est cependant considéré ici qu'indépendamment de son contexte d'utilisation, une fonction de hachage ne peut être reconnue conforme au référentiel que si elle présente une résistance suffisante à la recherche de collision. Ceci n'est pas contradictoire avec la possibilité qu'un mécanisme employant une fonction de hachage non conforme au référentiel puisse tout de même être jugé, dans sa globalité, comme atteignant un niveau de sécurité suffisant.

Mécanisme conforme

- Le mécanisme de hachage SHA-256 défini dans le FIPS 180-2 est conforme au référentiel.

Mécanisme non conforme

- Le mécanisme de hachage SHA-1 défini dans le FIPS 180-2 est vulnérable à une attaque en recherche de collision. La complexité de cette attaque est estimée à 2^{63}, et donc inférieure à 2^{80}. Le mécanisme de hachage SHA-1 n'est donc pas conforme au référentiel. Il ne respecte ni RègleHash-1, ni RègleHash-2.

2.3 Cryptographie asymétrique

Les mécanismes de cryptographie asymétrique reposent tous sur des problèmes mathématiques difficiles, généralement issus de la théorie des nombres (voir 2.3.2). L'emploi de tels types de problèmes, difficiles à résoudre pour un attaquant, est par conséquent primordial en termes de sécurité.

2.3.1 Problèmes mathématiques asymétriques

2.3.1.1 Factorisation

Le problème de la factorisation consiste à retrouver la décomposition en facteurs premiers d'un entier donné, obtenu de manière secrète par multiplication de deux nombres premiers, généralement de taille comparable. Un tel nombre composé est classiquement appelé « module ».

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

Règles et recommandations :

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. Les exposants secrets doivent être de même taille que le module.
4. Pour les applications de chiffrement, les exposants publics doivent être strictement supérieurs à $2^{16} = 65536$.

RecommandationFactorisation

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 $2^{16} = 65536$.

3. Il est recommandé que les deux nombres premiers p et q constitutifs du module soient de même taille et choisis aléatoirement uniformément.

Justification

- La taille des modules RSA est un sujet souvent très polémique. L'usage courant fait que l'utilisation de modules de 1024 bits est en général considéré comme suffisant pour garantir une sécurité pratique, même si les analyses effectuées par les plus grands spécialistes du domaine s'accordent sur l'idée que de tels modules n'apportent pas une sécurité suffisante, ou tout du moins comparable à celle que l'on exige des autres mécanismes cryptographiques. L'application d'un paradigme fondamental de la cryptographie, qui consiste à dimensionner les systèmes non pas en se plaçant juste à la limite des capacités d'attaquants connus (voir B.2.1) mais en s'imposant une marge de sécurité, milite pour l'emploi de modules d'au moins 2048 bits, même si aucun module de 1024 bits n'a été officiellement factorisé à ce jour[a]. Par conséquent, l'emploi de modules de 1024 bits est considéré comme une prise de risque incompatible avec des critères de sécurité raisonnables.
- Les paragraphes B.2.1 et B.2.2 fournissent des informations complémentaires sur les analyses liées au problème de la factorisation.
- L'emploi d'exposants secrets particuliers (comme des exposants secrets petits par exemple) afin d'améliorer les performances est à proscrire étant donné les attaques pratiques publiées à ce sujet.
- 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. Plus généralement, pour toute application, l'emploi de tels exposants est déconseillé pour des raisons de sécurité.
- 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 $p-1$ ou $q-1$. Pour éviter ces problèmes, il est recommandé de choisir p et q aléatoirement uniformément parmi les nombres premiers de taille égale à la moitié de la taille du module.

2.3.1.2 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 $\mathsf{GF}(p)$ à p éléments où p est un nombre premier ;
- les groupes des points de courbes elliptiques définies sur $\mathsf{GF}(p)$ où p est un nombre premier ;

a. Un résultat de 2007 annonce la factorisation d'un nombre de 1039 bits. Cependant ce nombre n'est pas de type module RSA.

- les groupes des points de courbes elliptiques définies sur $\mathsf{GF}(2^n)$.

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 $\mathsf{GF}(2^n)$. 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 $\mathsf{GF}(p)$ Le problème dit « du logarithme discret dans $\mathsf{GF}(p)$ » est fondé sur des calculs effectués dans le corps fini à p éléments, où p est un nombre premier également appelé « module ».

Règles et recommandations :

RègleLogp

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.

RecommandationLogp

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 (au lieu d'être multiple d'un nombre premier).

Justification

- Le problème du logarithme discret dans $\mathsf{GF}(p)$ 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'accompagnera 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 B.2.1) 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.

- La raison de recommander un sous-groupe d'ordre premier est que, 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 procédures de chiffrement de type ElGamal.

Logarithme discret dans les courbes elliptiques définies sur $\mathsf{GF}(p)$ Il est également possible de définir un problème de logarithme discret dans des structures plus complexes pour lesquelles aucun algorithme 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 ($\mathsf{GF}(p)$) ou binaire ($\mathsf{GF}(2^n)$).

Règles et recommandations :

RègleECp

On emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 250 bits.

1. 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 $\mathsf{GF}(p)$, ce problème devra vérifier les règles correspondantes.

RecommandationECp

1. Il est recommandé d'employer des sous-groupes dont l'ordre est premier (au lieu d'être multiple d'un nombre premier).

Justification

- Le problème du logarithme discret sur courbe elliptique semble aujourd'hui un problème très difficile. Il permet d'obtenir, pour des tailles de paramètres réduites, une sécurité 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 cryptosystème 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 partie 2.3.1.2 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 petit multiple (dans le

pire cas, 2) d'un grand premier, le problème Diffie–Hellman décisionnel dans ce sous-groupe peut être résolu avec une probabilité non-négligeable.

Mécanisme conforme

- L'emploi de 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. Il en est de même de l'emploi des courbes P-256, P-384 et P-521 définies dans le FIPS 186-4 de 2013, ainsi que des courbes `brainpoolP256r1`, `brainpoolP384r1` et `brainpoolP512r1` définies dans la RFC 5639.

Logarithme discret dans les courbes elliptiques définies sur $\mathsf{GF}(2^n)$

RÈGLES ET RECOMMANDATIONS :

RègleEC2

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 $\mathsf{GF}(2^n)$, ce problème devra vérifier les règles correspondantes.

RecommandationEC2

1. Il est recommandé d'employer des sous-groupes dont l'ordre est premier (au lieu d'être multiple d'un nombre premier).

Justification

- 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 $\mathsf{GF}(p)$ ne sont pas différenciées de celles définies sur $\mathsf{GF}(2^n)$.

Mécanisme conforme

- L'emploi des courbes B-283, B-409 et B-571 définies dans le FIPS 186-4 de 2013 est conforme au référentiel.

2.3.1.3 Autres problèmes et cryptographie « post-quantique »

Plusieurs autres problèmes ont été proposés dans le milieu académique afin de fournir des alternatives aux problèmes dits « classiques » cités ci-dessus, qui ont en commun d'être vulnérables à l'algorithme de Shor si le calcul quantique progresse suffisamment.

Parmi ces familles de problèmes dits « post-quantiques » figurent notamment des problèmes de géométrie des réseaux euclidiens (par exemple le problème LWE et ses variantes), 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 des graphes d'isogénies de courbes elliptiques.

La sécurité de ces problèmes n'a jusqu'à présent pas fait l'objet d'une étude aussi approfondie que celle des problèmes « classiques » que sont le logarithme discret ou la factorisation. De ce fait, les problèmes « post-quantiques » ne sont pas généralement considérés comme directement substituables aux problèmes « classiques » actuellement reconnus. Utilisés de manière combinée avec ceux-ci, ils peuvent cependant apporter une défense en profondeur contre la menace de l'ordinateur post-quantique.

RèglePQNonRégression

La sécurité des cryptosystèmes asymétriques doit reposer sur au moins un problème mathématique largement étudié et reconnu par la communauté académique.

Problèmes 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 étudiés. À condition de satisfaire notamment les conditions des paragraphes 2.3.1.1 et 2.3.1.2, ces problèmes sont conformes à la règle RèglePQNonRégression.
- Certains mécanismes de signature « post-quantiques » basés sur les fonctions de hachage disposent d'une preuve de sécurité par réduction à la sécurité de la fonction de hachage sous-jacente. À condition d'être employés de manière cohérente avec ces preuves de sécurité, ces mécanismes sont donc conformes au référentiel.

Justification

- 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.
- La règle RèglePQNonRégression (y compris combinée avec la recommanda-

tion RecommandationLongTermePQ) peut être satisfaite par l'utilisation de certains modes « hybrides », c'est-à-dire par la combinaison d'une primitive « post-quantique » avec une primitive « classique » (symétrique ou asymétrique fondée sur l'un des problèmes reconnus conformes) d'une façon qui garantisse une sécurité au moins équivalente à chacune des deux primitives. Une telle combinaison a une complexité qui est la somme des parties classique et post-quantique. Le coût de la partie classique d'un mode hybride (en particulier en termes de volume de communications) sera souvent marginal, alors que cette partie joue un rôle central dans l'assurance de sécurité pré-quantique.

RecommandationLongTermePQ

1. Pour une utilisation de mécanismes asymétriques au-delà du 1er janvier 2030, il est recommandé d'utiliser au moins un mécanisme fondé sur un problème pour lequel aucune attaque utilisant l'algorithmique quantique n'est connue.
2. Si la fonctionnalité fournie par le mécanisme asymétrique est potentiellement vulnérable à une attaque rétroactive, il est recommandé d'utiliser au moins un mécanisme fondé sur un problème pour lequel aucune attaque utilisant l'algorithmique quantique n'est connue.

Justification

- Les primitives d'échange de clé ou de chiffrement asymétrique sont généralement 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 et compromettre la confidentialité des données échangées. Par contraste, les primitives d'authentification sont dans de nombreux cas moins vulnérables aux attaques rétroactives : ainsi, la signature utilisée pour authentifier un établissement de clé n'est vulnérable que jusqu'à ce qu'elle soit vérifiée.
- Une méthode permettant de satisfaire à la fois cette recommandation et la règle RèglePQNonRégression est l'utilisation d'un cryptosystème « hybride », ajoutant à la sécurité d'un mécanisme asymétrique « classique » une protection supplémentaire contre un adversaire capable d'exécuter l'algorithme de Shor.
- Un autre type d'hybridation envisageable est d'associer un mécanisme asymétrique « classique » avec un mécanisme symétrique. La présence d'un secret partagé peut alors permettre d'ajouter une couche de protection contre un adversaire capable d'exécuter l'algorithme de Shor ; elle change en outre la nature du cryptosystème, qui acquiert certaines caractéristiques d'un cryptosystème symétrique. L'architecture de clés d'un tel cryptosystème doit en particulier tenir compte de la recommandation RecommandationGestSym.
- La condition temporelle donnée dans la recommandation RecommandationLongTermePQ 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.

2.3.2 Chiffrement asymétrique

Toute méthode de chiffrement asymétrique s'appuie sur un problème difficile de base. Ce dernier doit donc être en accord avec le niveau de robustesse recherché. Il est de plus possible pour certains mécanismes de chiffrement de faire la preuve, éventuellement sous certaines hypothèses, que la sécurité est équivalente à celle du problème de base et pas uniquement reliée de manière heuristique. Cette approche moderne de la cryptographie permet d'atteindre un niveau d'assurance meilleur que la simple approche qui consiste à constater l'absence d'attaques connues.

RÈGLES ET RECOMMANDATIONS :

RecommandationChiffAsym

1. Il est recommandé d'employer des mécanismes de chiffrement asymétrique disposant d'une preuve de sécurité.

Justification

- L'existence d'une preuve de sécurité apporte des garanties importantes sur la résistance du mécanisme.

Mécanisme conforme

- 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.

Mécanisme non conforme

- 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 messages chiffrés choisis adaptative) exploitant judicieusement un tel oracle pour retrouver le message clair correspondant à un chiffré donné [Ble98].

2.3.3 Signature asymétrique

Toute méthode de signature asymétrique s'appuie sur un problème difficile de base. Ce dernier doit donc être en accord avec le niveau de robustesse recherché. Il est de plus possible pour certains mé-

canismes de signature de faire la preuve, éventuellement sous certaines hypothèses, que la sécurité est équivalente à celle du problème de base et pas uniquement reliée de manière heuristique.

Les schémas de signature utilisent de plus en général des fonctions de hachage dont le niveau de robustesse (voir 2.2) doit bien entendu être en accord avec le niveau de robustesse souhaité pour le mécanisme de signature.

Règles et recommandations :

RecommandationSignAsym

1. Il est recommandé d'employer des mécanismes de signature asymétrique disposant d'une preuve de sécurité.

Mécanisme conforme

- Le mécanisme de signature asymétrique RSA-SSA-PSS[a] 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 asymétrique ECDSA défini dans le FIPS 186-4, ainsi que le mécanisme de signature asymétrique 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 186-4.

Mécanisme non conforme

- Le mécanisme de signature asymétrique RSASSA, mis en œuvre selon le document PKCS#1 v1.5 n'est pas conforme au référentiel lorsque l'exposant public e est petit et pour un mauvais choix d'implantation des vérifications liées au padding. En effet, Bleichenbacher a mis en évidence en 2006 une attaque permettant de forger des signatures dans ce cas [Ble06].

2.3.4 Authentification d'entités et établissement de clé

Comme il est rappelé en A.2.3, 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 2.3.2, 2.3.3, 2.2 et 2.4 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 le schéma d'établissement de clé de Diffie-

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

Hellman ; les règles et recommandations énoncées dans la section 2.3.1 s'appliquent alors aux problèmes mathématiques sur lesquels ces primitives reposent. Bien entendu, 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. Il est également souhaitable qu'un mécanisme d'établissement de clé assure la confidentialité dans le futur ou PFS (de l'anglais *perfect forward privacy*) des clés symétriques temporaires (ou *clés de session*) qu'il permet d'établir. 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 schéma reposant sur l'emploi par les deux entités de secrets éphémères effacés après utilisation. Bien entendu, les clés symétriques temporaires doivent elles-mêmes être effacées par les deux entités à l'échéance de leur période de validité (fin de session).

Les mécanismes utilisant des mots de passe sous une forme quelconque (pass-phrase, code d'identification personnel…) ainsi que les mécanismes s'appuyant sur des procédés biométriques ne sont pas de nature cryptographique et par conséquent ne sont pas traités dans le cadre de ce document. Bien entendu, ceci ne signifie pas qu'ils ne présentent aucun intérêt pour la sécurisation d'un système d'information.

Ce document n'a pas vocation à traiter des mécanismes d'authentification. Rappelons cependant quelques remarques élémentaires liées à l'utilisation de mots de passe :

- Si l'on souhaite dériver des clés secrètes à partir de mots de passe, ces derniers doivent être suffisamment longs et « non devinables » pour offrir une sécurité compatible avec les règles relatives aux tailles de clés. À titre d'exemple, des mots de passe de 8 caractères alphanumériques (chiffres et lettres majuscules ou minuscules) ne permettent pas de générer des clés de plus de 47 bits, et encore sous l'hypothèse très optimiste que ces mots de passe sont choisis aléatoirement.
- Il convient, lorsqu'on étudie la sécurité d'un mécanisme d'authentification, de distinguer les calculs qui peuvent être effectués « off-line », c'est-à-dire sans accès à une ressource telle qu'un serveur, et les opérations qui doivent être réalisées « on-line ». Ainsi un protocole d'authentification par mot de passe ne doit pas permettre de tests de vérifications off-line efficaces. 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.

2.4 Génération d'aléa cryptographique

Comme expliqué au paragraphe A.4, 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 primitives

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 ;
- **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…).

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

Dans le cas d'un ordinateur personnel, l'exemple le plus simple de source manuelle consiste à demander à l'utilisateur d'entrer au clavier une suite suffisamment longue de caractères « aléatoires », puis à en extraire une valeur secrète courte par hachage. Toutefois, de manière générale, il est préférable que l'interaction de l'utilisateur ne soit pas facilement reproductible et fasse intervenir également des valeurs comme les instants de frappe au clavier ou les positions successives de la souris.

Un **retraitement algorithmique** est un mécanisme de nature cryptographique destiné à combiner différentes sources d'aléa et à garantir dans la durée la qualité de l'aléa produit.

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.

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.

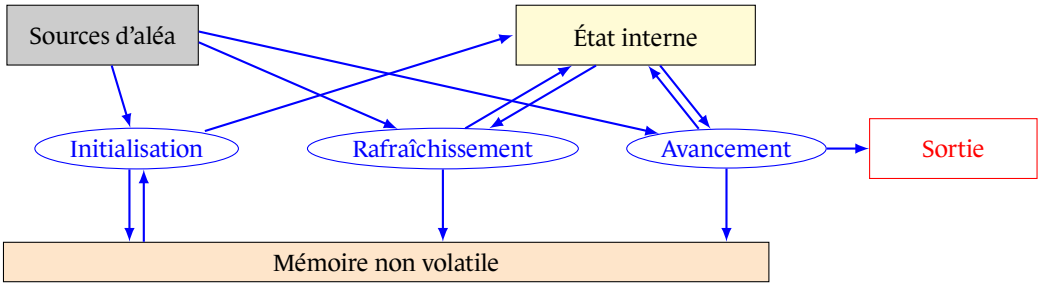

FIGURE 2.1 – Architecture générique pour la génération d'aléa cryptographique

Finalement, les différents éléments constituant un générateur d'aléa cryptographique peuvent être représentés de la manière suivante (figure 2.1), 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.

Note

Les règles et recommandations applicables aux générateurs d'aléa se fondent sur le constat qu'il est aujourd'hui très 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.

2.4.1 Architecture d'un générateur d'aléa

RÈGLES ET RECOMMANDATIONS :

RègleArchiGDA

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. L'état interne doit être au minimum de 128 bits. En l'absence d'un rafraîchissement suffisamment fréquent par un générateur physique d'aléa, cette limite inférieure est portée à 160 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 état une entropie voisine de sa longueur, ou tout au moins supérieure au seuil défini dans la règle RègleArchiGDA-3 si un raisonnement permet d'établir qu'aucune faiblesse n'en résulte.

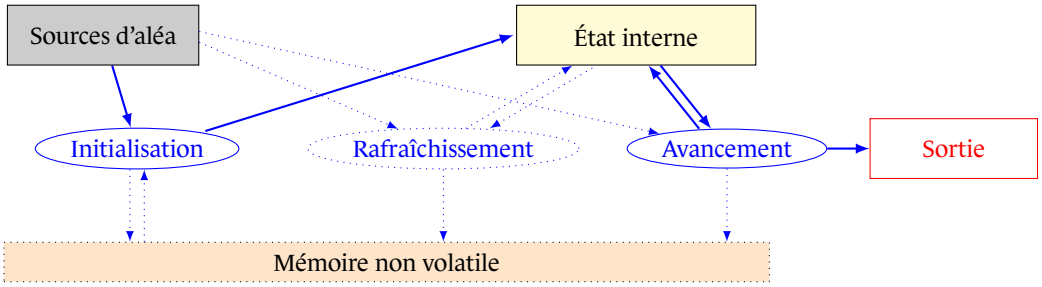

FIGURE 2.2 – Architecture minimale pour la génération d'aléa
(Les pointillés figurent les éléments recommandés.)

RecommandationArchiGDA

1. 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.

Justification

- 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 petite taille.
- 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'implantation aux algorithmes de retraitement que pour tout autre mécanisme cryptographique.

Remarque

- 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 utiliserait des clés secrètes, celles-ci sont considérées comme faisant partie de l'état interne, en particulier pour le calcul de sa taille.

2.4.2 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.

RÈGLES ET RECOMMANDATIONS :

RègleArchiGVA

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

RecommandationArchiGVA

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

Justification

- 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 bien entendu 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 que le générateur d'aléa produit bien de l'aléa vrai.
- 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.

Remarque

- Les tests statistiques concernés par la règle RègleArchiGVA-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.

2.4.3 Retraitement algorithmique

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

Règles et recommandations :

RègleAlgoGDA

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 ni des sources d'aléa (fiables).
3. En cas de compromission « simple » affectant ou bien l'état interne ou bien les sources d'aléa éventuellement présentes mais n'affectant pas simultanément ces deux types d'éléments, la sortie courante ne doit donner à l'attaquant aucune information exploitable sur les sorties passées.

Justification

- Le mécanisme de retraitement employé se doit d'utiliser des primitives cryptographiques (fonction de hachage, chiffrement par bloc…) conformes au référentiel.
- Le but du retraitement est de garantir la qualité cryptographique de l'aléa généré. De plus, comme la détection des pannes physiques est délicate, on souhaite limiter leur impact sur la sécurité du générateur final.

2.5 Gestion de clés

La gestion des clés est un problème à part entière, qui se révèle parfois aussi complexe que la détermination des procédés de chiffrement et d'intégrité. Ce problème n'a pas vocation à être couvert par le document présent. Cependant, afin de donner un premier aperçu du problème de la gestion des clés, quelques règles sont énoncées dans la suite de cette section

Avant de traiter des règles et recommandations relatives à la gestion des clés, rappelons que le recouvrement de clé est un mécanisme à part entière dont le niveau de robustesse doit, à ce titre, être estimé en utilisant les mêmes règles que pour tout autre mécanisme cryptographique.

2.5.1 Clés secrètes symétriques

Les deux principaux risques généraux dans l'emploi de clés secrètes sont, d'une part, l'usage d'une clé pour plusieurs emplois (en confidentialité et intégrité par exemple), et d'autre part l'emploi de clés partagées par un nombre important d'utilisateurs ou d'équipements (voir A.5). De telles clés sont désignées par le terme de « **clés communes** » au sens où elles sont détenues par de nombreux acteurs de même niveau hiérarchique au sein d'un système. Ces clés sont également appelées « **clés de réseau** » dans certaines applications.

Les clés communes ne doivent pas être confondues avec les « **clés maîtres** » utilisées afin de générer des « **clés dérivées** ». Dans ce cas, seules les clés dérivées sont présentes dans les produits et pas les clés maîtres ayant permis leur génération. Ceci est à distinguer du cas des « **clés différenciées** » qui sont générées localement à partir d'une même clé secrète pour être utilisées par des mécanismes distincts.

RÈGLES ET RECOMMANDATIONS :

RègleGestSym

1. L'emploi d'une même clé pour plus d'un usage est exclu
2. Les éventuelles clés différenciées utilisées avec un mécanisme conforme au référentiel doivent être générées en utilisant un mécanisme de diversification conforme au référentiel.
3. Les éventuelles clés dérivées doivent être générées en utilisant un mécanisme de diversification conforme au référentiel.

RecommandationGestSym

1. L'emploi de clés communes est déconseillé.

Justification

- L'emploi d'une même clé à plus d'un usage, par exemple pour chiffrer avec un mécanisme de confidentialité et assurer l'intégrité avec un mécanisme différent, est source de nombreuses erreurs. Ceci n'interdit cependant pas de différencier localement deux clés à partir d'une même clé secrète, à condition que le mécanisme de diversification soit conforme au référentiel.
- La règle RègleGestSym-3 indique la nécessité de dimensionner le système de manière à ce que les cibles privilégiées d'attaque comme les clés maîtres soient suffisamment protégées.
- L'emploi de clés communes est déconseillé car de telles clés sont des cibles pri-

vilégiées d'attaque.

2.5.2 Bi-clés asymétriques

Une infrastructure de gestion de clés est en général construite de manière hiérarchique, chaque clé publique étant certifiée par une clé de rang immédiatement supérieur jusqu'à arriver à une clé racine.

RÈGLES ET RECOMMANDATIONS :

RègleGestAsym

1. L'emploi d'une même bi-clé à plus d'un usage est exclu.
2. Les clés hiérarchiquement importantes, telles que les clés racine, doivent être générées et utilisées par des mécanismes conformes au référentiel.

Justification

- L'emploi d'une même bi-clé à plus d'un usage, par exemple pour chiffrer et signer, est une source d'erreurs graves.
- La règle RègleGestAsym-2 indique la nécessité de dimensionner le système de manière à ce que des cibles privilégiées d'attaque comme les clés racine soient suffisamment robustes.

Annexe A
Définitions et concepts

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. Pour plus de détails, on pourra par exemple consulter les ouvrages de référence suivants : [MvV97], [Sti01], [Vau06] et [CFA+06]. On pourra également trouver de nombreux éléments de réponse dans [Ste98] et [Sch01]. Une approche plus historique est présentée dans [Sin99].

> 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…), 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 primitives 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**, encore qualifiée de **conventionnelle** ou de 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 [DH76] 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 primitives symétriques et asymétriques sont abordées de manière séparée.

Une seconde dimension de classification des primitives cryptographiques concerne l'objectif de sécurité visé. Traditionnellement on distingue les besoins de confidentialité et/ou d'intégrité des données, d'authentification de données ou d'entités ainsi que les besoins de non-répudiation. D'autres fonctions sont bien entendues envisageables mais celles qui viennent d'être citées sont, de loin, les principales traitées par des moyens cryptographiques.

TABLE A.1 – Primitives cryptographiques offrant un service donné

<table>
<tr><th colspan="2">Service</th><th>Cryptographie symétrique</th><th>Cryptographie asymétrique</th></tr>
<tr><td colspan="2" rowspan="2">Confidentialité</td><td rowspan="2">Chiffrement conventionnel par bloc (A.1.1.1)
ou par flot (A.1.1.2)</td><td>Chiffrement à clé publique (A.2.1)</td></tr>
<tr><td>Échange de clé (A.2.3)</td></tr>
<tr><td colspan="2">Intégrité</td><td rowspan="3">Code d'authentification de message (A.1.3)</td><td rowspan="4">Signature numérique (A.2.2)</td></tr>
<tr><td rowspan="2">Authentification</td><td>de données</td></tr>
<tr><td>d'entités</td><td>Défi-réponse (A.1.4)</td></tr>
<tr><td colspan="2">Non-répudiation</td><td>Aucune primitive</td></tr>
</table>

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 bien entendu également l'être par tout autre moyen approprié, à commencer par des mesures organisationnelles non cryptographiques.

Assurer l'**intégrité** de données consiste à empêcher, ou tout du moins à détecter, toute altération non autorisée de données. Par altération, on entend toute modification, suppression partielle ou insertion d'information. L'intégrité des données est classiquement assurée en cryptographie par des codes d'authentification de message ou des mécanismes de signature numérique.

L'**authentification de données** vise à s'assurer qu'elles proviennent bien d'un interlocuteur particulier. Une telle authentification implique l'intégrité de ces informations et est assurée par les mécanismes cités ci-dessus. L'**authentification d'entités**, encore désignée sous le terme d'**identification**, 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.

La **non-répudiation** est un service visant à éviter qu'une entité puisse nier avoir effectué une certaine action. Le principal mécanisme de non-répudiation est la signature à clé publique, une signature sur un message devant en général rester valide, même si le signataire change ensuite d'avis.

Les principales primitives offertes par la cryptographie moderne peuvent maintenant être abordées. Le tableau A.1 permet de les répartir en fonction de leur nature symétrique ou asymétrique et de leur objectif de sécurité. Bien entendu, d'autres primitives très intéressantes peuvent être citées mais elles ne s'inscrivent pas naturellement dans un tel tableau. La suite de ce chapitre décrit sommairement ces primitives et rappelle les points essentiels, nécessaires à la compréhension du reste de ce document.

A.1 Cryptographie symétrique

Les primitives cryptographiques symétriques se caractérisent par le fait qu'elles utilisent des clés cryptographiques partagées par plusieurs personnes ou entités. Une **clé secrète symétrique** est donc simplement un élément secret. La sécurité d'un système utilisant de telles clés repose donc notamment sur la protection de ces secrets.

TABLE A.2 – Ordre de grandeur de la valeur de 2^n

n	2^n	Ordre de grandeur
32	2^{32}	Nombre d'hommes sur Terre
46	2^{46}	Distance Terre–Soleil en millimètres
46	2^{46}	Nombre d'opérations effectuées en une journée à raison d'un milliard d'opérations par seconde (1GHz)
55	2^{55}	Nombre d'opérations effectuées en une année à raison d'un milliard d'opérations par seconde (1GHz)
82	2^{82}	Masse de la Terre en kilogrammes
90	2^{90}	Nombre d'opérations effectuées en 15 milliards d'années (âge de l'univers) à raison d'un milliard d'opérations par seconde (1GHz)
155	2^{155}	Nombre de molécules d'eau sur Terre
256	2^{256}	Nombre d'électrons dans l'univers

Une clé secrète est généralement une suite quelconque de bits, c'est-à-dire de 0 et de 1, de taille fixée. Cette taille de clé, notée n ci-après, est déterminante pour la sécurité du système car on peut former exactement 2^n clés de longueur n. Les systèmes 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. La capacité à effectuer 2^n opérations fournit donc une borne inférieure au dimensionnement des systèmes symétriques.

Afin de fixer les idées sur les ordres de grandeur manipulés, et surtout de bien prendre conscience que 2^n est un nombre très rapidement gigantesque lorsque n croît, quelques exemples numériques concrets sont rassemblés dans le tableau A.2.

Ces chiffres montrent simplement que la fonction 2^n croît extrêmement rapidement avec n. La capacité, même en disposant de moyens très importants, de réaliser de l'ordre de 2^{128} calculs apparaît donc très peu plausible de nos jours. Pour ceux qui ont encore des doutes, il apparaît en tout état de cause qu'effectuer 2^{256} calculs est parfaitement impossible et ne le sera jamais. C'est une des rares certitudes que l'on peut avoir en cryptographie. Ceci va en particulier à l'encontre de l'idée reçue selon laquelle tout cryptosystème peut nécessairement être cassé par recherche exhaustive à condition d'y mettre les moyens.

A.1.1 Chiffrement symétrique

Les primitives symétriques les plus connues sont les algorithmes de chiffrement. Ceux-ci permettent, au moyen de clés secrètes connues des seuls émetteurs et récepteurs d'informations, de protéger la confidentialité de ces dernières, même si le canal de communication employé est écouté. Il doit cependant être d'ores et déjà bien clair que ceci laisse ouvert le problème majeur de l'échange initial de la clé secrète entre les correspondants.

Il faut noter que le seul terme admis en français est celui de chiffrement. On entend cependant souvent parler de « cryptage » qui est un anglicisme, voire de « chiffrage », mais ces mots sont

incorrects. L'opération inverse du chiffrement est le déchiffrement. On désigne par « décryptage », ou « décryptement », l'opération qui consiste à retrouver le clair correspondant à un chiffré donné sans connaître la clé secrète, après avoir trouvé une faille dans l'algorithme de chiffrement.

Les algorithmes de chiffrement symétrique permettent également de sécuriser le stockage d'informations, sans intention de les transmettre. On peut ainsi protéger la confidentialité d'informations enregistrées sur des supports potentiellement vulnérables.

La protection en confidentialité d'informations permet cependant uniquement de s'assurer que le contenu de ces informations est inaccessible à un attaquant. Par contre, nulle garantie d'intégrité ou d'authenticité n'est a priori fournie par les méthodes de chiffrement et rien ne permet d'exclure des possibilités d'attaques actives visant à modifier les informations transmises ou stockées. On peut ainsi facilement concevoir des scénarios d'attaques au cours desquels un attaquant peut modifier des communications protégées en confidentialité sans avoir à comprendre précisément ce qui est transmis. Un exemple de cette attaque est détaillé en section A.1.1.2.

Les méthodes de chiffrement symétrique, encore appelé chiffrement conventionnel ou chiffrement 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.

A.1.1.1 Chiffrement par bloc

Une primitive de chiffrement par bloc est un algorithme traitant les données à chiffrer par blocs de taille fixée. On notera k le nombre de bits de ces blocs de données ; typiquement, cette taille vaut 64 ou 128 bits en pratique. Un tel mécanisme permet donc uniquement de combiner une suite de k bits de données avec une clé de n bits afin d'obtenir un bloc de données chiffrées de même taille k que le bloc de données claires.

L'une des principales propriétés attendues d'un mécanisme de chiffrement par bloc est d'être facilement inversible si l'on dispose de la clé secrète de chiffrement. On veut également que, sans informations sur la clé secrète, il soit impossible en pratique de retrouver de l'information sur le message d'origine. Seuls les détenteurs de la clé secrète doivent être capables de transformer des données claires en données chiffrées et, inversement, des données chiffrées en données claires.

> L'un des exemples les plus connus d'algorithme de chiffrement par bloc est le DES (Data Encryption Standard) défini en 1977 par le NIST, institut de normalisation américain, comme standard de chiffrement à usage commercial. Il traite des blocs de $k = 64$ bits au moyen de clés de $n = 56$ bits. La sécurité du DES a fait couler depuis lors beaucoup d'encre. Cet algorithme peut cependant être considéré comme particulièrement bien conçu, la meilleure attaque pratique connue étant la recherche exhaustive sur les clés. La taille de ces clés est cependant sous-dimensionnée, ce qui rend aujourd'hui cette attaque réalisable, au moyen d'une machine dédiée, en quelques heures seulement.
>
> On utilise cependant toujours couramment le DES mais sous la forme du « triple DES », une variante utilisant des clés de 112 bits, inattaquable par recherche exhaustive. L'objectif à moyen terme est cependant de le remplacer par l'AES (Advanced Encryption Standard), sélectionné par le NIST au terme d'une compétition internationale. L'AES est conçu pour traiter des blocs de 128 bits au moyen de clés de 128, 192 ou 256 bits.
>
> Plus généralement, un algorithme de chiffrement par bloc combine des opérations de **substitution**, visant à remplacer des symboles par d'autres afin d'en cacher le sens, avec des opérations de **permutation** échangeant la position des symboles. Ces deux principes sont historiquement très anciens mais demeurent encore valables aujourd'hui, toute primitive de chiffrement par bloc pouvant être vue comme une combinaison intelligente de ces deux opérations.

À ce niveau, il convient de comprendre précisément ce que permet de faire un algorithme de chiffrement par bloc, et surtout ce qu'il ne permet pas. Tout d'abord, un tel algorithme permet

uniquement de traiter des blocs de taille fixe, relativement petite. Par conséquent, afin de chiffrer des messages de taille quelconque, il convient de définir comment le message doit être codé en une suite de blocs de taille fixe. Ceci implique de définir très précisément comment répartir l'information de tels messages en une suite de bits de longueur exactement un multiple de la taille k du bloc élémentaire. On appelle cette opération le « padding » ou le « bourrage ».

De plus, un algorithme de chiffrement par bloc est fondamentalement déterministe : à partir d'un bloc de données et d'une clé secrète, il produit toujours le même bloc de chiffré. Ainsi, un attaquant passif observant deux blocs de chiffré identiques peut immédiatement en déduire que les blocs de message clair correspondants sont identiques, sans pour autant apprendre la moindre information sur ce clair. Dans certaines circonstances, une telle information est cependant suffisante pour attaquer un système.

> À titre d'exemple, imaginons un système bancaire où, afin de vérifier la validité d'un PIN code (« Personal Identification Number ») à quatre chiffres de carte de crédit, ce code est chiffré et envoyé à un central bancaire. Si l'on utilise un simple chiffrement par bloc avec une clé bancaire fixe, à chaque PIN code va correspondre un unique chiffré. On peut dès lors imaginer par exemple qu'un attaquant observant les communications apprendra immédiatement qui a le même PIN code que lui.

Afin de traiter des messages de taille quelconque et d'assurer la confidentialité globale de ces messages, et pas uniquement une confidentialité « par bloc », il convient donc de définir un **mode opératoire** précisant comment convertir le message en une suite de blocs ainsi que le mode de chiffrement de ces blocs afin d'obtenir finalement le message chiffré. Notons que la définition d'un tel mode opératoire n'a nul besoin de tenir compte des détails de l'algorithme de chiffrement par bloc employé ; seul la taille k des blocs est réellement nécessaire. Notons encore qu'afin de rompre le caractère déterministe du chiffrement, il est nécessaire de « randomiser » le processus, c'est-à-dire d'introduire une valeur aléatoire.

> Le mode opératoire le plus connu est le CBC (« cipher-block chaining »). Une des nombreuses variantes fonctionne de la manière suivante : afin de chiffrer un message formé d'une suite de bits, on commence par ajouter à droite un bit valant 1 et autant de 0 que nécessaire afin d'obtenir un nombre total de bits multiple de la taille du bloc. Notons $x_1, x_2, \ldots, x_t$ les t blocs de message clair ainsi obtenus. On choisit ensuite un bloc aléatoire, noté `IV` pour « initial vector », indépendant du message et des précédents chiffrements. Si l'on note $\mathsf{E}_K(x)$ le résultat du chiffrement du bloc x avec la clé K, le chiffré $(c_0, c_1, c_2, \ldots, c_t)$ du message est obtenu en posant $c_0 = \texttt{IV}$ et en calculant successivement $c_i = \mathsf{E}_K(c_{i-1} \oplus x_i)$ pour i allant de 1 à t (l'opérateur « ou-exclusif » noté « $\oplus$ » représente l'addition bit à bit, sans retenue). Graphiquement, on obtient la représentation symbolique de la figure A.1.
>
> Cet exemple de mode opératoire illustre la phase initiale de padding, consistant à compléter le message par un bit valant 1 suivi de bits nuls. Il montre également que l'emploi de données aléatoires, via l'`IV`, permet de rendre le chiffrement non déterministe. Ainsi, le chiffrement du même message deux fois de suite n'a qu'une chance infime d'utiliser le même `IV` et par conséquent l'ensemble du chiffré va différer à cause du mécanisme de rebouclage faisant intervenir le bloc de chiffré $c_i - 1$ lors du chiffrement du bloc de message clair x_i. Notons encore que dans cette variante du mode CBC, l'`IV` est transmis en clair, sans avoir besoin d'être chiffré.
>
> Afin de déchiffrer un message $(c_0, c_1, c_2, \ldots, c_t)$, il suffit de calculer $x_i = c_{i-1} \oplus \mathsf{D}_K(c_i)$ pour i allant de 1 à t, $\mathsf{D}_K(c)$ désignant le déchiffré du bloc c avec la clé secrète K. Il est facile de vérifier que le message ainsi obtenu est bien le message initial. Notons enfin que ce mode a naturellement une propriété d'auto-synchronisation ; si des blocs de chiffré sont perdus en cours de transmission, il suffit d'obtenir deux blocs successifs intacts afin de pouvoir reprendre correctement le déchiffrement.

A.1.1.2 Chiffrement par flot

Les primitives de chiffrement par flot utilisent une approche différente du chiffrement par bloc au sens où elles considèrent généralement le message à chiffrer comme une suite de bits qui sont

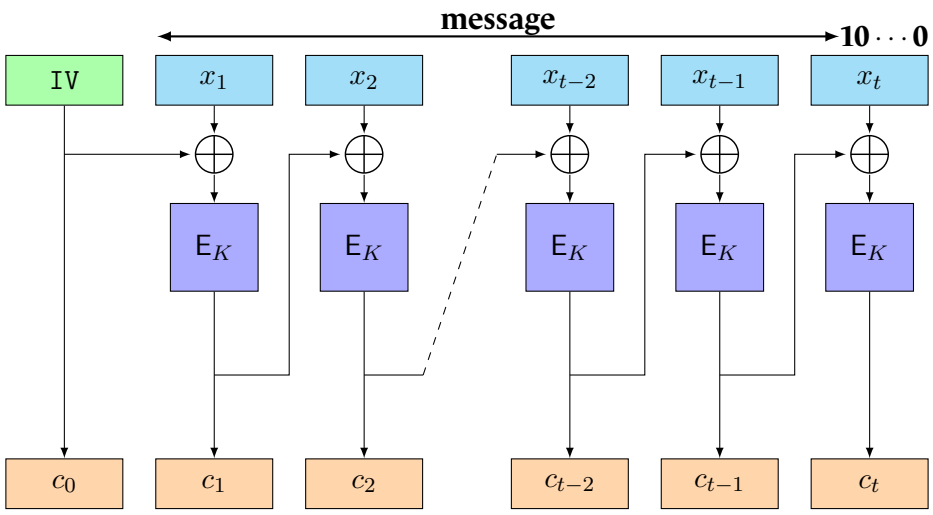

FIGURE A.1 – Mode opératoire CBC

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 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 $m_1, m_2, m_3, \ldots$ ainsi qu'une clé également vue comme une suite de bits $k_1, k_2, k_3, \ldots$, le chiffré du message est alors très simplement obtenu au moyen de l'opération de « ou-exclusif bit à bit » (appelée le XOR), où le i-ème bit de chiffré c_i s'obtient par addition (sans retenue) du i-ème bit de message avec le i-ème bit de clé, soit $c_i = m_i \oplus k_i$. Ainsi $0 \oplus 0 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$ et $1 \oplus 1 = 0$, le dernier cas étant le seul où l'opération XOR diffère de l'addition classique.

Afin de déchiffrer, il suffit d'appliquer la même opération de ou-exclusif bit à bit du chiffré avec la clé secrète car $c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i = m_i \oplus (k_i \oplus k_i) = m_i \oplus 0 = m_i$, en remarquant que, quelle que soit la valeur k_i de chaque bit de clé, $k_i \oplus k_i$ vaut toujours 0.

Il est facile de démontrer que le chiffrement de Vernam est parfaitement sûr, en termes 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 évidemment 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.

Claude Shannon a démontré en 1949 l'impossibilité de réaliser des primitives cryptographiques parfaites reposant sur des clés de taille strictement inférieure à celle du message à chiffrer et par conséquent l'impossibilité de réaliser des primitives parfaites utilisables en pratique. L'approche moderne ne viole pas cette idée mais admet une sécurité imparfaite ; tout l'art du cryptographe est d'estimer ce degré d'imperfection et de concevoir des primitives pour lesquelles il peut être rendu négligeable. On ne cherche donc pas à se protéger contre un adversaire disposant d'une puissance de calcul infinie mais plutôt d'une puissance de calcul pour laquelle on sait estimer une borne supérieure.

L'idée maîtresse 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, et par conséquent parfaitement reproductible, des suites d'allure aléatoire pouvant être utilisées comme des clés de même longueur que le message avec l'algorithme de chiffrement de Vernam [1]. Le déchiffrement agit ensuite de même, en générant la même suite à partir de la clé secrète.

En écho à la mise en garde de l'introduction de la section A.1.1, le chiffrement par flot dérivé de l'algorithme de Vernam fournit un exemple simple et particulièrement éloquent du fait qu'assurer la confidentialité, même de manière parfaite, n'entraîne pas automatiquement une protection en intégrité du message transmis. En effet, si un attaquant désire inverser un bit de message clair, c'est-à-dire transformer un 0 en 1 ou inversement, il lui suffit d'ajouter 1 au bit de chiffré c_i et donc de transmettre $c'_i = c_i \oplus 1$. Lors du déchiffrement, le destinataire va calculer $c'_i \oplus k_i = c_i \oplus 1 \oplus k_i = m_i \oplus 1$; si m_i vaut 0 le résultat obtenu est un 1 et, inversement, si m_i vaut 1 le résultat est $1 \oplus 1 = 0$. Par conséquent, même si l'attaquant n'a aucune idée de la valeur du bit modifié, il peut à coup sûr et sans effort l'inverser. À titre d'exemple d'application, on peut par exemple imaginer 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, facile à connaître. On peut alors inverser un bit de cette somme et ainsi facilement transformer un faible montant en une somme très importante, comme si l'on transformait, en notation décimale, 0000017 euro en 1000017 euro.

La confusion classique entre confidentialité et intégrité est souvent renforcée par des images un peu trop simplistes du mécanisme d'action des primitives cryptographiques. On présente ainsi souvent le chiffrement comme la version électronique d'une enveloppe opaque ou, pire, d'un coffre-fort. Comme on imagine mal pouvoir modifier des informations

1. Plus précisément, cette approche conduit à une classe particulière d'algorithmes de chiffrement par flot appelés « binary additive stream ciphers ».

contenues dans un coffre sans pouvoir en prendre connaissance, on peut à tort s'imaginer que le chiffrement protège totalement les données, tant en confidentialité qu'en intégrité. L'exemple précédent montre cependant qu'employé seul, le chiffrement peut se révéler parfaitement inefficace face à certaines menaces. Si l'on désire réaliser des « coffres-forts numériques », il faut donc au minimum, en plus de la confidentialité, assurer l'intégrité des données. Ceci peut se faire en combinant deux mécanismes séparés ou bien au moyen d'un mécanisme conçu pour assurer simultanément la confidentialité et l'intégrité. Une telle mise en place de mécanisme doit cependant être réalisée avec le plus grand soin.

On parle de chiffrement par flot **synchrone** lorsque la suite chiffrante est calculée à partir de la clé secrète, indépendamment du message à chiffrer. Inversement, les algorithmes de chiffrement par flot **asynchrones**, ou **auto-synchronisants** utilisent des suites chiffrantes dépendant de la clé secrète mais également d'un certain nombre de bits de texte chiffré. L'intérêt avancé pour une telle approche est de permettre une sorte de resynchronisation automatique du déchiffrement, même si des portions de message chiffré sont perdues lors de la transmission.

Notons qu'une telle propriété tient plus de la correction d'erreurs de transmission que d'une quelconque protection de nature cryptographique des données. On peut dès lors s'interroger sur l'adéquation de telles techniques ainsi que sur la menace réelle qu'elle vise à éviter. Notons également que certains modes opératoires de chiffrement par bloc, tel que le mode CBC vu précédemment, possèdent aussi cette propriété d'auto-synchronisation à condition que des blocs entiers soient perdus.

A.1.2 Sécurité du chiffrement

Au-delà de la description des primitives de chiffrement, il convient de définir précisément ce que l'on attend comme propriétés de sécurité de leur part. L'idée intuitive selon laquelle aucune information sur le message clair ne doit pouvoir être déduite du chiffré par un attaquant est délicate à définir précisément et nécessite beaucoup de soin. La cryptographie moderne a cependant développé des **modèles de sécurité** très précis afin d'expliciter sans ambiguïté ce que l'on attend des primitives de chiffrement.

Afin de préciser la difficulté d'une définition précise de ce que l'on attend du chiffrement, reprenons l'exemple de la randomisation vue section A.1.1.1. Une propriété naturellement attendue d'un algorithme de chiffrement est d'être non-inversible par quelqu'un qui ne dispose pas de la clé secrète. En l'absence de toute attaque connue, cette propriété semble vérifiée par l'AES ou le triple DES. Il a cependant déjà été remarqué qu'un chiffrement déterministe (c'est le cas de l'AES ou du triple DES), aussi bon soit-il, entraîne que deux chiffrements du même message génèrent des chiffrés identiques et qu'une part d'information est ainsi dévoilée. La propriété de non-inversibilité est donc insuffisante et doit être raffinée.

L'idée selon laquelle la connaissance de messages chiffrés ne doit révéler aucune information sur les messages clairs associés est classiquement formalisée par la notion de « sécurité sémantique ». Une telle définition est complexe mais peut se résumer à l'intuition suivante : si un mécanisme de chiffrement est tel qu'en proposant deux messages différents de son choix (mais de même taille), notés M_0 et M_1, et en obtenant le chiffré C de l'un des deux, on est incapable de savoir lequel des deux messages a été chiffré, ceci signifie que la vue d'un chiffré ne contient aucune information exploitable sur le clair associé, quels que soient les messages traités.

Il est de plus possible de renforcer encore ce modèle en tenant compte de capacités éventuellement très évoluées d'un attaquant. On peut par exemple reprendre l'expérience précédente en permettant à celui qui propose les deux messages M_0 et M_1 d'obtenir en plus le chiffré de n'importe quel autre message de son choix ainsi que le déchiffré de n'importe quel chiffré, évidemment différent de C. S'il est encore impossible de deviner si C est le chiffré de M_0 ou M_1, il est clair que le mécanisme de chiffrement sera résistant dans un grand nombre de scénarios d'attaque.

Les modèles de sécurité les plus forts sont obtenus par la combinaison d'un objectif de sécurité exigeant (par exemple la sécurité sémantique) et d'un attaquant disposant de ressources puissantes (par exemple, des oracles de chiffrement et/ou de déchiffrement). Ces modèles offrent donc les meilleures garanties. Le niveau de sécurité le plus élevé correspond au modèle de « l'indistinguabilité face aux attaques à chiffrés choisis adaptatives », désigné par l'acronyme IND-CCA2.

Les modèles de sécurité utilisés en cryptographie moderne peuvent parfois paraître trop généreux pour les attaquants, mais ils sont justifiés par le fait que dans certaines circonstances, les attaquants contre lesquels on veut se prémunir ont une grande marge de manœuvre. Évaluer la sécurité d'une

primitive dans de tels modèles de sécurité permet également de minimiser les risques de faille de sécurité lors de leur composition [2] avec d'autres mécanismes afin de concevoir des systèmes complets. De plus, utiliser des primitives sûres face à des attaquants très puissants est un important gage de qualité. Enfin, la cryptographie moderne propose des primitives 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.1.3 Authentification et intégrité de messages

Des techniques cryptographiques symétriques permettent également de garantir l'intégrité de données transmises, qu'elles soient déjà protégées en confidentialité ou non. De tels mécanismes visent à garantir qu'aucune altération des données n'a eu lieu au cours de leur transmission. L'inadéquation des mécanismes de chiffrement pour garantir une telle intégrité des données a déjà été mentionnée. Notons par ailleurs que les méthodes cryptographiques visent en général à se prémunir face à des attaques volontaires, potentiellement intelligentes, par opposition aux techniques de codage et aux protocoles de transmission visant à détecter ou à corriger des erreurs aléatoires et involontaires.

Plus précisément, la principale technique permettant d'assurer l'intégrité des données consiste à calculer un **code d'authentification de message** (souvent appelé MAC pour « message authentication code ») à partir des données à protéger et d'une clé secrète partagée avec celui à qui le message est destiné. Ce code d'authentification, typiquement long de 128 bits, est ensuite ajouté au message et transmis. Après réception, le code d'authentification est recalculé à l'aide de la clé secrète et du message reçu, potentiellement corrompu. Le résultat obtenu est comparé au MAC reçu ; s'ils sont identiques, il est extrêmement probable que les données sont intègres et proviennent donc de la bonne personne.

> Il faut insister sur la différence conceptuelle fondamentale existant entre chiffrement et calcul de code d'authentification de message. Par contre, à un niveau plus technique, de nombreuses similarités peuvent réapparaître. L'algorithme le plus connu de calcul de MAC est le CBC-MAC. Le code d'authentification est alors simplement calculé en appliquant le mode de chiffrement CBC (décrit figure A.1) 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 A.2. On voit clairement que toute modification, même minime, du message à protéger engendre un résultat totalement différent comme MAC. Il faut cependant se garder de penser qu'un tel mécanisme est sûr, dans un sens général, sur cette simple impression initiale, car une telle analyse ne constitue pas une preuve de sécurité.

La protection de l'intégrité d'un message garantit également, vis-à-vis du destinataire, son authenticité car la connaissance de la clé secrète est nécessaire pour générer des MAC corrects. Par contre, cette authentification n'a pas de valeur vis-à-vis d'un tiers car rien ne permet de savoir par quel détenteur de la clé secrète un MAC est réellement généré. De plus, toute vérification par un tiers nécessite de lui révéler la clé secrète utilisée. Ceci est intrinsèquement lié à la nature symétrique du mécanisme.

On désigne parfois les codes d'authentification de message sous le terme de **signature symétrique** mais il doit être bien entendu que ce qualificatif de signature est impropre pour la simple raison que la propriété de non-répudiation n'est pas assurée. Actuellement, seuls les mécanismes asymétriques, qui évitent justement que la connaissance de la clé secrète ne soit nécessaire pour vérifier la validité d'une signature, peuvent réellement rendre de tels services de sécurité.

2. C'est-à-dire utilisation conjointe.

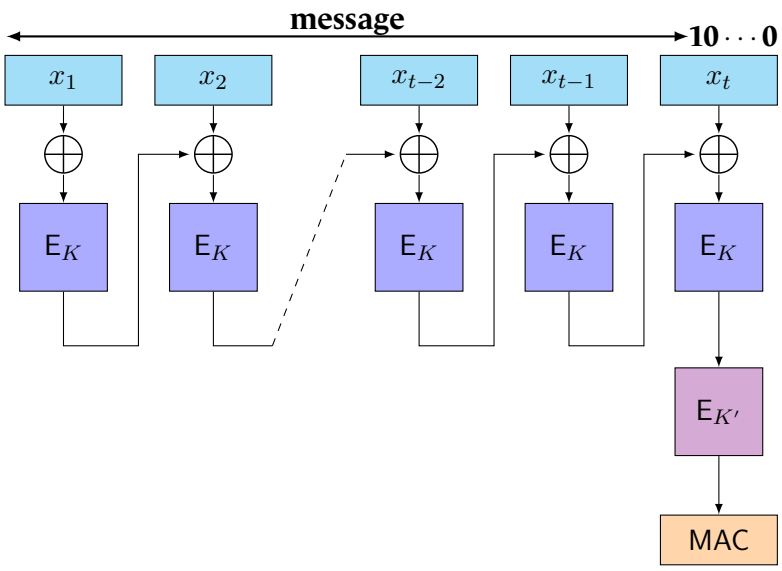

FIGURE A.2 – Mode opératoire CBC-MAC

Notons enfin qu'un code d'authentification de message valide ne permet que de garantir l'authenticité de ce message. Si l'on souhaite assurer l'intégrité et l'authenticité d'un ensemble de messages formant une communication, et éviter par exemple le rejeu ou la suppression de certains messages accompagnés de MACs valides, il convient de soigner les méthodes de chaînage employées.

A.1.4 Authentification d'entités

À partir des primitives de chiffrement et de calcul de MAC qui viennent d'être décrites, il est facile de dériver des mécanismes permettant d'authentifier des entités, ce terme étant pris au sens large. La principale différence entre l'authentification d'entités et celle de messages est le caractère **interactif** de la première alors que la seconde doit pouvoir être effectuée en une seule transmission, comme par exemple dans des applications de messagerie sécurisée.

La méthode symétrique la plus utilisée afin de s'assurer de l'identité d'un correspondant est de partager avec lui une clé secrète. Afin de s'assurer ensuite que quelqu'un se présentant sous son identité est bien la bonne personne, l'authentification va consister à s'assurer que ce dernier possède bien la clé secrète. La technique la plus simple, s'apparentant directement aux techniques de mot de passe, consiste à transmettre le secret. Les inconvénients sont cependant multiples, à commencer par le risque qu'un simple attaquant passif intercepte le secret et l'utilise ensuite. Par conséquent, une bien meilleure méthode est celle dite par « challenge-response », que l'on peut traduire par « question-réponse » ou « défi-réponse ». L'idée est d'envoyer à celui que l'on souhaite authentifier un message quelconque, de lui demander de le chiffrer en utilisant la clé secrète partagée, puis de renvoyer le résultat obtenu. Si la réponse reçue se déchiffre correctement et aboutit à la question posée, il y a une forte probabilité pour que la personne qui a calculé le chiffré possède effectivement la clé secrète et, par conséquent, qu'elle soit la personne qu'elle prétend être. Notons cependant que ceci n'est vrai que si aucune question déjà posée n'est réutilisée lors des authentifications suivantes.

Le problème de la non-réutilisation des questions est très sensible mais peut facilement être résolu si l'on dispose d'une source de données aléatoires, sans avoir à mémoriser les questions déjà posées. En effet, il est possible de calculer la

probabilité qu'une même question de n bits soit obtenue deux fois si les questions sont tirées au hasard. Par application du fameux « paradoxe des anniversaires », on montre que la première collision apparaît en moyenne après environ $2^{n/2}$ tirages. Par conséquent, si le nombre maximal d'authentifications avec une même clé est nettement inférieur à $2^{n/2}$, l'emploi de challenges aléatoires de n bits est suffisant, le risque de poser deux fois la même question étant négligeable. À titre d'application, des questions de 64 bits sont parfaitement adaptées jusqu'à plusieurs millions d'authentifications et des challenges de 128 bits potentiellement utilisables sans limite atteignable en pratique.

Le problème majeur de telles authentifications symétriques réside, comme dans le cas du chiffrement et de l'intégrité de messages, dans la nécessité de partager une clé secrète entre personne identifiante et personne identifiée. Ceci implique de plus l'impossibilité de distinguer ces deux personnes. On peut ainsi se demander s'il est vraiment nécessaire que celui qui vérifie l'identité connaisse nécessairement le secret associé à l'identité d'un individu. Une fois de plus, la réponse à ce problème est apportée par la cryptographie asymétrique dont les bases vont maintenant être décrites.

A.2 Cryptographie asymétrique

L'idée majeure de la cryptographie asymétrique est que les opérations publiques (le chiffrement, la vérification de signature…) n'ont pas nécessairement besoin d'utiliser les mêmes clés que les opérations privées (le déchiffrement, la signature…). Ainsi, la cryptographie asymétrique, telle que décrite par W. Diffie et M. Hellman [DH76], 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. 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 [3]. 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 fort 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.

3. Les nombres premiers étant les nombres divisibles uniquement par 1 et par eux-mêmes. Par exemple, 2, 3, 5, 7 sont premiers, mais 6 — qui est divisible par 1, 2, 3 et 6 — ne l'est pas.

Parmi les problèmes asymétriques, comme celui de la factorisation, on distingue les problèmes asymétriques dits « à trappe ». Ils se fondent sur une opération facile à réaliser mais difficile à inverser à moins de disposer d'un élément supplémentaire appelé la « trappe ». L'exemple le plus connu, mais également le plus utilisé dans les produits actuels, est l'opération sur laquelle repose le fameux cryptosystème RSA. Cette opération est directement liée au problème de la factorisation, la connaissance des facteurs premiers constituant la trappe permettant d'inverser facilement le calcul.

En termes de comparaison avec le monde physique, on peut par exemple penser au mélange de limaille de fer et de limaille de cuivre. Comme dans le cas des pots de peinture de couleurs primaires différentes, l'opération de mélange est aisée mais l'opération inverse est très complexe, à moins de disposer d'un aimant qui constitue en quelque sorte la trappe.

Les recherches en cryptographie de ces 30 dernières années ont permis de disposer de quelques problèmes mathématiques asymétriques utilisables à des fins cryptographiques. Ils ne sont cependant pas nombreux et reposent presque tous sur des problèmes issus de la théorie des nombres : factorisation et calculs de « logarithmes discrets » dans des structures mathématiques variées, dont les « courbes elliptiques » sont des exemples particulièrement intéressants.

L'idée de base des problèmes de logarithme discret consiste à utiliser une structure mathématique contenant un nombre fini d'éléments que l'on peut combiner au moyen d'une opération possédant des propriétés semblables à celles de l'addition ou de la multiplication sur les entiers classiques. Une telle structure est appelée « groupe » en algèbre, le groupe étant dit additif si l'opération est notée $+$, ou multiplicatif si l'opération est notée $\times$. On peut ensuite définir une version itérée de l'opération agissant sur les éléments du groupe. Si l'on note multiplicativement l'opération et si x désigne un élément du groupe alors le produit de n copies de x se note « x à la puissance n », soit x^n. Le calcul de x^n est facile, même pour de très grandes valeurs de l'entier n. Par contre, dans certains groupes, l'opération inverse consistant, à partir de x et de x^n, à retrouver l'entier n est très difficile, bien que mathématiquement parfaitement définie. On parle de calcul de « logarithme discret », n étant appelé « logarithme de x^n en base x ». Comme dans le cas de la factorisation, il doit être clair que le calcul de logarithme n'est pas impossible en théorie ; il suffit par exemple de tester toutes les valeurs possibles de n, même s'il existe de bien meilleures méthodes. Cependant, pour des dimensionnements suffisants, aucune méthode efficace permettant d'effectuer de tels calculs en temps raisonnable dans les groupes utilisés en cryptographie n'est connue. Les « courbes elliptiques », si leurs paramètres sont choisis avec soin, sont de tels exemples de groupes dans lesquels on suppose que le calcul de logarithme discret est particulièrement difficile.

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 cryptosystème. 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 128 bits, voire 256 bits pour les plus longues. Les tailles de clés symétriques et asymétriques ne sont donc pas comparables [4].

Notons enfin que l'emploi d'un problème réellement difficile et correctement dimensionné est une condition nécessaire afin d'obtenir un niveau de sécurité recherché. Bien entendu, ceci est très loin d'être suffisant car bien d'autres sources de failles de sécurité existent, que ce soit dans l'emploi du problème, dans les modes opératoires, dans la gestion des clés, dans les problèmes de conception ou d'implantation des protocoles, *etc.*

4. 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 PC en moins d'une heure.

A.2.1 Chiffrement asymétrique

L'idée essentielle du **chiffrement asymétrique**, encore souvent appelé chiffrement à clé publique, est que rien n'impose à l'émetteur d'un message chiffré d'être capable de déchiffrer les messages qu'il envoie. Dit autrement, aussi naturelle qu'elle puisse paraître, l'intuition selon laquelle la même clé doit être utilisée à la fois pour le chiffrement et le déchiffrement n'est pas fondamentalement justifiée.

Dans le cadre du chiffrement, l'idée est d'utiliser des paires de clés, ou bi-clés, composées d'une **clé privée**[5] et d'une **clé publique** associée. Afin de chiffrer un message à l'attention d'un correspondant, on utilise la clé publique de ce dernier. Après transmission, l'opération inverse de déchiffrement est effectuée en utilisant la clé privée. Les propriétés du mécanisme sont telles que la connaissance de la clé publique ne permet pas de retrouver la clé privée. Par conséquent, la clé publique peut, comme son nom l'indique, être largement diffusée. Par contre, la clé privée doit être gardée confidentielle.

> 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 la cryptographie 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 du 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 « randomiser » le chiffrement 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, c'est-à-dire l'ensemble des messages, est restreint à un petit sous-ensemble

5. L'usage veut que l'on réserve le terme de clé secrète aux applications symétriques et le terme de clé privée aux applications asymétriques. Une clé publique est donc en général naturellement associée à une clé privée, alors qu'une telle association n'a aucun sens pour une clé secrète.

des messages possibles. Cette problématique est d'ailleurs similaire à celle rencontrée dans le cas symétrique et exposée en section A.1.1.1.

> 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.

A.2.2 Signature cryptographique asymétrique

La **signature cryptographique** 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 [6], à la manière des MACs vus au chapitre A.1.1. 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, certains diront même en mieux. Notons cependant qu'ici encore la certification de la clé publique est un problème crucial qui est abordé rapidement ci-dessous dans le chapitre A.5 consacré 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.

> On a souvent tendance à présenter 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 schémas de signature ne peuvent être utilisés à des fins de chiffrement.

A.2.3 Authentification asymétrique d'entités et établissement de clé

Les idées permettant une authentification interactive d'entités vues au chapitre A.1.1 peuvent bien entendu s'étendre aux primitives asymétriques. Afin d'authentifier quelqu'un 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

6. 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.

des schémas 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.

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 [7]. 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 [DH76] 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 l'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 avec une personne dont on ignore l'identité 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 schéma 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 [8].

A.2.4 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.

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 [9] ». Les preuves en clé publique n'assurent ainsi jamais une « sécurité inconditionnelle »

7. Cette clé secrète pouvant par exemple par la suite être utilisée pour chiffrer des messages.
8. Dans le premier cas, il suffirait en effet à un attaquant d'attendre le succès de la phase d'authentification pour mettre en oeuvre une attaque par le milieu. Dans le second cas, il lui suffirait de réaliser une attaque par le milieu contre le schéma d'établissement de clé, puis de relayer sur les canaux chiffrés ainsi établis les messages du protocole d'authentification entre ces entités.
9. En un temps raisonnable.

— c'est-à-dire face à un attaquant de puissance infinie [10]. 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 schéma sans résoudre un certain problème mathématique.

> Afin d'illustrer l'importance mais également la difficulté de définition d'un modèle de sécurité, prenons l'exemple de la signature. 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 [BR94] : la sécurité « asymptotique » et la sécurité « exacte ». Quand la première se contente d'assurer que le schéma 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 schéma 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 schéma 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 schéma que de résoudre le problème). On voit ainsi que les schémas à préférer sont ceux apportant les réductions les plus fines possibles, car les autres schémas nécessitent de plus grandes clés et paramètres pour une même garantie de sécurité. De même, s'il n'est pas possible de sélectionner un schéma avec une réduction fine, il faut en 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 au chapitre A.1.1, 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 Fonctions de hachage

Un certain nombre de primitives cryptographiques, ou directement utilisées en cryptographie, ne sont pas paramétrées par des clés. La classification usuelle basée sur la nécessité ou l'absence de nécessité de partager un secret commun ne s'applique pas. Ces primitives sont néanmoins souvent classées parmi les primitives symétriques dans la mesure où leurs caractéristiques principales empruntent beaucoup aux primitives à clé secrète ou que le cadre de leur utilisation nécessite qu'émetteur et récepteur réalisent les mêmes opérations (de manière symétrique, donc).

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 un **condensat**, encore appelé **empreinte** ou **haché**, de taille fixée. On demande de plus à une fonction de hachage cryptographique d'être en pratique non inversible, c'est-à-dire qu'il ne soit pas possible étant donné un condensat de trouver un message dont l'image par la fonction de hachage est égale à ce condensat : on dit que le calcul d'un antécédent est diffcile. On demande en outre qu'il ne soit pas possible de trouver deux messages distincts ayant même condensat. On dit alors que la fonction de hachage est **sans collision**.

> Bien entendu, 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.

10. La puissance étant ici la mémoire et la puissance de calcul.

Une fois encore le paradoxe des anniversaires peut s'appliquer ; si l'on note n le nombre de bits du condensat et si la fonction de hachage peut être considérée comme ayant un comportement relativement aléatoire malgré le fait qu'elle soit parfaitement définie en tous points [11], alors, 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 le même condensat. Ceci fournit une borne inférieure à la taille nécessaire des sorties des fonctions de hachage cryptographiques.

Une des nombreuses applications des fonctions de hachage apparaît dans les schémas de signature numérique afin de réduire le message de longueur quelconque à un simple condensat de taille fixée et petite. Elles sont également utilisées afin de réaliser certains codes d'authentification de message (par exemple HMAC).

Suite aux publications consécutives à l'annonce d'attaques sur les principales familles de fonctions de hachage en août 2004, le panel des fonctions de hachage considérées comme sûres s'est beaucoup réduit puisque n'y apparaissent plus ni MD5 (définitivement cassé) ni SHA-1 (qui n'est plus conforme au référentiel). Dans l'attente de la publication de nouveaux standards, dont par exemple SHA-3, et de leur mise à l'épreuve, la famille SHA-2 reste utilisable.

A.4 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.

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 vrai** ou 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

11. C'est-à-dire que, avant d'avoir fait le calcul, la valeur du hachage en un point est aléatoire pour un utilisateur. Une bonne fonction de hachage a ce comportement aléatoire, alors qu'une simple fonction de troncature par exemple n'est pas du tout aléatoire.

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 vrai. 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.5 Gestion de clés

A.5.1 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 au chapitre A.2.1.

A.5.2 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

Cette annexe présente quelques informations issues d'annonces et de publications académiques permettant de contribuer à la justification de certains dimensionnements cryptographiques.

B.1 Records de calculs cryptographiques

Les records de calculs cryptographiques permettent de donner une borne inférieure à la difficulté pratique de certains problèmes. Ils concernent essentiellement le « cassage » de clés symétriques par énumération de l'ensemble des clés possibles (recherche dite exhaustive) ainsi que la résolution de problèmes mathématiques issus de la théorie des nombres (factorisation et logarithme discret dans des structures variées).

B.1.1 Records de calculs en cryptographie symétrique

Les principaux records de calcul rendus publics ont utilisé le réseau Internet et le bénévolat d'internautes acceptant d'effectuer des calculs sur leur ordinateur personnel en « tâche de fond ». Les principaux résultats sont d'une part le cassage de clés DES de 56 bits :

- 17 juin 1997, **96 jours** de calcul distribué sur Internet ;
- 23 février 1998, **41 jours** de calcul distribué sur Internet [1] ;
- 17 juillet 1998, **56 heures** de calcul sur une machine spécifique [2] dont le coût a été estimé à 250 000 dollars ;
- 19 janvier 1999, **22 heures** de calcul en combinant la machine précédemment citée et des calculs sur Internet.

D'autre part le cassage de clés de chiffrement RC5 :

- 19 octobre 1997, version **56 bits** cassée après 250 jours de calcul sur Internet ;
- 14 juillet 2002, version **64 bits** cassée après 1757 jours de calcul sur Internet.

Il va de soi que ces « records » ne tiennent pas compte des capacités de calcul de services gouvernementaux spécialisés qui ne recherchent bien évidemment pas la publicité. Ceci ne doit cependant pas engendrer de paranoïa excessive, comme expliqué au paragraphe A.1.1.

B.1.2 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 :

1. Voir http://www.distributed.net.
2. Voir http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker.

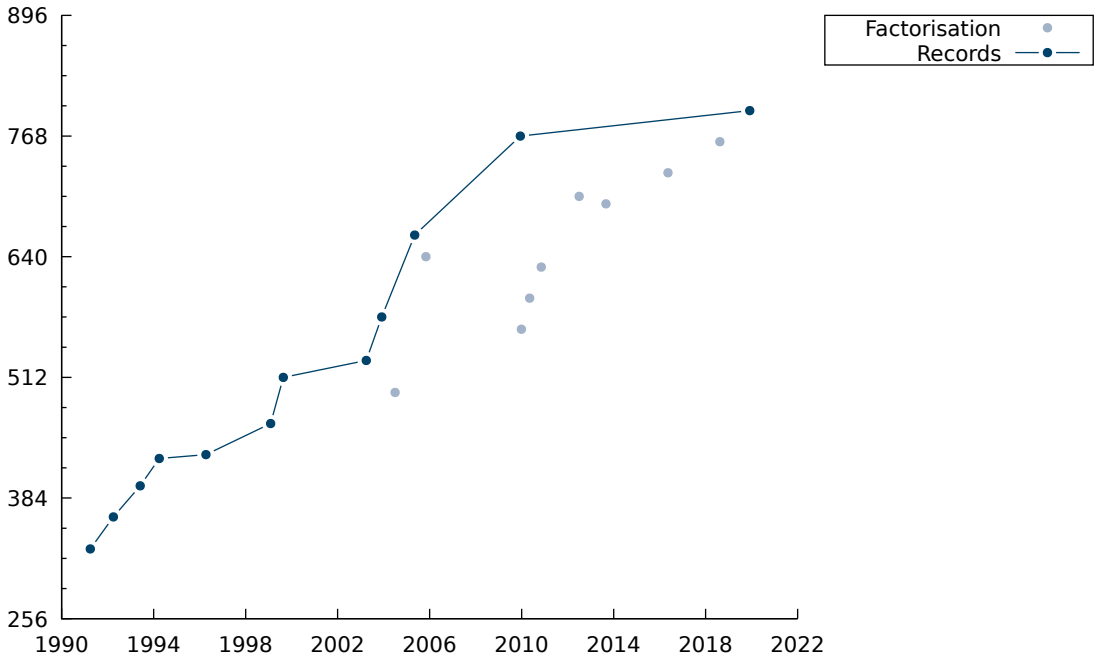

FIGURE B.1 – Tailles des factorisations de grands entiers publiées depuis 1988. Les records sont mis en évidence.

- 9 juin 1993, Denny, Dodson, Lenstra et Manasse, **397** bits (120 chiffres décimaux) ;
- 2 avril 1994, Atkins, Graff, Lenstra et Leyland, **426** bits (129 chiffres décimaux) ;
- 10 avril 1996, Lenstra et al., **432** bits (130 chiffres décimaux) ;
- 2 février 1999, te Riele et al., **466** bits (140 chiffres décimaux) ;
- 22 août 1999, te Riele et al., **512** bits (155 chiffres décimaux) ;
- 18 janvier 2002, Bahr, Franke et Kleinjung, **524** bits (158 chiffres décimaux) ;
- 1er avril 2003, Bahr, Franke, Kleinjung, Lochter et Boehm, **530** bits (160 chiffres décimaux) ;
- 3 décembre 2003, Franke et Kleinjung, **576** bits (174 chiffres décimaux) ;
- 9 mai 2005, Bahr, Boehm, Franke et Kleinjung, **663** bits (200 chiffres décimaux) ;
- 12 décembre 2009, Kleinjung et al., **768** bits (232 chiffres décimaux) ;
- 2 décembre 2019, Thomé et al, **795** bits (240 chiffres décimaux).

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 [3].

B.1.2.1 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

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

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.1.2.2 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).

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.1.3 Records de calcul de logarithme discret dans GF(p)

Les principaux records successifs en termes de calcul de logarithme discret dans un corps fini premier à p éléments GF(p) sont les suivants. Pour chaque record, la taille — exprimée en bits et en chiffres décimaux — du nombre premier p est indiquée :

- 25 novembre 1996, Weber, Denny et Zayer, **281** bits (85 chiffres décimaux) ;
- 26 mai 1998, Joux et Lercier, **298** bits (90 chiffres décimaux) ;
- novembre 1999, Joux et Lercier, **331** bits (100 chiffres décimaux) ;
- 19 janvier 2001, Joux et Lercier, **364** bits (110 chiffres décimaux) ;
- 17 avril 2001, Joux et Lercier, **397** bits (120 chiffres décimaux) ;
- 18 juin 2005, Joux et Lercier, **431** bits (130 chiffres décimaux) ;
- 5 février 2007, Bahr, Franke et Kleinjung, **530** bits (160 chiffres décimaux).

B.1.4 Records de calcul de logarithme discret dans GF(2^n)

Les principaux records successifs en termes de calcul de logarithme discret dans un corps fini à 2^n éléments GF(2^n) avec n un nombre premier sont :

- 1992, Gordon et McCurley, GF(2^{401}) soit **401** bits (121 chiffres décimaux) ;
- 25 septembre 2001, Joux et Lercier, GF(2^{521}) soit **521** bits (157 chiffres décimaux) ;
- 23 février 2002, Thomé, GF(2^{607}) soit **607** bits (183 chiffres décimaux) ;
- 22 septembre 2005, Joux et Lercier, GF(2^{613}) soit **613** bits (185 chiffres décimaux) ;
- 10 avril 2013, Barbulescu et al., GF(2^{809}) soit **809** bits (183 chiffres décimaux).

Les principaux records successifs en termes de calcul de logarithme discret dans un corps fini à 2^n éléments GF(2^n) avec n un nombre composé sont :

- 11 février 2013, Joux, GF(2^{1778}) soit **1778** bits (536 chiffres décimaux) ;
- 19 février 2013, Gologlu, Granger, McGuire et Zumbragel, GF(2^{1971}) soit **1971** bits (594 chiffres décimaux) ;
- 22 mars 2013, Joux, GF(2^{4080}) soit **4080** bits (1229 chiffres décimaux) ;
- 11 mai 2013, Granger, Granger, McGuire et Zumbragel, GF(2^{6120}) soit **6120** bits (1843 chiffres décimaux) ;

- 21 mai 2013, Joux, $\mathsf{GF}(2^{6168})$ soit **6168** bits (1857 chiffres décimaux) ;
- 31 janvier 2014, Granger, Kleinjung et Zumbragel, $\mathsf{GF}(2^{9234})$ soit **9234** bits (2780 chiffres décimaux).

On notera en particulier qu'à taille de corps équivalente, le calcul de logarithme discret est bien plus facile dans $\mathsf{GF}(2^n)$ que dans $\mathsf{GF}(p)$.

B.1.5 Records de calcul de logarithme discret dans $\mathsf{GF}(p^n)$

Les principaux records en termes de calcul de logarithme discret dans un corps fini à p^n éléments $\mathsf{GF}(p^n)$, pour p égal à 3, sont les suivants :

- 19 février 2010, Hayashi et al., $\mathsf{GF}(3^{426})$ soit **676** bits (204 chiffres décimaux) ;
- 17 juin 2012, Hayashi, Shimoyama, Shinohara et Takagi, $\mathsf{GF}(3^{582})$ soit **923** bits (278 chiffres décimaux) ;
- 28 janvier 2014, Adj, Menezes, Oliveira et Rodriuez-Henriquez, $\mathsf{GF}(3^{822})$ soit **1303** bits (393 chiffres décimaux).

Les principaux records en termes de calcul de logarithme discret dans un corps fini à p^n éléments $\mathsf{GF}(p^n)$, pour p supérieur à 3, sont les suivants :

- 28 juin 2005, Lercier et Vercauteren, $\mathsf{GF}(370801^{18})$ soit **334** bits (101 chiffres décimaux) ;
- 24 octobre 2005, Joux et Lercier, $\mathsf{GF}(65537^{25})$ soit **401** bits (121 chiffres décimaux) ;
- 9 novembre 2005, Joux et Lercier, $\mathsf{GF}(370801^{30})$ soit **556** bits (168 chiffres décimaux) ;
- août 2006, Joux, Lercier, Smart et Vercauteren, $\mathsf{GF}(p^3)$, où p est un nombre premier de 132 bits (40 chiffres décimaux), soit **394** bits (119 chiffres décimaux) ;
- 24 décembre 2012, Joux, $\mathsf{GF}(33553771^{47})$ soit **1175** bits (354 chiffres décimaux) ;
- 6 janvier 2013, Joux, $\mathsf{GF}(33341353^{57})$ soit **1425** bits (429 chiffres décimaux).

On notera en particulier qu'à taille de corps équivalente, le calcul de logarithme discret pour p et n de taille moyenne est plus facile que dans $\mathsf{GF}(p)$.

B.1.6 Calcul de logarithme discret sur courbe elliptique

La société Certicom a publié le 6 novembre 1997 une liste de challenges [4] concernant le problème du logarithme discret sur courbe elliptique. Les challenges sont de trois types : courbe elliptique « quelconque » définie sur $\mathsf{GF}(p)$, courbe elliptique « quelconque » définie sur $\mathsf{GF}(2^n)$, et courbe de Koblitz également définie sur $\mathsf{GF}(2^n)$. 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.

Le tableau B.1 reproduit les résultats annoncés jusqu'à aujourd'hui ainsi que les prochains challenges non-résolus et le nombre d'opérations qui ont été nécessaires.

Le 8 juillet 2009, Bos, Kaihara, Kleinjung, Lenstra et Montgomery ont résolu une instance du problème du logarithme discret sur une courbe elliptique définie sur $\mathsf{GF}(p)$ où p est un nombre premier de **112** bits.

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

TABLE B.1 – Challenges Certicom

Challenge	Date d'annonce	Nombre d'opérations
ECCp-79	06/12/1997	2^{40}
ECCp-89	12/01/1998	2^{44}
ECCp-97	18/03/1998	2^{47}
ECCp-109	06/11/2002	2^{54}
ECCp-131	Non résolu	
ECC2-79	16/12/1997	2^{40}
ECC2-89	09/02/1998	2^{44}
ECC2-97	22/09/1999	2^{47}
ECC2-109	15/04/2004	2^{54}
ECC2-131	Non résolu	
ECC2K-95	21/05/1998	2^{44}
ECC2K-108	04/04/2000	2^{51}
ECC2K-130	Non résolu	

B.2 Étude de la taille des clés d'après l'article de Lenstra [Len04]

A. Lenstra a publié en 2004 un article visant à comparer les niveaux de robustesse des divers problèmes employés en cryptographie. Ce travail académique doit bien entendu être considéré avec beaucoup de prudence ; il a cependant le mérite de proposer des modèles à la fois motivés et raisonnables.

Note

Mise en garde : ce travail est mentionné dans ce document car il est le plus complet réalisé à ce jour dans ce domaine par des chercheurs du milieu académique. Il est, de plus, souvent référencé, ainsi que sa version antérieure, datant de 2001. Le fait de le citer et d'en mentionner certains des résultats dans cette partie ne constitue cependant pas une caution de l'article pris dans son intégralité. En particulier, la responsabilité de certaines affirmations est laissée à leurs auteurs.

B.2.1 Évolution des tailles de clés symétriques

Il est facile d'estimer l'évolution des tailles de clés symétriques nécessaires sous l'hypothèse d'une loi de Moore selon laquelle la quantité de mémoire et la puissance de calcul disponibles pour un prix donné doublent tous les 18 mois. Le graphique de la figure B.2 indique l'évolution des tailles de clés nécessaires afin de maintenir un niveau de sécurité équivalent à celle du DES (56 bits) en 1982 : en vertu de ce qui précède, la longueur de clé nécessaire augmente d'un bit tous les 18 mois. Le graphique se lit donc de la façon suivante. L'année est donnée en abscisse. L'ordonnée indique la taille de clé en bits offrant une sécurité équivalente à la sécurité du DES en 1982. À titre de comparaison, on a représenté en pointillés ce que donnerait l'extrapolation basée sur une loi de

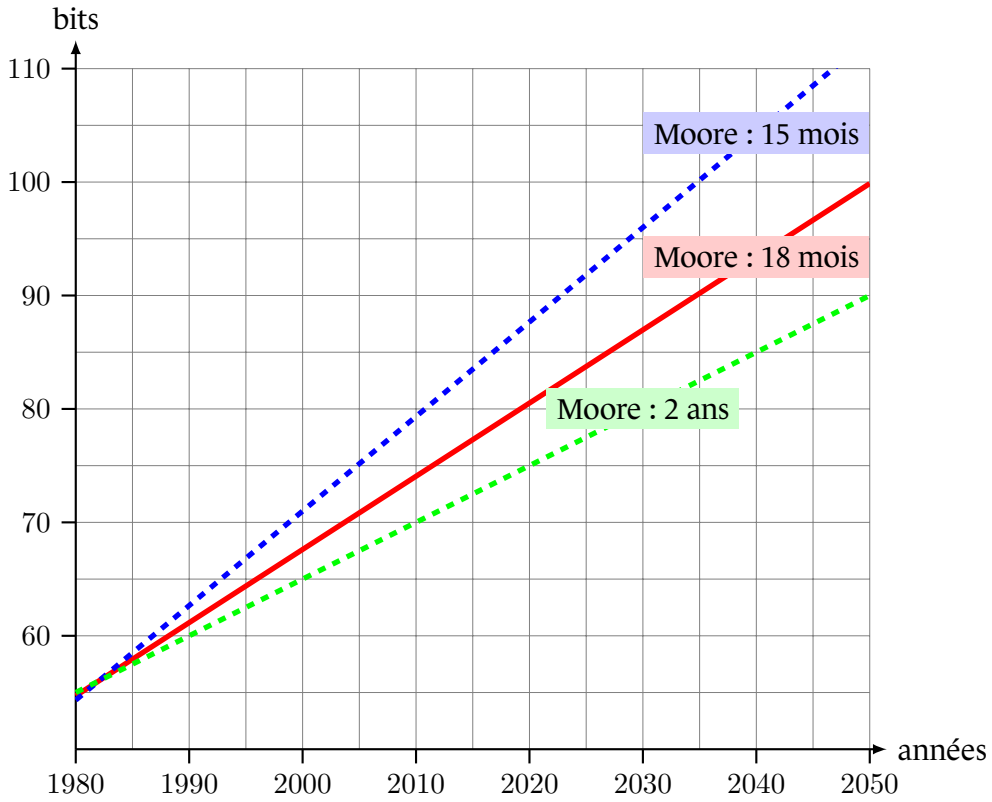

FIGURE B.2 – Taille de clé symétrique offrant une sécurité équivalente à celle du DES en 1982

Moore plus optimiste (doublement tous les 15 mois) ou plus pessimiste (doublement tous les 24 mois).

Remarque

- Des clés de 128 bits apportent un très haut niveau de sécurité face à une attaque par recherche exhaustive.
- Des clés de 256 bits offrent une sécurité démesurée face aux attaques par recherche exhaustive.
- En conclusion, on peut se protéger efficacement contre les attaques par recherche exhaustive, même à un horizon très lointain.

B.2.2 Évolution des tailles de modules en cryptographie asymétrique

Le même genre d'évaluation peut être réalisé pour des modules asymétriques.

Le graphique de la figure B.3 se lit de la façon suivante. L'année est donnée en abscisse. L'ordonnée indique la taille de module en bits offrant une sécurité équivalente à la sécurité du DES en 1982. En poursuivant le raisonnement de l'article de Lenstra [Len04], on adopte les hypothèses suivantes : tout d'abord les records publiés laissent à penser que les implantations matérielles conduisent à un meilleur rapport performances/coût ; de fait la loi de Moore d'un doublement de puissance du matériel tous les 18 mois est applicable ; par ailleurs, les améliorations dans l'implantation des

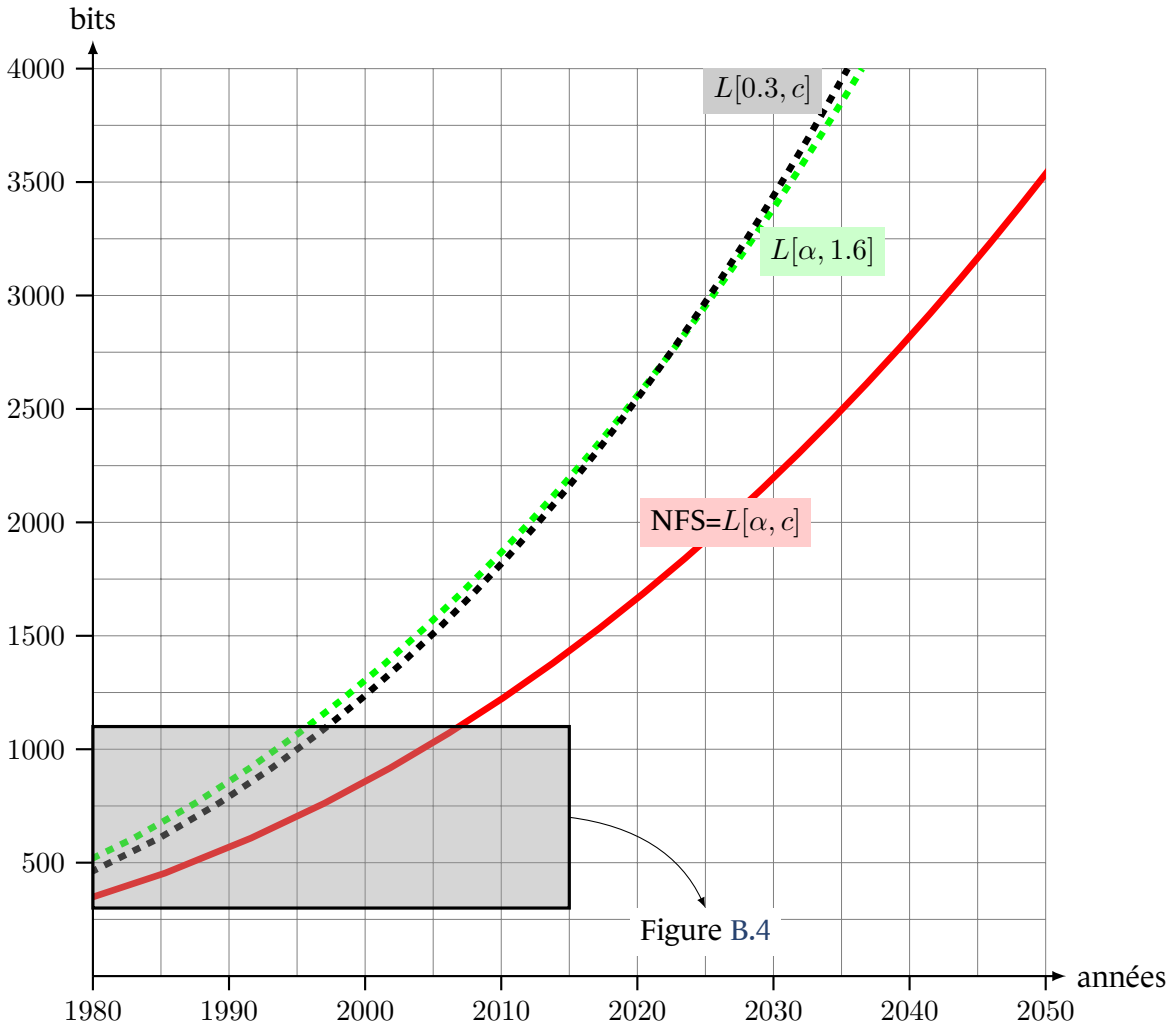

FIGURE B.3 – Taille de module offrant une sécurité équivalente à celle du DES en 1982. Influence de la constante (en vert) et de l'exposant (en noir) du crible algébrique (NFS). Les paramètres actuels de l'algorithme NFS sont $\alpha = 1/3$ et $c = \sqrt[3]{64/9} \approx 1.92$.

algorithmes de factorisation sont également à prendre en compte : l'expérience montre qu'elles correspondent assez bien à une progression de type loi de Moore. L'article de Lenstra fait l'hypothèse que les deux facteurs de progrès (améliorations du matériel et avancées algorithmiques) sont indépendants, et peuvent donc s'accumuler ; en l'occurrence, on obtient une division par deux de la difficulté de factoriser un module de taille donnée au bout de 9 mois.

Sur la figure B.3, la courbe rouge en trait plein représente l'évolution des tailles de module en se basant sur les hypothèses précédentes. Par ailleurs, les éventuels progrès algorithmiques, en factorisation d'entiers comme en calcul de logarithmes discrets, peuvent avoir une influence considérable sur le choix des tailles de paramètres. Le meilleur algorithme de factorisation connu aujourd'hui, le crible algébrique ou NFS, a une complexité notée $L[\alpha, c]$ qui dépend de deux facteurs : une « constante » c et un « exposant » α. La courbe verte en pointillé représente les tailles de module offrant une sécurité équivalente au DES en 1982, en supposant que l'on dispose d'un algorithme pour lequel la constante c vaut 1.60 au lieu de 1.92 (avec un exposant inchangé et une loi de Moore basée sur 9 mois). La courbe noire en pointillé représente le même phénomène en supposant que

l'on dispose d'un algorithme pour lequel l'exposant α vaut 0.30 au lieu de $1/3$ (Les autres paramètres étant pris, là encore, identiques à ceux de la courbe rouge). On notera que ces deux courbes s'intersectent.

Remarque

- La croissance est nettement non linéaire à cause de l'existence d'algorithmes dits « sous-exponentiels » tels que les cribles quadratiques et algébriques.
- À moyen terme, l'emploi de modules de grande taille s'impose.
- Aucune distinction n'est faite ici entre problème de factorisation et de logarithme discret dans $\mathsf{GF}(p)$ car cette distinction aurait peu de sens en pratique. On considère cependant que le problème du logarithme discret est légèrement plus difficile que celui de la factorisation ; en pratique, dans les gammes de tailles envisagées, 100 à 200 bits les séparent, cette approximation étant très imprécise.

Une comparaison de cette courbe et des records annoncés publiquement est nécessaire. Les courbes de la figure B.3 ont vocation à guider les choix de paramètres pour éviter le cassage d'un système, alors que les records ne traduisent que le savoir-faire académique en matière de cassage. Ces records ne rendent pas compte des réalisations non publiées.

Ces précautions étant prises, observons les records publiés et comparons-les aux courbes précédentes. Dans la figure B.4 :

- Les 10 records de factorisation sont indiqués par les billes bleues. La droite noire et épaisse est une interpolation linéaire de ces records.
- La courbe rouge continue correspond à la courbe rouge de la figure B.3 : meilleur algorithme connu (crible algébrique NFS), soutenu par une loi de Moore de doublement tous les 9 mois.
- La courbe rouge en pointillé représente le même scénario décalé de 12,5 ans.

Que peut-on conclure de ce graphe ? Il est tout d'abord évident que les points expérimentaux sont peu nombreux et donc certainement peu significatifs. De plus, on n'a pas tenu compte de l'effort de calcul qui a été nécessaire pour obtenir chacun de ces records. On constate cependant que l'hypothèse faite sur la loi de Moore (progrès conjoints du matériel et de la qualité d'implantation des algorithmes) correspond assez bien à une avance de 12 ans sur les records (ou sur leur interpolation). Dans tous les cas, il convient de rester extrêmement prudent quant à l'interprétation de ces résultats : les courbes obtenues ici sont basées sur des hypothèses qui, tout en étant réalistes, sont avant tout arbitraires. De plus ces courbes sont extrêmement proches les unes des autres et semblent très sensibles à une petite variation des paramètres.

En conclusion, des principes cryptographiques élémentaires poussent à tenir compte de toutes les hypothèses disponibles permettant de dimensionner correctement un système afin de se mettre à l'abri des attaquants les plus puissants. C'est le parti pris dans l'article de Lenstra et également par la plupart des experts du domaine. Les observations qui viennent d'être faites peuvent cependant pousser certains concepteurs à assumer le risque d'employer des modules plus petits afin de gagner en performance tout en maintenant une sécurité « apparemment suffisante ». Le débat peut bien entendu se prolonger à l'infini car il est impossible de le trancher objectivement avec des arguments indiscutables.

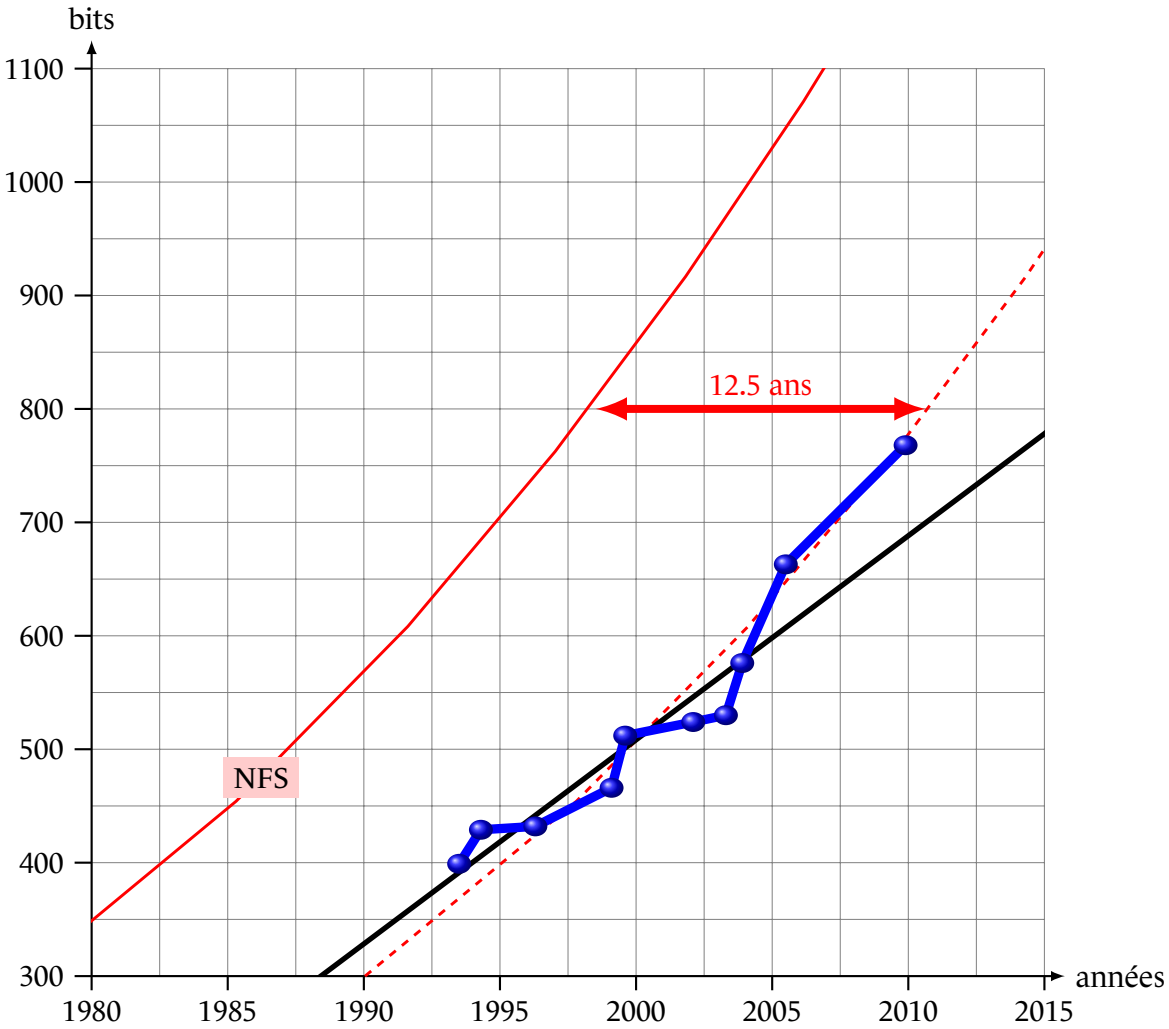

FIGURE B.4 – Comparaison entre les records de factorisation et les tailles théoriques nécessaires pour une sécurité équivalente à celle du DES en 1982

B.2.3 Évolution des tailles de courbes elliptiques

Le même travail est réalisé dans le cas de courbes elliptiques. La figure B.5 résume les différentes situations, et se lit de la façon suivante. L'année est donnée en abscisse. L'ordonnée indique la taille de module en bits offrant une sécurité équivalente à la sécurité du DES en 1982. La courbe rouge correspond au scénario actuel, en se basant sur une loi de Moore offrant un doublement de la puissance de calcul tous les 18 mois (autrement dit, la même hypothèse que les courbes rouges continues des figures précédentes). Une loi de Moore très optimiste d'un doublement tous les ans mène à la courbe en bleu et en pointillé. Enfin en tenant compte d'hypothétiques progrès algorithmiques (algorithmes en $O(q^{0.4})$ au lieu de $O(q^{0.5})$ où q désigne le nombre de points de la courbe), on obtient la courbe en noir et en pointillé.

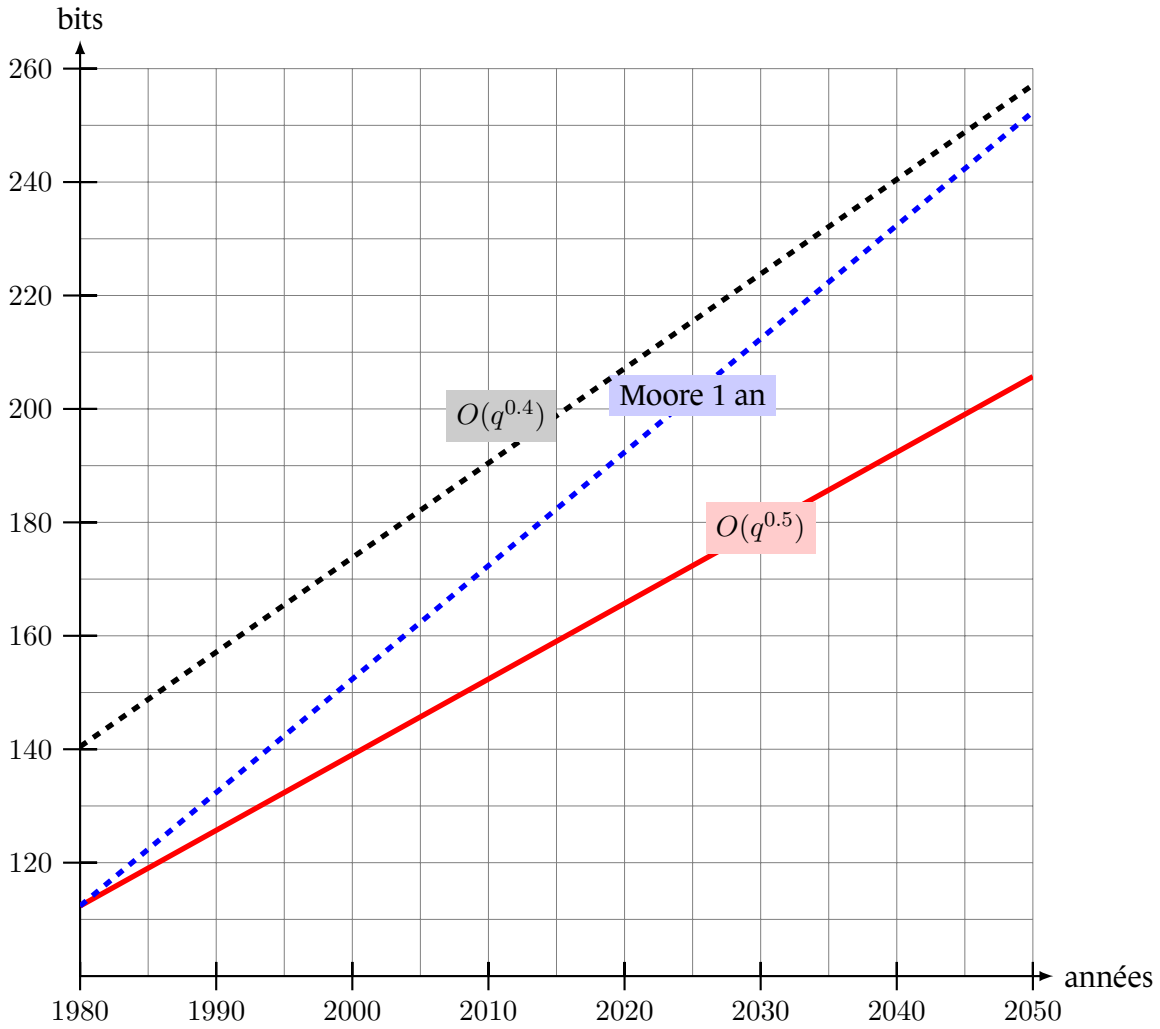

FIGURE B.5 – Taille de courbes elliptiques offrant une sécurité équivalente à celle du DES en 1982, et influence des potentiels progrès algorithmiques (en noir).

B.2.4 Équivalence de sécurité entre tailles de module asymétrique et de clé symétrique

Afin de comparer clés symétriques et asymétriques, on peut reprendre les données précédentes et tracer les équivalences, indépendamment du temps. On obtient les courbes de la figure B.6 comparant les clés symétriques et les modules asymétriques. Le graphique se lit de la façon suivante. Le nombre de bits des clés symétriques est donné en abscisse. L'ordonnée indique la taille en bits du module asymétrique offrant une sécurité équivalente. La courbe en rouge résulte de l'interprétation de la formule de complexité du meilleur algorithme connu (crible algébrique ou NFS). La courbe verte en pointillé représente le scénario où un progrès algorithmique permet un gain de 25% sur la constante c, soit une valeur $c = 1.44$. La courbe noire et en pointillé représente le scénario où un progrès de 25% est fait sur la valeur de l'exposant α, soit une valeur $\alpha = 1/4$.

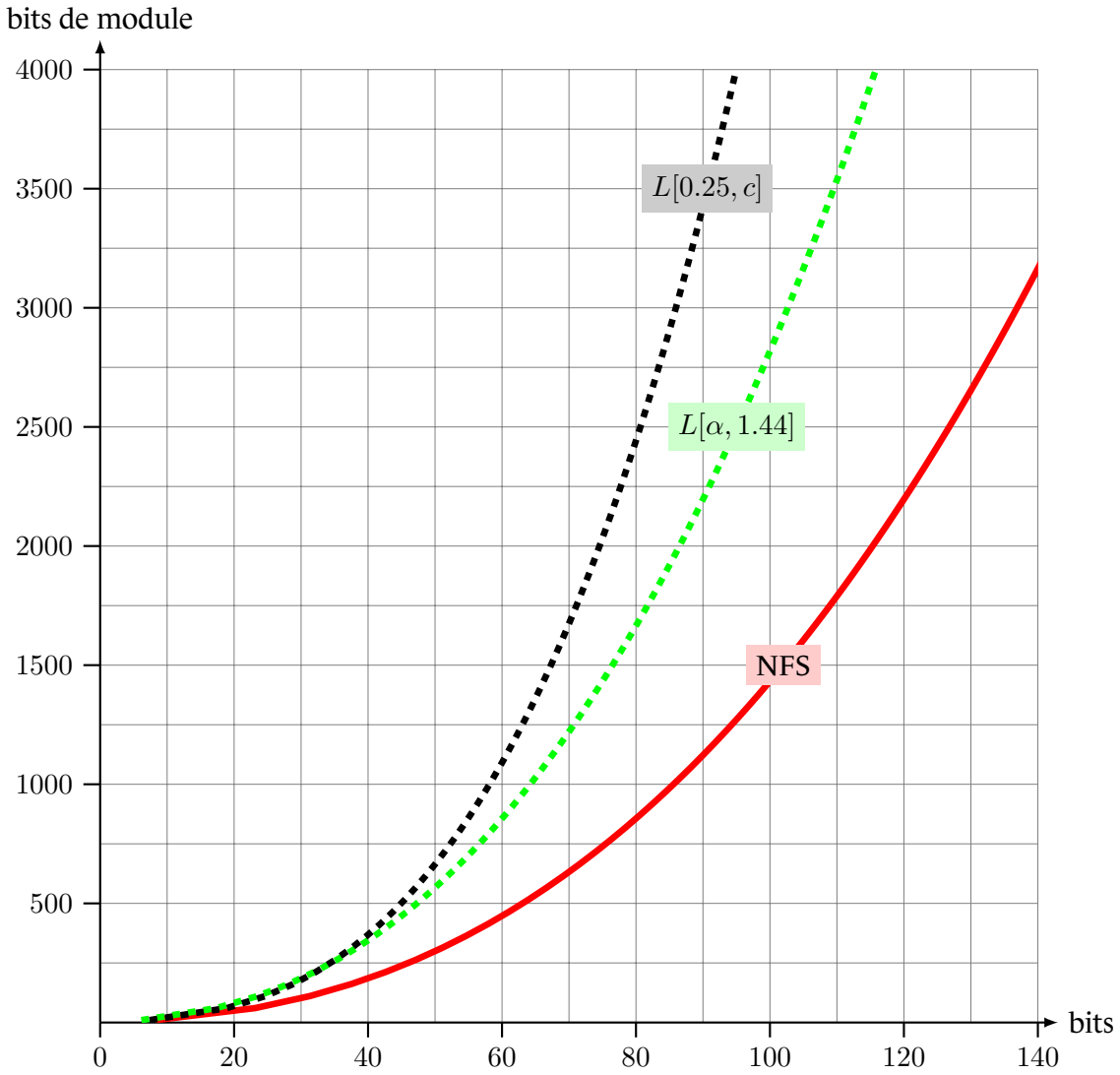

FIGURE B.6 – Équivalence entre taille de module asymétrique et taille de clé symétrique. Influence des progrès algorithmiques (gain de 25%) sur la constante (courbe verte) ou l'exposant (courbe noire) du crible algébrique.

Remarque

- Rechercher une sécurité équivalente en cryptographie asymétrique basée sur le problème de la factorisation ou sur celui sur logarithme discret à celle apportée en cryptographie symétrique par une clé secrète de plus de 128 bits mène à des tailles de clé très importantes, peu utilisables en pratique.
- Il est absurde de chercher à utiliser des modules d'une sécurité comparable à celle d'une clé symétrique de 256 bits.
- Il est courant de chercher à comparer la taille des clés symétriques et celle des clés asymétriques en estimant le « nombre de bits asymétriques » équivalent, en termes de sécurité, à un « bit symétrique ». En d'autres termes, ceci n'est rien d'autre que la pente de la courbe de la Figure B.6. On peut par conséquent énoncer en termes courants que : « pour des clés longues d'environ 1024 bits,

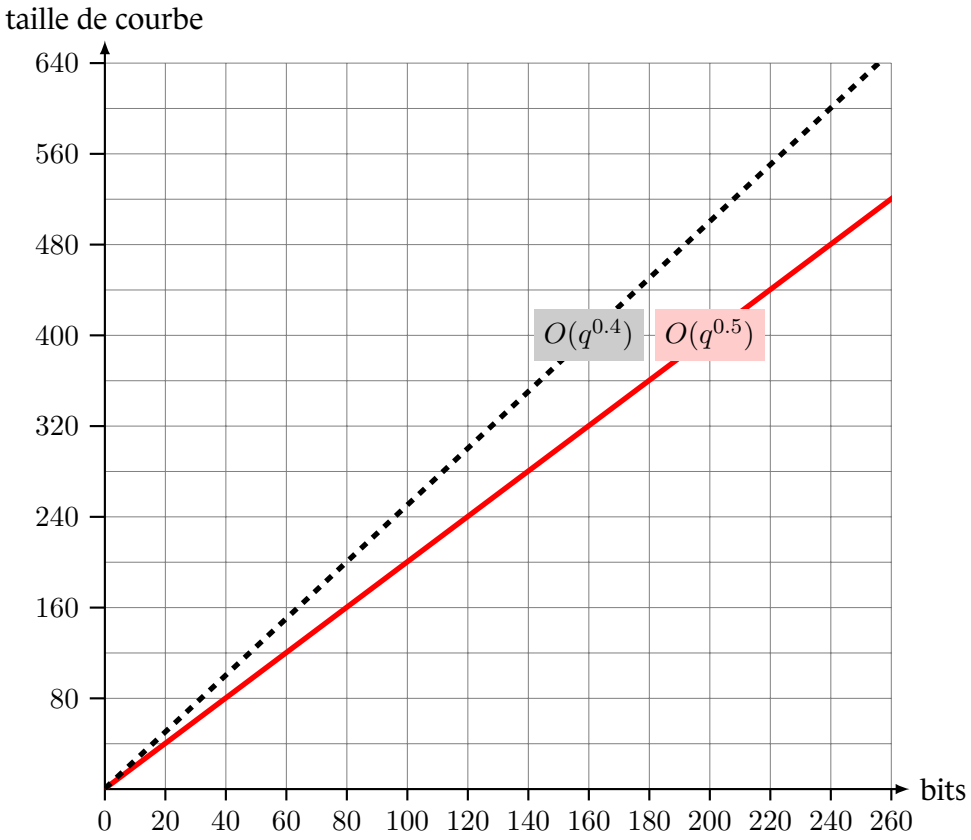

FIGURE B.7 – Équivalence entre dimensionnement de cryptosystème à base de courbe elliptique et taille de clé symétrique

un bit symétrique équivaut à 51 bits asymétriques », « pour des clés longues d'environ 2048 bits, un bit symétrique équivaut à 77 bits asymétriques ».

- On peut également reformuler ces affirmations sous la forme suivante : « pour doubler l'effort nécessaire afin de factoriser un module de 1024 bits, il convient d'utiliser un module de 1024+51=1075 bits ».

B.2.5 Équivalence de sécurité entre tailles de courbe elliptique et de clé symétrique

Le même procédé peut être appliqué pour comparer les clés symétriques et les cryptosystèmes à base de courbes elliptiques. On obtient les courbes de la figure B.7. Le graphique se lit de la façon suivante. Le nombre de bits des clés symétriques est donné en abscisse. L'ordonnée indique la taille en bits de la courbe elliptique offrant une sécurité équivalente. La courbe en rouge résulte de l'interprétation de la formule de complexité du meilleur algorithme connu (en $O(q^{0.5})$). La courbe noire en pointillé représente le scénario où un progrès algorithmique permet d'avoir une complexité en $O(q^{0.4})$.

Annexe C
Bibliographie

[Ble98] 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, 1998.

[Ble06] Daniel Bleichenbacher. Crypto Rump Session, 2006.

[BR94] Mihir Bellare and Philipp Rogaway. Optimal asymmetric encryption. In *Proceedings of Eurocrypt 1994*, volume 839 of *LNCS*, pages 92–111. Springer-Verlag, 1994.

[CFA+06] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[Len04] Arjen Lenstra. *Handbook of Information Security*, volume 2, chapter Key Lengths. Wiley, 2004.

[MvV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. http://www.cacr.math.uwaterloo.ca/hac.

[Sch01] Bruce Schneier. *Cryptographie appliquée*. Vuibert, 2001.

[Sin99] Simon Singh. *Histoire des codes secrets*. JC Lattès, 1999.

[Ste98] Jacques Stern. *La science du secret*. Éditions Odile Jacob, 1998.

[Sti01] Douglas Stinson. *Cryptographie, théorie et pratique*. Vuibert, 2001.

[Vau06] Serge Vaudenay. *A Classical Introduction to Cryptography : Applications for Communications Security*. Springer, 2006.

Liste des règles

Liste des recommandations

Liste des tableaux

Liste des figures

Index des termes et des acronymes utilisés

ANSSI-PG-083
Version 2.04 - 2020-01-01

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

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

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE