Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

Secrétariat général
de la défense
nationale
*Direction centrale de la sécurité
des systèmes d'information*

Paris, le 19 décembre 2006

N° 2741/SGDN/DCSSI/SDS/LCR

Mécanismes cryptographiques

Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques de niveau de robustesse *standard*

Version 1.10

(Remplace la version 1.02 N°2791/SGDN/DCSSI/SDS/AsTeC/CD-SF du 19 novembre 2004)

51 boulevard de La Tour-Maubourg - 75700 PARIS 07 SP - Tél 01 71 75 84 65 - Fax 01 71 75 84 90

Version	Modifications
Version 1.10	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 : o algorithmes de chiffrement par flot o algorithmes de chiffrement par bloc ($Recom_E$AlgoBloc-1) ; o modes opératoires de chiffrement ; o problèmes mathématiques asymétriques ; o génération d'aléa. - rajout d'une mention concernant le statut de SHA-1.
Version 1.02 du 25 octobre 2004	Première version applicable.

Laboratoire de Cryptographie de la DCSSI
SGDN/DCSSI/SDS/Crypto
51 boulevard de La Tour-Maubourg, 75700 Paris 07 SP
Crypto.DCSSI@sgdn.pm.gouv.fr

A. Table des matières

B. Introduction

La cryptographie moderne met à la disposition des concepteurs de systèmes d'informations 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 dans les années 1980 et 1990, 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 l'emploi correct des mécanismes. Ce document vise à expliciter ces règles ainsi que certaines recommandations.

B.1. Objectif du document

Nous décrivons dans ce document des règles et des recommandations relatives au premier niveau de robustesse cryptographique. Ces niveaux sont définis ci-dessous en B.3.

Note importante : ce document est édité en trois versions :

- **la présente version du document est non classifiée ; elle traite uniquement du premier niveau de robustesse, qualifié de « standard ». Ce document à pour vocation d'être largement diffusé, en particulier de manière électronique.**
- une seconde version du document est également non classifiée mais n'est pas diffusée par voie électronique ; elle traite des deux premiers niveaux de robustesse, « standard » et « renforcé ». Les informations qu'elle contient sont issues du savoir-faire et de l'état de l'art cryptographiques publics.
- Une troisième version traite de l'ensemble des trois niveaux de robustesse, niveau « élevé » compris. Le caractère sensible de certaines règles mettant en œuvre un principe de précaution justifie une classification du document.

Les **règles** définissent des principes qui doivent a priori être suivis par tout mécanisme visant un niveau de robustesse donné. L'observation de ces règles est une condition généralement nécessaire, mais non suffisante, à la reconnaissance du niveau de robustesse visé par le mécanisme. Par contre, le fait de suivre l'ensemble des règles, qui sont par nature très génériques, ne garantit pas la robustesse du mécanisme cryptographique ; seule une analyse spécifique permet de s'en assurer.

En plus des règles, nous définissons également des **recommandations**. Elles ont pour but de guider dans le choix de certaines primitives et d'inciter à certains dimensionnements permettant un gain considérable en termes de sécurité pour un coût souvent modique. Il va de soi qu'en tant que recommandations, leur application

peut être plus librement modulée en fonction d'autres impératifs tels que des contraintes de performance.

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, nous avons tenté de justifier chaque règle et recommandation contenue dans ce document. 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 se base sur l'état de l'art actuel en cryptographie. Sa prise en compte ainsi que certaines hypothèses classiques telles que la loi de Moore sur l'évolution de la puissance de calcul disponible permettent 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 des grands nombres.

Par ailleurs, l'estimation des niveaux de résistance qui seront nécessaires afin de garantir la sécurité d'informations à 10 ou 20 ans est délicate. Elle est cependant requise par de nombreuses applications comme la protection de la confidentialité de certaines informations ou les applications de signature électronique nécessitant 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, nous avons tenté de développer une analyse valable à plus de 15 ans, ce qui n'est pas sans risque. 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 trente ans, le plus ancien étant certainement le DES, standardisé en 1977.

B.2. Limites du champ d'application du document

Ce document traite des règles et recommandations concernant le choix et le dimensionnement de mécanismes cryptographiques. Il est complété par le document intitulé « Gestion des clés cryptographiques – Règles et recommandations concernant la gestion des clés utilisées dans des mécanismes cryptographiques » qui traite plus

spécifiquement des aspects liés à la création, la distribution et la manipulation de clés. Ces aspects ne sont donc pas traités par le présent document. Sont également explicitement exclus de ce document :

- 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 de niveau standard 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*, SPA, DPA, HODPA,…) ou par injection de faute (DFA),
- les aspects liés à l'éventuelle confidentialité des mécanismes ainsi qu'à la nécessité de diversifier ces mécanismes selon les contextes d'emploi,
- 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 menace 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 « Fourniture nécessaires à l'analyse de mécanismes cryptographiques – version 1.2 » N°2336/SGDN/DCSSI/SDS du 6 novembre 2006,
- les mécanismes non cryptographiques assurant cependant des fonctions de sécurité tels que l'emploi de mots de passes, 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.

B.3. Niveaux de robustesse cryptographiques

Deux **niveaux de robustesse cryptographique** sont définis dans ce document mais seules les règles et recommandations du premier niveau sont ensuite indiquées. Ces différents niveaux apparaissent naturellement lorsque l'on traite de la résistance de mécanismes cryptographiques. On désignera ces deux niveaux selon les termes employés dans le cadre de la qualification, à savoir « *standard* » et « *renforcé* ».

On notera que les règles et recommandations définies dans ce document ne tiennent pas compte des conditions d'emploi des mécanismes cryptographiques. Une telle approche peut surprendre car elle est différente de celle appliquée à l'évaluation du niveau de robustesse des produits et des systèmes. Elle a pour avantage majeur de prendre en compte la majorité des scénarios d'attaque, y compris ceux qui auraient pu

être oubliés ou négligés par les concepteurs d'un produit. Cette approche est de plus justifiée par l'existence de mécanismes dont la sécurité est assurée y compris en environnement très contraignant, sans pour autant nécessiter beaucoup plus de ressources que des primitives plus médiocres.

Lien entre niveau de robustesse cryptographique et niveau de qualification de produit

Les niveaux de robustesse s'appliquent aux mécanismes cryptographiques et non pas aux produits qui les emploient. L'objet de ce document est de contribuer à l'évaluation de la robustesse intrinsèque des mécanismes, sans tenir compte, dans cette phase initiale, des conditions particulières d'emploi du mécanisme dans un produit. Cette manière de procéder permet notamment de réutiliser les résultats d'expertises cryptographiques d'un produit à l'autre. L'expérience montre de plus que cette méthode d'expertise permet de détecter des voies d'attaque difficiles à concevoir si l'on se limite dès le départ aux seuls scénarios apparemment réalistes.

Le but d'un mécanisme cryptographique est cependant d'assurer une fonction de sécurité dans un environnement donné et en combinaison avec d'autres mécanismes. Il est par conséquent a priori envisageable, lorsque cela se justifie réellement, d'utiliser par exemple un mécanisme de niveau standard au sein d'un produit visant une qualification de niveau renforcé.

Dans le même ordre d'idée, il doit être bien entendu que la composition de mécanismes cryptographiques d'un niveau de robustesse donné ne garantit absolument pas l'obtention d'un niveau de robustesse global équivalent.

Le choix de primitives cryptographiques d'un niveau de robustesse au moins équivalent au niveau global visé pour le produit les employant, s'il est généralement fortement souhaitable, n'est donc ni nécessaire ni automatiquement suffisant pour garantir la sécurité de produits. Cet état de fait doit toujours rester présent à l'esprit des concepteurs et évaluateurs de produits cryptographiques. **Il est cependant sain en règle générale d'employer des mécanismes de niveau de robustesse au moins équivalent à celui du produit associé.**

Lien entre niveau de robustesse cryptographique et niveau de classification d'information traitée

Le fait de définir deux niveaux de robustesse de mécanismes ne doit pas non plus faire croire qu'ils doivent automatiquement être associés aux niveaux de classification d'information « *restricted*[1] » et « confidentiel », bien qu'une correspondance naturelle soit évidente. En accord avec ce qui est énoncé au paragraphe précédant, le choix du niveau de robustesse requis pour protéger des informations d'un niveau donné de classification ne peut être effectué qu'en tenant compte de la menace, de l'environnement, des contraintes d'implantation,…

[1] La mention « diffusion restreinte » n'ayant pas de valeur légale, nous préférons désigner par le terme anglais « *restricted* » cet éventuel premier niveau de classification.

Définitions des niveaux de robustesse

Nous définissons maintenant les deux premiers niveaux de robustesse cryptographique.

Niveau Standard (définition)

Le niveau de robustesse cryptographique dit « **standard** » est atteint par les mécanismes pour lesquels aucune attaque dans un modèle de sécurité réaliste n'a pu être exhibée, y compris de la part d'un attaquant disposant de compétences élevées et d'une capacité de calcul très importante, typiquement de l'ordre de grandeur de celle dont peuvent disposer les nations majeures.

À ce niveau, le point de vue de l'attaquant est privilégié ; un niveau de robustesse standard n'implique donc pas nécessairement que le mécanisme soit cryptographiquement entièrement satisfaisant. Ce niveau est le moins contraignant permettant de garantir une sécurité de nature cryptographique. Le risque majeur d'une telle approche est de s'exposer à des attaques réalistes non décelées au cours de l'évaluation.

Niveau Renforcé (définition)

Le niveau de robustesse cryptographique dit « **renforcé** » est atteint par les mécanismes de niveau standard qui appliquent, en plus, les règles de bonnes pratiques cryptographiques issues de l'état de l'art moderne en cryptographie académique.

Il est bien entendu très difficile d'énumérer de manière exhaustive ou de définir très précisément ces « règles de bonnes pratiques » qui font partie du savoir-faire des experts du domaine. La cryptographie moderne a cependant atteint un degré de maturité suffisant pour permettre en pratique de s'accorder facilement sur ces règles.

Absence de niveau de robustesse inférieur à « *standard* »

Le premier niveau de robustesse défini, à savoir le niveau standard, permet d'atteindre, en pratique, un niveau de sécurité qui peut être jugé très, voire même trop important. En particulier, nous ne définissons pas de niveau de robustesse que l'on pourrait qualifier de « moyen » au sens où les mécanismes d'un tel niveau seraient potentiellement attaquables de manière opérationnelle mais en nécessitant des moyens de calcul et/ou des capacités d'expertise importantes. Ce choix est justifié par le fait que la cryptographie moderne offre dans tous les domaines des mécanismes de niveau au moins standard dont le coût en termes de temps de calcul, de mémoire requise et de difficulté d'implantation est comparable à d'éventuels mécanismes qualifiés de « moyens ». Une originalité de la cryptographie est en effet de souvent permettre d'atteindre assez facilement un niveau de robustesse important pour un surcoût modique. De même, de nombreuses recommandations contenues dans ce document sont liées à l'observation qu'un gain de sécurité considérable peut souvent être obtenu pour un coût minime. Notons qu'un tel raisonnement ne peut bien évidemment pas s'appliquer à l'ensemble des domaines de la sécurité des systèmes

d'information mais semble au contraire plutôt spécifique à la cryptographie. Nous nous refusons donc à définir un niveau « moyen » dont le but serait uniquement de permettre la labellisation de mécanismes dont l'évaluation aurait montré qu'ils n'atteignent pas le niveau de robustesse standard.

En conclusion, ce document est conçu en prenant le parti d'employer des mécanismes cryptographiques apportant une sécurité forte. Cette approche est très classique en cryptographie. Les menaces considérées sont donc celles liées à des attaquants réalistes mais très puissants. Une conséquence directe est que nous ne cherchons pas à définir les tailles de clés ou de paramètres qui mettraient tout juste à l'abri d'attaquants moins puissants. De telles estimations visant à dimensionner un système cryptographique en se plaçant juste à la limite des capacités des attaquants « menaçants » peut éventuellement se justifier dans une analyse de risque « multi-origine ». Une telle approche est cependant bien trop spécifique pour être développée dans un document générique.

B.4. Organisation du document

Ce document est organisé de la manière suivante :

- l'ensemble des règles et recommandations sont regroupées dans le chapitre C, à partir de la page 13 ; elles sont repérées selon la codification suivante : les premières lettres (Règle ou Recom) indiquent si l'on à affaire à une règle ou une recommandation, l'indice suivant (S) indique le niveau de robustesse standard, le domaine d'application est ensuite précisé et, finalement, un chiffre permet de distinguer les règles d'une même catégorie. Par exemple, **Règle$_S$Fact-3** désigne la règle de niveau standard numéro 3 concernant le problème de la factorisation ;
- des références bibliographiques apparaissent en page 34,
- en annexe, un rappel non mathématique des principaux concepts cryptographiques nécessaires à la compréhension de ce document est proposé à partir de la page 36,
- toujours en annexe, des informations issues de publications du milieu académique sur le dimensionnement des mécanismes cryptographiques sont regroupées à partir de la page 56.

Ce document ne comporte volontairement aucun tableau récapitulatif des tailles minimales de paramètres requis pour le niveau de robustesse standard. 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.

B.5. Mise à jour et classification 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 annuelle semble à la fois réaliste et suffisante. La collecte de

commentaires et la diffusion des révisions sont effectuées par le laboratoire de cryptographie de la DCSSI[2].

[2] Pour toute correspondance, utiliser l'adresse courrier ou e-mail (non sécurisée) en page 3 du document.

C. Règles et recommandations

Les règles et recommandations contenues dans ce document sont organisées de manière très comparable aux rappels cryptographiques proposés en annexe F, à partir de la page 36. Elles s'adressent à un lecteur familier avec ces concepts qui ne sont par conséquent pas systématiquement rappelés.

C.1. Cryptographie symétrique

C.1.1. Chiffrement symétrique

C.1.1.1. Taille de clé symétrique

Nous définissons dans cette section les propriétés attendues de clés utilisées par des mécanismes symétriques. Par taille de clé, nous entendons le nombre de bits effectifs de la clé, i.e. le nombre de bits réellement variables[3]. 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 bits 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.

Niveau Standard

Règle$_S$CléSym-1. La taille minimale des clés symétriques utilisées jusqu'en 2010 est de 80 bits.

Règle$_S$CléSym-2. La taille minimale des clés symétriques devant être utilisées au-delà de 2010 est de 100 bits.

Recom$_S$CléSym-1. La taille minimale recommandée des clés symétriques est de 128 bits.

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

[3] 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.

F.1, G.1.1 et G.2.1) indiquent cependant que des clés de 80 bits devraient être inattaquables d'ici à 2010.

- L'emploi de clés de moins de 80 bits semble par contre plus 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 G.1.1).
- Une recherche exhaustive sur des clés de 100 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 tel que l'AES.

Remarques :

- 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.

C.1.1.2. 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 F.1.1.1). Les règles et recommandations concernant la taille effective de la clé ont été présentées au paragraphe précédent.

C.1.1.2.1. Taille de bloc

Niveau Standard

Règle$_S$BlocSym-1. La taille minimale des blocs de mécanismes de chiffrement par bloc est de 64 bits.

Recom$_S$BlocSym-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 quelconques 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* » ou en « *paradoxe des anniversaires* » (voir F.1.3, paragraphe 57) ; ceci signifie que certaines attaques deviennent opérationnelles dès que plus de $2^{n/2}$ blocs de message sont traités, où *n* désigne la taille en bits du bloc. Dans le cas de blocs de 64 bits, la limite de sécurité est donc seulement de quelques milliards 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.

C.1.1.2.2. 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 mais, 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 $\boldsymbol{N_{op}}$ d'opérations de calcul nécessaire à l'attaque, une opération étant équivalente à un chiffrement de bloc,
- le nombre $\boldsymbol{N_{bloc}}$ de blocs à faire chiffrer ou déchiffrer afin de réaliser l'attaque,
- la quantité $\boldsymbol{N_{mem}}$ de mémoire nécessaire, par exemple pour stocker des précalculs.

Niveau Standard

Règle$_S$AlgoBloc-1. Pour un algorithme de chiffrement ne devant pas être utilisé après 2010, aucune attaque nécessitant moins de $N_{op}=2^{80}$ opérations de calcul ne doit être connue.

Règle$_S$AlgoBloc-2. Pour un algorithme de chiffrement visant à être utilisé après 2010, aucune attaque nécessitant moins de $N_{op}=2^{100}$ opérations de calcul ne doit être connue.

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

Remarque importante :

- Les règles ne font pas mention du nombre de blocs $\boldsymbol{N_{bloc}}$ à faire chiffrer ou déchiffrer afin de réaliser l'attaque, ni de la quantité de mémoire $\boldsymbol{N_{mem}}$

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 atteint le niveau standard même s'il ne vérifie pas les règles **Règle$_S$AlgoBloc-1** et/ou **Règle$_S$AlgoBloc-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.
- Au niveau de robustesse standard, 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, au niveau standard, un gage de qualité très important.

Mécanisme(s) de niveau standard :

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

Remarques :

- Le triple DES, i.e. 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. En tant que mécanisme de chiffrement par bloc, le triple DES est de niveau standard. Ce mécanisme ne suit cependant pas les recommandations émises ci-dessus en termes de taille de clé et de taille de bloc. De plus, il convient 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.

C.1.1.2.3. 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 F.1.1.1, 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.

Niveau Standard

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 du niveau standard se veulent malgré tout relativement génériques.

Règle$_S$ModeChiff-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 $2^{n/2}$ où *n* est la taille en bits du bloc.

Recom$_S$ModeChiff-1. L'emploi d'un mode opératoire de chiffrement non déterministe est recommandé.

Recom$_S$ModeChiff-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.

Justification :

- De nombreux modes, tel que le CBC (voir F.1.1.1, page 40), 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. Descendre en dessous de cette borne compromettrait gravement la sécurité du mécanisme de chiffrement.
- Au niveau de robustesse standard, 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.
- Afin d'assurer la confidentialité des informations, un mode opératoire de chiffrement ne doit pas être déterministe afin 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 F.1.1.1) 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 juste 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(s) de niveau standard :

- Le mode de chiffrement CBC utilisant une primitive de chiffrement standard comme l'AES et des valeurs initiales aléatoirement choisies pour chaque message et transmises en clair est un mécanisme de chiffrement symétrique de niveau standard. Ce mécanisme est rappelé page 40.

C.1.1.3. Chiffrement par flot

Les algorithmes de chiffrement par flot[4,5] constituent l'autre grande famille de mécanismes de chiffrement symétrique (voir F.1.1.2).

C.1.1.3.1. 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). Toutefois, 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.

D'autre part, les règles et recommandations en termes de chiffrement par flot se fondent sur le constat qu'à ce jour une grande majorité des propositions académiques, même récentes, ont été cryptanalysées. Ces règles sont cependant susceptibles d'être revues si des avancées théoriques importantes sont effectuées dans le domaine du chiffrement par flot.

Niveau Standard

Règle$_S$ChiffFlot-1. Pour un algorithme de chiffrement par flot ne devant pas être utilisé après 2010, aucune attaque nécessitant moins de 2^{80} opérations de calcul ne doit être connue.

Règle$_S$ChiffFlot-2. Pour un algorithme de chiffrement par flot visant à être utilisé après 2010, aucune attaque nécessitant moins de 2^{100} opérations de calcul ne doit être connue.

Recom$_S$ChiffFlot-1. Il est recommandé d'employer des algorithmes de chiffrement par flot largement éprouvés dans le milieu académique.

Recom$_S$ChiffFlot-2. Il est recommandé d'employer des algorithmes par bloc et non des algorithmes de chiffrement par flot. Il est ainsi possible, si les propriétés du chiffrement par flot sont requises, d'utiliser un mode opératoire de chiffrement par bloc de niveau standard simulant un chiffrement par flot.

---

[4] « *stream cipher* » en anglais.

[5] 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 est le seul à disposer 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.

Remarque importante :

- 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 atteint le niveau standard même s'il ne vérifie pas les règles **Règle$_S$ChiffFlot-1** et/ou **Règle$_S$ChiffFlot-2**.

Justification :

- Au niveau de robustesse standard, 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.
- Au niveau standard, l'emploi d'algorithmes largement étudiés par la communauté académique offre un gage de qualité très important.
- Les algorithmes de chiffrement par flot 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 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. Nous recommandons par conséquent l'emploi de chiffrement par bloc, en combinaison avec des modes opératoires satisfaisants, 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 de chiffrement par bloc ayant un comportement comparable, 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 comparable à un chiffrement par flot.

C.1.2. 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.

Niveau Standard

Règle$_S$IntegSym-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 de niveau de robustesse standard.

Remarques :

- **Par confusion avec les modes opératoires de chiffrement, l'utilisation de « valeurs initiales » est souvent constatée pour des mécanismes d'intégrité tel que le CBC-MAC (voir page 44) ; 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, tel que le CBC-MAC, ne sont sûrs que si l'on traite au plus de l'ordre de $2^{h/2}$ blocs de messages clairs, où h désigne la taille en bits du bloc. Pour un mécanisme de chiffrement utilisant des blocs de h=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 page 44 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 facile 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 de niveau standard :

- 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 de niveau standard (à condition, bien entendu, de ne pas utiliser de valeur initiale). Ce mécanisme est rappelé page 44.

Mécanisme **n'atteignant pas** le niveau standard :

- Le mode d'intégrité CBC-MAC « *retail* » recommandé ci-dessus n'atteint pas le niveau standard 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.

C.1.3. Authentification d'entités

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.

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 clé. 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* », i.e. 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 « en 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.

D'autre part, comme rappelé en F.1.3, les mécanismes interactifs d'authentification d'entités reposent en général sur des mécanismes de génération d'aléa et de chiffrement ; les règles énoncées dans les sections C.1.1 et C.3.2 s'appliquent donc directement. Bien entendu, l'évaluation du niveau de robustesse global du mécanisme doit être effectuée avec soin, même si des primitives de niveau compatible sont employées.

C.2. Cryptographie asymétrique

Les mécanismes de cryptographie asymétrique utilisent tous un problème de base la plupart du temps issu de la théorie des nombres (voir F.2, page 46). L'emploi de problèmes réellement difficiles pour un attaquant est par conséquent primordial en termes de sécurité.

C.2.1. Problèmes mathématiques asymétriques

C.2.1.1. Factorisation

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

D'autre part, l'utilisation du problème de la factorisation est en général effectuée au moyen du cryptosystème RSA. Deux autres données, appelées « exposant public » et « exposant secret » sont alors utilisées.

Niveau Standard

Règle$_S$Fact-1. La taille minimale du module est de 1536 bits, pour une utilisation ne devant pas dépasser l'année 2010.

Règle$_S$Fact-2. La taille minimale du module est de 2048 bits, pour une utilisation ne devant pas dépasser l'année 2020.

Règle$_R$Fact-3. Pour une utilisation au-delà de 2020, la taille minimale du module est de 4096 bits.

Règle$_S$Fact-4. Les exposants secrets doivent être de même taille que le module.

Règle$_S$Fact-5. Pour les applications de chiffrement, les exposants publics doivent être strictement supérieurs à 2^{16}=65536.

Recom$_S$Fact-1. Il est recommandé d'employer des modules d'au moins 2048 bits, même pour une utilisation ne devant pas dépasser 2010.

Recom$_S$Fact-2. Il est recommandé, pour toute application, d'employer des exposants publics strictement supérieurs à 2^{16}=65536.

Recom$_S$Fact-3. Il est recommandé que les deux nombres premiers p et q constitutifs du module soient de même taille et générés aléatoirement.

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 G.1.2, page 57) mais en s'imposant une marge de sécurité, milite pour l'emploi de modules d'au moins 1536 bits, même si aucun module de 1024 bits n'a été officiellement factorisé à ce jour. Par conséquent, nous considérons que l'emploi de modules de 1024 bits constitue une prise de risque incompatible avec les critères du niveau de robustesse cryptographique standard.
- Les annexes G.1.2 et G.2.2 fournissent des informations complémentaires sur les analyses liées au problème de la factorisation.
- L'emploi d'exposants secrets particuliers afin d'améliorer les performances est à proscrire à cause des attaques pratiques publiées à ce sujet.
- L'emploi d'exposants publics très petits, tel 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 taille trop différente peut compromettre la sécurité. 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 parmi les nombres premiers de taille égale à la moitié de la taille du module.

C.2.1.2. Logarithme discret dans GF(p)

Le problème dit « du logarithme discret dans GF(p) » se base sur des calculs effectués dans le corps fini premier à *p* éléments, où *p* est un nombre premier.

Niveau Standard

Règle$_S$Logp-1. La taille minimale de modules premiers est de 1536 bits pour une utilisation ne devant pas dépasser l'année 2010.

Règle$_S$Logp-2. La taille minimale de modules premiers est de 2048 bits pour une utilisation ne devant pas dépasser l'année 2020.

Règle$_S$Logp-3. Pour une utilisation au delà de 2020, la taille minimale de modules premiers est de 4096 bits.

Règle$_S$Logp-4. Pour une utilisation ne devant pas dépasser 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 160 bits.

Règle$_S$Logp-5. Pour une utilisation au-delà de 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 256 bits.

Recom$_S$Logp-1. Il est recommandé d'employer des modules premiers d'au moins 2048 bits, même pour une utilisation ne devant pas dépasser 2010.

Recom$_S$Logp-2. Il est recommandé d'employer des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 256 bits, même pour une utilisation ne devant pas dépasser 2010.

Justification :

- Le problème du logarithme discret 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 G.1.2) 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.

C.2.1.3. Logarithme discret dans GF(2^n)

Le problème dit « du logarithme discret dans GF(2^n) » se base sur des calculs effectués dans le corps fini à 2^n éléments, où *n* est un entier.

Niveau Standard

Règle$_S$Log2-1. La valeur minimale de *n* est de 2048 bits, pour une utilisation ne devant pas dépasser l'année 2020.

Règle$_S$Log2-2. Pour une utilisation au-delà de 2020, la valeur minimale de *n* est de 4096 bits.

Règle$_S$Log2-3. Pour une utilisation ne devant pas dépasser 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 160 bits.

Règle$_S$Log2-4. Pour une utilisation au-delà de 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 256 bits.

Recom$_S$Log2-1. Il est recommandé de privilégier des primitives à clé publique ne se fondant pas sur le problème du logarithme discret dans GF(2^n).

Justification :

- Le problème du logarithme discret dans GF(2^n), bien que difficile, semble plus facile, à taille de corps égale, que dans GF(p). Dans de nombreuses applications, l'emploi de GF(2^n) par rapport à GF(p) n'est justifié que par des raisons d'efficacité de calcul. Pour des raisons de sécurité, nous conseillons par conséquent de préférer l'emploi de GF(p).

C.2.1.4. Courbes elliptiques définies dans GF(p)

Il est également possible de définir des problèmes de logarithme discret dans des structures plus complexes pour lesquels 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 (GF(p)) ou binaire (GF(2^n)).

Niveau Standard

Règle$_S$ECp-1. Pour une utilisation ne devant pas dépasser 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 160 bits.

Règle$_S$ECp-2. Pour une utilisation au-delà de 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 256 bits.

Règle$_S$ECp-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 dans GF(p), ce problème devra vérifier les règles correspondantes pour le niveau standard.

Justification :

- Le problème du logarithme discret sur courbe elliptique semble aujourd'hui un problème très difficile. Il permet d'obtenir, pour des tailles de paramètre 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.

Mécanisme(s) de niveau standard :

- L'emploi des courbes P-256, P-384 et P-521 définies dans le FIPS 186-2 du 27/01/2000 est recommandé au niveau standard.

C.2.1.5. Courbes elliptiques définies dans GF(2^n)

Niveau Standard

Règle$_S$EC2-1. Pour une utilisation ne devant pas dépasser 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 160 bits.

Règle$_S$EC2-2. Pour une utilisation au-delà de 2010, on emploiera des sous-groupes dont l'ordre est multiple d'un nombre premier d'au moins 256 bits.

Règle$_S$EC2-3. Le paramètre *n* doit être un nombre premier.

Règle$_S$EC2-4. 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 dans GF(2^n), ce problème devra vérifier les règles correspondantes pour le niveau standard.

Justification :

- L'emploi de *n* composé réduit considérablement la difficulté du calcul de logarithme discret et affaiblit donc le mécanisme correspondant.
- Au niveau de robustesse standard, nous ne différentions pas les courbes elliptiques définies sur GF(p) de celles définies sur GF(2^n).

Mécanisme(s) de niveau standard :

- L'emploi des courbes B-283, B-409 et B-571 définies dans le FIPS 186-2 du 27/01/2000 est recommandé au niveau standard.

C.2.1.6. Autres problèmes

Plusieurs autres problèmes ont été proposés dans le milieu académique afin de fournir des alternatives aux problèmes cités précédemment. Ces problèmes n'étant que très peu utilisés aujourd'hui en pratique, leur sécurité est à traiter au cas par cas.

C.2.2. 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.

Niveau Standard

Recom$_S$ChiffAsym-1. Il est recommandé d'employer des mécanismes de chiffrement asymétrique disposant d'une preuve de sécurité.

Mécanisme(s) de niveau standard :

- Le mécanisme de chiffrement asymétrique RSAES-OAEP défini dans le document PKCS#1 v2.1 est de niveau standard à condition de respecter les règles **Règle$_S$Fact-1**, **Règle$_S$Fact-2**, **Règle$_S$Fact-3** et **Règle$_S$Fact-4**.

C.2.3. Signature numérique

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 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.

Les schémas de signature utilisent de plus en général des fonctions de hachage dont le niveau de robustesse (voir C.3.1) doit bien entendu être en accord avec le niveau de robustesse souhaité pour le mécanisme de signature.

Niveau Standard

Recom$_S$SignAsym-1. Il est recommandé d'employer des mécanismes de signature asymétrique disposant d'une preuve de sécurité.

Mécanisme(s) de niveau standard :

- Le mécanisme de signature asymétrique RSASSA-PSS[6] défini dans le document PKCS#1 v2.1 est de niveau standard à condition de respecter les règles **Règle$_S$Fact-1**, **Règle$_S$Fact-2**, **Règle$_S$Fact-3** et **Règle$_S$Fact-4**.
- Le mécanisme de signature asymétrique ECDSA, à base de courbes elliptiques, défini dans le FIPS 186-2 est de niveau standard s'il utilise l'une des courbes P-256, P-384, P-521, B-283, B-409 ou B-571.

[6] RSASSA-PSS: "*RSA Signature Scheme with Appendix – Provably Secure encoding method for digital Signatures".*

C.2.4. Authentification asymétrique d'entités et échange de clés

Comme il est rappelé en F.2.3, les mécanismes interactifs d'authentification d'entités et d'échange de clés reposent en général sur des mécanismes de génération d'aléa, de hachage et de chiffrement ou de signature à clé publique. Les règles énoncées dans les sections C.2.2, C.2.3, C.3.1 et C.3.2 s'appliquent donc directement.

C.3. Autres primitives cryptographiques

C.3.1. Fonction de hachage

Les fonctions de hachage cryptographiques doivent avoir plusieurs propriétés telle que l'absence de « collisions » (voir F.3.1). De telles collisions peuvent cependant toujours être trouvées au moyen d'attaques génériques dites « par paradoxe des anniversaires ». Un des buts lors de la conception d'une fonction de hachage est par conséquent de faire en sorte que cette attaque soit la meilleure possible. En pratique, ceci impose que la taille des empreintes soit le double de celle d'une clé symétrique permettant d'atteindre le même niveau de robustesse.

D'autre part, les fonctions de hachage sont généralement 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.

Niveau Standard

Règle$_S$Hash-1. La taille minimale des empreintes générées par une fonction de hachage ne devant pas être utilisée au-delà de 2010 est de 160 bits.

Règle$_S$Hash-2. La taille minimale des empreintes générées par une fonction de hachage devant être utilisée au-delà de 2010 est de 256 bits.

Règle$_S$Hash-3. 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.

Recom$_S$Hash-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.
- Nous insistons 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 collision. Dans certaines applications, cette propriété semble trop forte car seul le calcul de « pré-image » doit être irréalisable. Nous considérons cependant qu'indépendamment de tout contexte une fonction de hachage ne peut être reconnue d'un niveau de robustesse donné que vis-à-vis de la propriété d'absence de collision. Ceci n'est pas contradictoire avec la possibilité qu'un mécanisme employant une fonction de hachage qui ne soit pas de niveau standard puisse quand même être jugée, dans sa globalité, comme atteignant le niveau standard.

Mécanisme de niveau standard :

- Le mécanisme de hachage SHA-256 défini dans le FIPS 180-2 est de niveau standard.

Mécanisme **n'atteignant pas** le niveau standard :

- Le mécanisme de hachage SHA-1 défini dans le FIPS 180-2 a récemment fait l'objet d'une attaque en recherche de collision. La complexité de cette attaque est estimée à 2^{63}, c'est à dire inférieure à 2^{80}. Même si cette attaque n'a pas conduit, au moment de la rédaction de ce document, au calcul d'une collision explicite, sa seule existence montre une faille sérieuse dans la sécurité de cette fonction. Le mécanisme de hachage SHA-1 n'atteint donc pas le niveau standard.

C.3.2. Génération d'aléa cryptographique

Comme expliqué en F.3.2, la génération d'aléa consiste en général en la combinaison d'un processus physique ayant un comportement au moins partiellement aléatoire et d'une étape de « **retraitement algorithmique** » employant un générateur pseudo-aléatoire. L'analyse du processus physique sort du cadre de la cryptographie. Par contre, l'analyse du retraitement algorithmique peut être effectuée au moyen de méthodes classiquement employées en cryptographie en tenant compte d'hypothèses sur la « qualité » de l'aléa physique.

Dans la suite, on distinguera deux types de sources physiques. Le premier rassemble des éléments provenant de l'environnement du mécanisme et présentant des variations susceptibles d'être partiellement aléatoires (horloge interne d'un PC, mouvements de la souris, intervalles de frappe au clavier, signaux d'interruption matériels, etc.). Ce type de source est appelé « **éléments extérieurs** ». Il n'est pas considéré comme une véritable source d'aléa continu et ne participe donc pas directement à la sécurité du générateur d'aléa. Pour cette raison, il ne figure pas sur la représentation de la Figure 1. Il est cependant possible de se servir des données issues de ce type de source comme graine initiale dans un mécanisme de retraitement algorithmique pseudo-aléatoire. Le second type de source physique consiste en des mécanismes physiques spécialement conçus pour générer de l'aléa tout au long de la durée de vie du système. Ce type de source est appelé « **générateur physique d'aléa** » et doit vérifier certaines règles et recommandations détaillées en C.3.2.2.

Combiné à un générateur physique d'aléa, il est possible, et même recommandé, d'utiliser des « **éléments secrets** » comparables à des clés secrètes

spécifiques à chaque équipement et donc non partagées. Enfin, afin d'éviter toute forme de réinitialisation du générateur, l'emploi d'un accumulateur consistant en une « **mémoire non volatile** » peut être très utile.

On obtient alors l'architecture suivante dans laquelle tous les composants ne sont pas forcément nécessaires pour chaque niveau de robustesse visé :

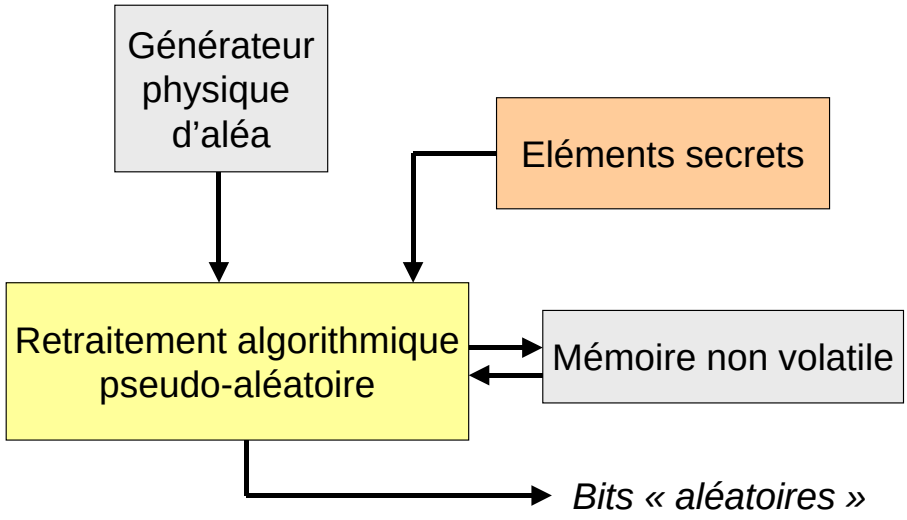


Figure 1. Architecture générique de générateur d'aléa

Les règles et recommandations en termes de générateur d'aléa se basent 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.

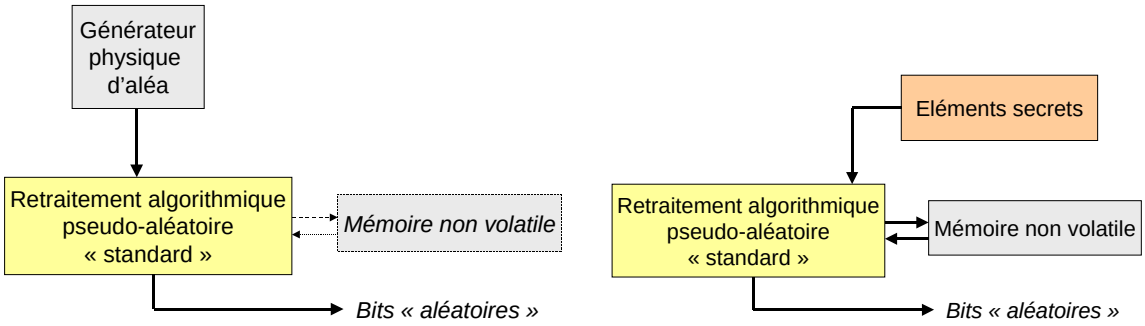
C.3.2.1. Architecture d'un générateur d'aléa

Niveau Standard

Règle$_S$ArchiGDA-1. Un retraitement algorithmique pseudo-aléatoire doit être employé.

Règle$_S$ArchiGDA-2. Un générateur physique d'aléa ou des éléments secrets combinés avec une mémoire non volatile doivent être employés.

Recom$_S$ArchiGDA-1. Dans tous les cas, l'emploi d'une mémoire non volatile comme accumulateur est recommandé.



Justification :

- L'emploi d'un générateur physique non retraité est exclu. De même, l'emploi de retraitements élémentaires, souvent qualifiés de « lissage », est également exclu.

Remarques :

- Le retraitement algorithmique pseudo-aléatoire 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'un « germe » aléatoire de petite taille. En particulier, sans la part d'aléa initial apportée par le germe, ce dernier pouvant par exemple constituer les éléments secrets mentionnés ci-dessus, le comportement de l'algorithme de retraitement est parfaitement prévisible.
- 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 même règles d'implantation aux algorithmes de retraitement que pour tout autre mécanisme cryptographique.

Mécanisme(s) de niveau standard :

- Le mécanisme de retraitement cryptographique d'aléa défini dans la norme ANSI X9.31, et rappelé page 52, est de niveau standard s'il est utilisé avec l'AES comme mécanisme de chiffrement par bloc.

C.3.2.2. Générateur physique d'aléa

Le générateur physique d'aléa est conçu pour générer de l'aléa tout au long de la durée de vie du système. Il faut donc garantir, d'une certaine manière, la qualité et la fiabilité de cet aléa.

Niveau Standard

Règle$_S$GVA-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.

Règle$_S$GVA-2. Des tests statistiques en sortie du générateur physique ne doivent pas faire apparaître de défauts dans l'aléa généré.

Recom$_S$GVA-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 au niveau standard, 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 RègleGVA-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.

C.3.2.3. Retraitement algorithmique pseudo-aléatoire

Niveau Standard

Règle$_S$AlgoGDA-1. Le retraitement algorithmique pseudo-aléatoire employé doit être de niveau standard.

Justification :

- Pour être au niveau standard, le mécanisme de retraitement pseudo-aléatoire cryptographique employé se doit d'utiliser des algorithmes de niveau de robustesse au moins standard (fonction de hachage, chiffrement par bloc,…).

C.3.3. 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 est donc traité dans un document qui lui est spécifiquement consacré. Ce document s'intitule « Gestion des clés cryptographiques – Règles et recommandations concernant la gestion des clés utilisées dans des mécanismes cryptographiques ». Il est également décliné en trois versions qui suivent les mêmes principes de diffusion que le présent document. Afin de donner un premier aperçu du problème de la gestion des clés, quelques règles sont énoncées dans la suite de cette section. Ces règles ne sont pas exhaustives et le lecteur est invité à se reporter au document spécialisé pour connaître l'étendue des règles régissant la gestion de clés.

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. Ce problème est également traité dans le document « Gestion des clés cryptographiques ».

C.3.3.1. 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 F.3.3.1). Nous désignons de telles clés 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.

Niveau Standard

Règle$_S$GestSym-1. L'emploi d'une même clé pour plus d'un usage est exclu.

Règle$_S$GestSym-2. Les éventuelles clés différenciées utilisées avec un mécanisme de niveau de robustesse standard doivent être générées en utilisant un mécanisme de diversification de niveau standard.

Règle$_S$GestSym-3. Les éventuelles clés dérivées doivent être générées en utilisant un mécanisme de diversification de niveau « cohérent ».

Recom$_S$GestSym-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 de niveau de robustesse cryptographique standard.
- La règle **Règle$_S$GestSym-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 privilégiées d'attaque.

C.3.3.2. Bi-clés asymétriques

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

Niveau Standard

Règle$_S$GestAsym-1. L'emploi d'un même bi-clé à plus d'un usage est exclu.

Règle$_S$GestAsym-2. Les clés hiérarchiquement importantes, telles que les clés racine, doivent être générées et utilisées par des mécanismes de niveau « cohérent ».

Justification :

- L'emploi d'un même bi-clé à plus d'un usage, par exemple pour chiffrer et signer, est une source d'erreurs graves.
- La règle **Règle$_S$Kasym-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.

D. Bibliographie

[1] W. Diffie et M. Hellman. *New directions in cryptography*. IEEE Transactions on Information Theory, 22 (1976), 644-654.

[2] Arjen Lenstra et Eric Verheul. *Selecting Cryptographic Key Sizes*. Journal of Cryptology, volume 14, numéro 4, pp.255-293, Springer-Verlag, décembre 2001.

[3] Alfred Menezes, Paul J. van Oorschot et Scott A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997. (**`http://www.cacr.math.uwaterloo.ca/hac`**)

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

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

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

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

E. Acronymes, abréviations et définitions

	Définition	Page(s)
AES	Advanced Encryption Standard	15
ANSI	American National Standard Institute	
AsTeC	Cellule d'Assistance Technique en Cryptographie de la DCSSI	
CBC	Cipher-Block Chaining mode of operation	16
CBC-MAC	Cipher-Block Chaining Message Authentication Code	19
CCA	Chosen Ciphertext Attack	16
CFB	Cipher Feedback mode of operation	16
CTR	Counter mode of operation	16
DCSSI	Direction Centrale de la Sécurité des Systèmes d'Information	
DES	Data Encryption Standard	15
DFA	Differential Fault Analysis	7
DPA	Differential Power Analysis	7
DSA	Digital Signature Algorithm	26
ECB	Electronic Codebook mode of operation	16
ECDSA	Elliptic Curve Digital Signature Algorithm	26
FIPS	Federal Information Processing Standard (standard du NIST)	
$GF(2^n)$	Corps fini (Galois Field) binaire à 2^n éléments	23, 25
GF(p)	Corps fini (Galois Field) premier à *p* éléments	23, 24
HODPA	High Order Differential Power Analysis	7
MAC	Message Authentication Code	19
NIST	National Institute of Standards and Technology	
OAEP	Optimal Asymmetric Encryption Padding	25
OFB	Output Feedback mode of operation	16
PIN	Personnal Identification Number	39
PKCS	Public Key Cryptography Standards	
PSS	Provably Secure encoding method for digital Signatures	26
RSA	"Rivest-Shamir-Adleman" (cryptographie asymétrique)	25, 26
SGDN	Secrétariat Général de la Défense Nationale	
SHA	Secure Hash Algorithm	27
SPA	Simple Power Analysis	7

F. Définitions et concepts

1 L'objet de ce chapitre est de rappeler les définitions et concepts essentiels en cryptographie afin de bien comprendre les règles et recommandations émises dans ce document. Ces rappels couvrent le strict minimum. Ils sont bien évidemment sommaires et volontairement non mathématiques. Pour plus de détails, on pourra par exemple consulter l'excellent ouvrage de référence [3]. On pourra également trouver de nombreux éléments de réponse dans [6], [4] et [7]. Une approche plus historique est présentée dans [5].

2 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.

3 La **cryptologie**, science 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, elle est récemment entrée dans une nouvelle ère sous la double influence de très importants progrès de nature théorique et technique d'une part et de l'émergence de nouveaux besoins issus de ce qu'il est convenu d'appeler la société de l'information, d'autre part.

4 La cryptologie traite de la conception, de la sécurité et de l'emploi de mécanismes cryptographiques. Ces derniers sont souvent qualifiés de **primitives** cryptographiques afin d'insister sur leur caractère élémentaire, indivisible, mais également sur la nécessité de les assembler afin d'obtenir des systèmes complets capables de rendre des services de sécurité. On parle également **d'algorithmes** cryptographiques afin d'indiquer que la précision de description visée doit être suffisant pour permettre leur implantation.

5 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 la confidentialité, l'intégrité ou l'authenticité de l'information ainsi que des mécanismes techniquement proches tel que l'identification d'entités. 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.

6 Une autre manière de séparer la cryptologie en deux branches, orthogonalement à la précédente, est de nature plus technique. On distingue habituellement la cryptographie dite **symétrique**, encore qualifiée de **conventionnelle** ou de cryptographie **à clé secrète**, de la cryptographie dite **asymétrique**, ou **à clé publique**. Cette séparation est issue de l'article fondateur de W. Diffie et M. Hellman [1] qui, en 1976, ont profondément révolutionné la cryptologie en suggérant par exemple que pour chiffrer un message à l'attention d'un destinataire donné il ne soit pas forcément nécessaire de partager au préalable un secret avec lui. Une nouvelle approche de la cryptologie, qualifiée d'asymétrique, était née. Depuis lors on classe souvent les primitives selon leur nature symétrique ou asymétrique. Dans la suite de ce chapitre nous avons choisi cette première séparation afin de rappeler les différentes primitives cryptographiques.

7 Une seconde dimension de classification des primitives cryptographiques se base sur 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 que nous venons de citer sont, de loin, les principales traitées par des moyens cryptographiques.

8 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.

9 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.

10 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.

11 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, un message signé devant en général le demeurer, même si le signataire change ensuite d'avis.

12 Nous pouvons maintenant aborder les principales primitives offertes par la cryptographie moderne. Le tableau ci-dessous 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.

<table>
<tr><th colspan="2"></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 (F.1.1.1)
ou par flot (F.1.1.2)</td><td>Chiffrement à clé publique (F.2.1)</td></tr>
<tr><td>Échange de clé (F.2.3)</td></tr>
<tr><td colspan="2">Intégrité</td><td rowspan="3">Code d'authentification de message (F.1.2)</td><td rowspan="4">Signature numérique
(F.2.2, F.2.3)</td></tr>
<tr><td rowspan="2">Authentification</td><td>De données</td></tr>
<tr><td>d'entités</td><td>Défi-réponse (F.1.3)</td></tr>
<tr><td colspan="2">Non-répudiation</td><td></td></tr>
</table>

F.1. Cryptographie symétrique

13 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.

14 Une clé secrète est généralement une suite quelconque de bits, i.e. 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 semble réalisable par des moyens informatiques raisonnables. La capacité à effectuer 2^n opérations fournit donc une borne inférieure au dimensionnement des systèmes symétriques.

15 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, nous avons rassemblé dans le tableau ci-dessous quelques exemples numériques concrets :

n	2^n
32	$2^{32} \approx$ nombre d'Hommes sur Terre
46	$2^{46} \approx$ distance Terre – Soleil en millimètres $2^{46} \approx$ nombre d'opérations effectuées en une journée à raison d'un milliard d'opérations par seconde (1GHz)
55	$2^{55} \approx$ nombre d'opérations effectuées en une année à raison d'un milliard d'opérations par seconde (1GHz)
82	$2^{82} \approx$ masse de la Terre en kilogrammes
90	$2^{90} \approx$ nombre d'opérations effectuées en 15 milliards d'années (âge de l'univers) à raison d'un milliard d'opérations par seconde (1GHz)
155	$2^{155} \approx$ nombre de molécules d'eau sur Terre
256	$2^{256} \approx$ nombre d'électrons dans l'univers

16 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 que la capacité d'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.

F.1.1. Chiffrement symétrique

17 Les primitives symétriques les plus connues sont les algorithmes de chiffrement[7]. Ils 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.

18 Les algorithmes de chiffrement 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.

19 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. Le paragraphe 39 fournit un exemple concret d'une telle attaque.

20 Les méthodes de chiffrement symétrique, encore appelées « 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.

F.1.1.1. Chiffrement par bloc

21 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.

22 Les principales propriétés attendues d'un mécanisme de chiffrement par bloc sont d'être facilement inversible si l'on dispose de la clé secrète de chiffrement, on parle alors de déchiffrement, mais impossible en pratique si l'on ne dispose pas de cette information. Ainsi, seuls les détenteurs de la clé secrète sont capables de transformer des données claires en données chiffrées ou, inversement, des données chiffrées en données claires.

---

[7] Le seul terme admis en français est celui de « chiffrement ». On entend cependant souvent parler de « cryptage », voire de « chiffrage », mais ces mots sont incorrects. L'opération inverse du chiffrement est le « déchiffrement ». Il est cependant admis de désigner par « décryptage », ou « décryptement », un déchiffrement effectué de manière illicite par un attaquant, typiquement sans disposer de la clé secrète mais après avoir trouvé une faille dans l'algorithme de chiffrement.

23 L'exemple le plus connu d'algorithme de chiffrement par bloc est le DES (*Data Encryption Standard*) défini en 1977 par le NIST, institut de standardisation 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 réaliste, au moyen d'une machine dédiée, en quelques heures seulement.

24 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.

25 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.

26 À ce niveau, il convient de comprendre précisément ce que permet de faire un algorithme de chiffrement par bloc, et surtout de ne pas faire. 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 ».

27 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 chiffrés 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.

28 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…

29 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 le traitement initial du message en une suite de blocs ainsi que le mode de chiffrement de ces blocs afin d'obtenir au final 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, i.e. d'introduire une certaine part de variabilité aléatoire.

30 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 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, … x_t les t blocs de message clair ainsi obtenus. On choisit ensuite un bloc aléatoire, noté *IV* pour « *initial value* », indépendant du message et des précédents chiffrements. Si l'on note $E_K(x)$ le résultat du chiffrement du bloc x avec la clé K, le chiffré (c_0, c_1, c_2, … c_t) du message est obtenu en posant $c_0=IV$ et en calculant successivement $c_i=E_K(c_{i-1}\oplus x_i)$ 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 2.

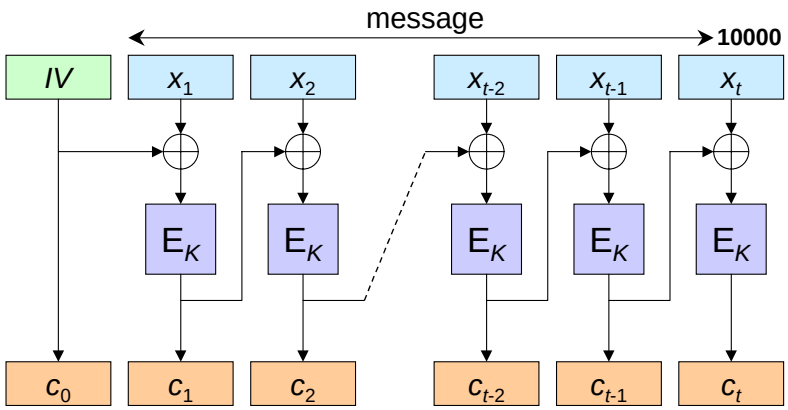


Figure 2. Mode opératoire CBC

31 Cet exemple de mode opératoire illustre la phase initiale de « *padding* », ou « bourrage », 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.

32 Afin de déchiffrer un message (c_0, c_1, c_2, … c_t), il suffit de calculer $x_i=c_{i-1}\oplus D_K(c_i)$ pour i allant de 1 à t, $D_K(c)$ 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 chiffrés sont perdus en cours de transmission, il suffit d'obtenir deux blocs successifs intacts afin de pouvoir reprendre correctement le déchiffrement.

F.1.1.2. Chiffrement par flot

33 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 combinés avec une séquence de bits dérivée de la clé secrète.

34 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, … ainsi qu'une clé également vue comme une suite de bits k_1, k_2, k_3, … le chiffré du message est alors très simplement obtenu au moyen de l'opération de « ou-exclusif bit à bit » 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 $0\oplus0=0$, $0\oplus1=1\oplus0=1$ et $1\oplus1=0$, seule cette dernière opération diffère de l'addition classique.

35 Afin de déchiffrer, il suffit d'appliquer la même opération d'addition 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.

36 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 est utilisée 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.

37 Il est assez facile de montrer que le cryptosystème de Vernam est le seul mécanisme de chiffrement permettant d'assurer une confidentialité parfaite des données. Claude Shannon en avait déduit en 1949 l'impossibilité de réaliser des primitives cryptographiques 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.

38 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[8]. Le déchiffrement agit ensuite de même, en générant la même suite à partir de la clé secrète.

39 En écho à la mise en garde du paragraphe 19, l'exemple du 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, i.e. 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.

40 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 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 au moyen de deux mécanismes séparés mais combinés ou bien 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.

41 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-**

[8] Plus précisément, cette approche conduit à une classe particulière d'algorithmes de chiffrement par flot appelée « *binary additive stream cipher* ».

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.

42 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. De plus, certains modes opératoires de chiffrement par bloc, tel que le mode CBC vu précédemment, ont également cette propriété d'auto-synchronisation à condition que des blocs entiers soient perdus (voir paragraphe 32).

F.1.1.3. Sécurité du chiffrement

43 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 ceci 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.

44 Afin de préciser la difficulté d'une définition précise de ce que l'on attend du chiffrement, reprenons l'exemple de la randomisation vue au paragraphe 29. 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. Nous avons cependant déjà remarqué qu'un tel chiffrement déterministe, 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.

45 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'expérience de pensée suivante : si un mécanisme de chiffrement est tel qu'en proposant deux messages différents[9] de son choix, 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.

46 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.

47 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 potentiellement très puissant forme . Ces modèles offrent donc les meilleures garanties. Le niveau de sécurité le plus élevé correspond au modèle de « l'inditinguabilité face aux attaques à chiffrés choisis adaptatives » noté par l'acronyme IND-CCA2.

[9] différents mais de même taille

48 Les modèles de sécurité utilisés en cryptographie moderne peuvent paraître exagérés mais ils sont justifiés par le fait que dans certaines circonstances, les attaquants contre lesquels on peut vouloir se prémunir ont une grande marge de manœuvre dans leurs attaques. 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 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.

F.1.2. Authentification et intégrité de messages

49 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. Nous avons déjà mentionné l'inadéquation des mécanismes de chiffrement afin de garantir une telle intégrité des données. Notons de plus 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.

50 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 transmis, potentiellement corrompu. Le résultat obtenu est comparé au MAC transmis ; s'ils sont identiques, il est extrêmement probable que les données sont intègres.

51 Nous insistons 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 au paragraphe 30, au message, sans utiliser de vecteur d'initialisation (*IV*), et de ne conserver comme 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 3. 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 c'est loin d'être le cas.

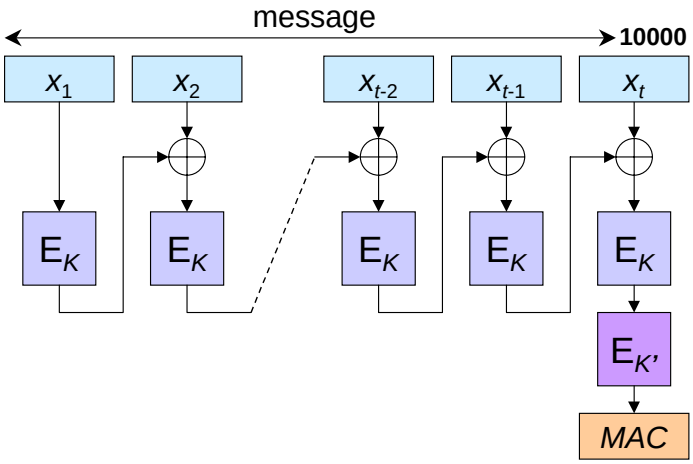


Figure 3. CBC-MAC

52 La protection de l'intégrité d'un message garantit également dans une certaine mesure son authenticité car la connaissance de la clé secrète est nécessaire afin de générer des MACs 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.

53 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. Seuls les mécanismes asymétriques, évitant justement que la connaissance de la clé secrète soit nécessaire afin de vérifier la validité de la signature, peuvent réellement rendre de tels services de sécurité.

54 Notons enfin qu'un code d'authentification de message valide permet 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.

F.1.3. Authentification d'entités

55 À partir des primitives de chiffrement et de calcul de MAC que nous venons de décrire, 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.

56 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, est bien 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.

57 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 de l'ordre de $2^{n/2}$ 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.

58 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 nous allons maintenant décrire les bases.

F.2. Cryptographie asymétrique

59 L'idée majeure de la cryptographie asymétrique est que certaines opérations sont faciles à réaliser mais difficiles à inverser. 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 le monde mathématique, il existe de telles opérations inversibles mais asymétriques dans leur difficulté. On les appelle souvent problèmes difficiles ou asymétriques.

60 La plus connue de ces opérations asymétriques est la multiplication de nombres premiers, i.e. divisible uniquement par 1 et par eux-mêmes. L'opération inverse, consistant à retrouver ces nombres à partir du résultat, est appelée factorisation 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, nous ne connaissons à ce jour aucune méthode efficace permettant de retrouver ces facteurs premiers. Insistons sur le fait qu'une telle factorisation n'est pas impossible ; elle est mathématiquement parfaitement définie et l'on connaît des algorithmes forts simples permettant de la retrouver. Le problème réside dans la complexité temporelle, i.e. dans le temps de calcul nécessaire à ces

méthodes pour trouver le résultat. De plus, rien ne permet de dire qu'il n'en existe pas. On peut tout juste affirmer qu'aucune méthode efficace de factorisation utilisant une technologie disponible n'a été rendue publique à ce jour.

61 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 « trappe ». L'exemple le plus connu, mais également un des seuls utilisables en pratique, est l'opération sur laquelle repose le fameux cryptosystème RSA. Cette opération est directement liée au problème de la factorisation, la connaissance de facteurs premiers constituant la trappe permettant d'inverser facilement le calcul.

62 En termes d'image, 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.

63 Les recherches en cryptographie de ces 25 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 calcul de « logarithmes discrets » dans des structures mathématiques variées, dont les « courbes elliptiques », sont des exemples particulièrement intéressants.

64 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. 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. En général, 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, nous ne connaissons pas à ce jour de méthode efficace permettant d'effectuer de tels calculs en temps raisonnable dans les groupes utilisés en cryptographie. 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.

65 Nous n'entrerons pas ici dans des détails nécessitant de lourds rappels mathématiques. Notons 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-1024, ceci signifie que l'on utilise RSA avec un paramètre, appelé module, long de 1024 bits et obtenu par multiplication de deux nombres premiers de 512 bits chacun. Il ne faut par conséquent pas s'étonner de voir apparaître en cryptographie asymétrique des paramètres ou des clés de 1024 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.

66 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, …

F.2.1. Chiffrement asymétrique

67 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'idée selon laquelle la même clé doit être utilisée à la fois pour le chiffrement et le déchiffrement n'est pas fondamentalement justifiée.

68 Dans le cadre du chiffrement, l'idée est d'utiliser des paires de clés, ou **bi-clés**, composées d'une **clé privée**[10] 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.

69 Une image classique consiste à imaginer un coffre-fort, comme il en existe couramment, nécessitant une combinaison secrète pour être ouvert mais pouvant être fermé par n'importe qui. L'équivalent de la clé publique est ce coffre, librement accessible et disponible pour tout émetteur. La clé privée est la combinaison permettant d'ouvrir le coffre. Afin de transmettre un document, il « suffit » de récupérer un coffre ouvert appartenant au destinataire, d'y enfermer le document en question et de l'envoyer. Lors de la réception, la connaissance de la clé privée permet d'ouvrir le coffre et d'obtenir le document. Bien entendu, cette image trouve rapidement ses limites, en particulier en termes de disponibilité des coffres. Par contre, dans le contexte numérique, la distribution des clés publiques est bien plus aisée. De plus, une même clé publique peut être utilisée par de nombreux émetteurs alors que les coffres doivent être produits en autant d'exemplaires que nécessaire. Rappelons enfin que, tout comme dans le cas du chiffrement symétrique, aucune intégrité n'est a priori garantie.

70 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.

71 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

[10] 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.

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, i.e. l'ensemble des messages, est restreint à un petit sous-ensemble des messages possibles. Cette problématique est d'ailleurs similaire à celle rencontrée dans le cas symétrique et exposée à partir du paragraphe 27.

72 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 transmettre, chiffré avec le mécanisme à clé publique, que cette clé de session. On peut ainsi transmettre la clé de session de manière sécurisée à son interlocuteur. Ensuite, tout message peut être chiffré avec la clé de session de manière conventionnelle et très efficace.

F.2.2. Signature numérique

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

74 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 formés 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[11], à la manière des MACs vus au chapitre F.1.2. Afin de vérifier la validité d'une signature, il suffit de disposer de la clé publique. Par conséquent, quiconque peut potentiellement s'assurer de l'authenticité d'un message puisque la clé publique peut être librement rendue disponible. Ceci réalise donc très exactement une version électronique de la signature manuscrite classique, certains diront même en mieux que la signature traditionnelle. Notons cependant qu'ici encore la certification de la clé publique est un problème crucial que nous abordons rapidement ci-dessous dans le chapitre F.3.3 consacré à la gestion de clé, et qui est repris plus en détail dans le référentiel intitulé « Gestion des clés cryptographiques – Règles et recommandations concernant la gestion des clés utilisées dans des mécanismes cryptographiques », spécifiquement consacré à cette problématique.

75 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.

76 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, chiffrement et signature sont en effet très proches, la signature d'un message s'apparentant

[11] Plus exactement, le mécanisme consistant à calculer une signature et à l'ajouter au message est appelé « signature avec *appendix* ». 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.

fortement à l'opération de déchiffrement. Ce cas est cependant exceptionnel, la grande majorité des schémas de signature ne pouvant être utilisés à des fins de chiffrement.

F.2.3. Authentification asymétrique d'entités et échange de clés

77 Les idées permettant une authentification interactive d'entités vues au chapitre F.1.3 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.

78 Il existe cependant des mécanismes spécifiquement conçus pour le cadre interactif. Ces primitives dites « à divulgation nulle de connaissance » ou « *zero-knowledge* », bien qu'encore peu utilisées en pratique, sont à la source de la plupart des schémas de signature numérique. On citera par exemple le standard de signature américain DSA (« *digital signature algorithm* »), directement dérivé du protocole d'authentification de Schnorr.

79 Les méthodes asymétriques permettent également de directement résoudre le problème de mise en accord de clé symétrique visant à permettre à deux correspondants ne partageant initialement aucun secret commun de se mettre d'accord sur une clé secrète commune via un moyen de communication potentiellement écouté, voire entièrement contrôlé, par un attaquant. Les travaux initiaux de W. Diffie et M. Hellman [1], déjà mentionnés, ne proposaient d'ailleurs de solution qu'à ce problème.

80 Techniquement, la remarque de base de W. Diffie et M. Hellman est que si l'on utilise une structure mathématique dans laquelle le calcul de logarithme discret est difficile, comme décrit au paragraphe 64, on peut facilement choisir un grand entier secret a et publier x^a sans révéler d'information sur a. 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 $K=y^b=(x^a)^b=x^{(ab)}$. Par contre, un attaquant qui écoute passivement la communication ne peut apprendre que y et z; à ce jour, nous ne connaissons pas de méthode plus efficace que le calcul de logarithmes discrets pour retrouver les secrets a et b afin d'en déduire le secret partagé K.

81 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.

82 Les mécanismes d'authentification et d'échange de clé sont généralement utilisés de concert en pratique car, d'une part, échanger 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 échange de clé doit cependant être réalisé avec soin.

F.2.4. Sécurité des primitives asymétriques

83 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.

84 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 ». Ce ne sont donc pratiquement jamais des preuves au sens de la théorie de l'information, i.e. assurant la sécurité face à un attaquant de puissance infinie (voir paragraphe 37).

85 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 « forge existentielle », consistant à réussir à générer une signature valide d'un message non maîtrisé, à un « cassage total » retrouvant la clé privée de signature.

86 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 F.1.1.3, les modèles de sécurité utilisés en cryptographie moderne peuvent paraître excessifs mais l'expérience montre que c'est loin d'être le cas et qu'il est important de se placer dans ce cadre contraignant afin d'évaluer correctement les primitives.

F.3. Autres primitives cryptographiques

87 Un certain nombre de primitives cryptographiques, ou directement utilisées en cryptographie, ne peuvent pas être classées dans une approche opposant symétrique et asymétrique. C'est en particulier le cas de celles qui n'utilisent pas directement d'éléments secrets comme les clés.

F.3.1. Fonction de hachage

88 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, i.e. 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, ni de pouvoir trouver deux messages distincts ayant même condensat. On dit alors que la fonction de hachage est **sans collision**.

89 Bien entendu, une fonction de hachage étant une fonction d'un ensemble de taille 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.

90 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 bien que déterministe, alors, afin de 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 des sorties des fonctions de hachage cryptographiques.

91 La principale application 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.

F.3.2. Génération d'aléa cryptographique

92 La cryptographie fait un usage intensif de données aléatoires, typiquement afin de générer des clés privées ou secrètes 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 très importante et nécessite de disposer de données aléatoires « de qualité ».

93 À titre d'exemple particulièrement frappant de la nécessité de disposer de bon aléa pour certaines applications, citons celui du 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.

94 Bien que ne remettant pas en cause la sécurité de DSA, on connaît aujourd'hui une attaque qui permet de retrouver la clé privée de signature à condition de disposer de signatures pour lesquelles seulement 3 bits du nombre aléatoire à usage unique sont connus. Cette attaque fonctionne donc très efficacement même si les 157 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 juste partiellement aléatoires.

95 La génération de bits réellement aléatoires, i.e. 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, pire, un microprocesseur de carte à puce. Il existe cependant des dispositifs physiques spécifiques tirant parti de phénomènes supposés aléatoires 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** ».

96 Il est également possible de générer du « **pseudo-aléa** », i.e. 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.

97 Afin d'illustrer la notion de générateur pseudo-aléatoire, décrivons rapidement celui normalisé par l'ANSI sous la référence X9.31 qui reprend celui de la norme X9.17. Il utilise une clé secrète K et une primitive de chiffrement par bloc notée E_K. Ce processus itératif utilise une variable interne, gardée secrète et notée $Etat_i$. Elle est mise à jour lors de chaque itération par le mécanisme décrit ci-dessous Figure 4. La valeur initiale de la variable d'état, ainsi que la clé K forment le germe. Le bloc noté $ALEA_i$ contient les données pseudo-aléatoires générées lors de la $i^{ème}$ itération. Enfin, le bloc Ext_i contient d'éventuelles données extérieures, optionnelles, permettant par exemple de diversifier le mécanisme au moyen de données aléatoires de mauvaise qualité, par exemple issues de l'horloge.

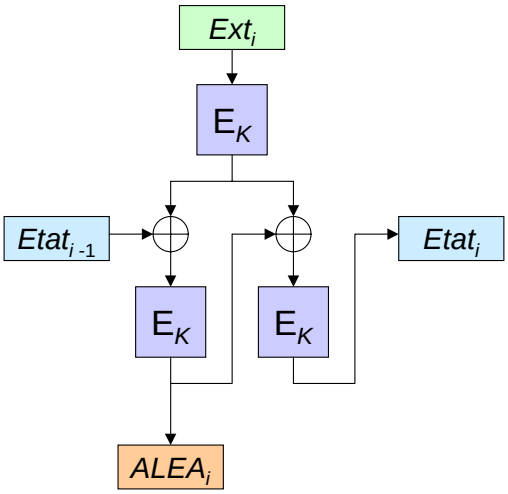


Figure 4. Générateur pseudo-aléatoire X9.31

98 Notons enfin que, par définition, il est en pratique impossible de distinguer du pseudo-aléa correctement généré de véritables bits aléatoires. Malgré l'existence de notions théoriques bien définies issues de la théorie de l'information, comme celle d'entropie, la seule manière de tester des bits 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.

F.3.3. Gestion de clés

F.3.3.1. Clés secrètes symétriques

99 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 organisationnelles.

100 Une durée de vie maximale, appelée **crypto-période**, est de plus 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.

101 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.

102 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 asymétrique permettant de s'en passer.

103 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 ?

104 Une manière simple de résoudre ce problème avec une technique asymétrique est de faire choisir à chaque membre du groupe un bi-clé dont la clé publique est certifiée par une autorité. Chaque membre doit donc uniquement mémoriser son bi-clé et la clé publique de l'autorité. On peut ensuite utiliser une des techniques d'identification évoquées au chapitre F.2.3.

F.3.3.2. Bi-clés asymétriques

105 La gestion des bi-clés en cryptographie asymétrique est à la fois plus simple et plus complexe que dans le cas symétrique. Plus simple, mais également plus sûr, 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.

106 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. De telles constructions sont désignées sous le terme d'**infrastructure de gestion de clé** (**IGC** ou **PKI** pour « *public key infrastructure* »).

107 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.

G. Éléments académiques de dimensionnement cryptographique

Nous présentons dans cette annexe quelques informations issues d'annonces et de publications académiques permettant de contribuer à la justification de certains dimensionnements cryptographiques.

G.1. Records de calculs cryptographiques

Les records de calculs cryptographiques permettent d'estimer 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).

G.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 « tache 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[12],
- 17 juillet 1998, **56 heures** de calcul sur une machine spécifique[13] dont le coût a été estimé à 250 000 $,
- 19 janvier 1999, **22 heures** de calcul en combinant la machine précédemment citée et des calculs sur Internet,

ainsi que 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 F.1.

[12] Voir `http://www.distributed.net`

[13] Voir `http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker`

G.1.2. Records de calcul de factorisation

Les principaux records successifs en termes de calcul de factorisation de modules produits de deux nombres premiers de taille comparable sont :

- juin 1993, **399** bits (120 chiffres décimaux)
- avril 1994, **429** bits (129 chiffres décimaux)
- 10 avril 1996, **432** bits (130 chiffres décimaux)
- 2 février 1999, **466** bits (140 chiffres décimaux)
- 22 août 1999, **512** bits (155 chiffres décimaux)
- Janvier 2002, **524** bits (158 chiffres décimaux)
- 1er avril 2003, **530** bits (160 chiffres décimaux)
- 3 décembre 2003, **576** bits (176 chiffres décimaux)
- 9 juin 2005, **663** bits (200 chiffres décimaux)

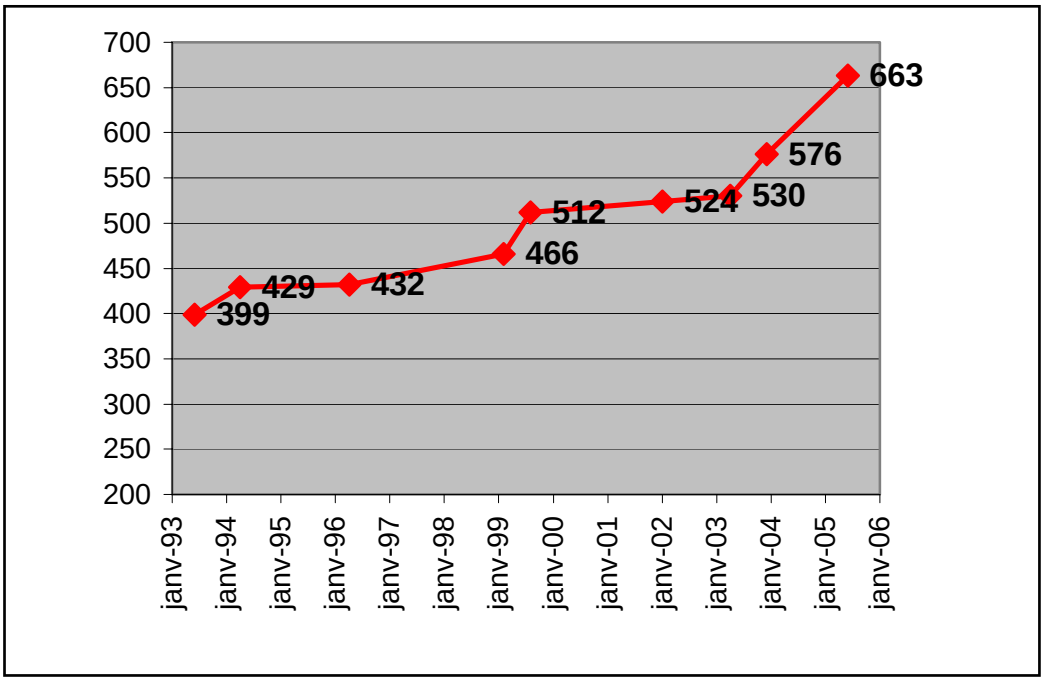


Figure 5. Records de factorisation

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. La plupart des challenges ont été proposés par la société RSA[14].

[14] Voir `http://www.rsasecurity.com/rsalabs/challenges`

Factorisation par des machines dédiées

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

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, aux modules RSA).

L'algorithme SNFS (crible algébrique dit « spécial ») a ainsi permis de factoriser un entier de 244 chiffres décimaux (i.e. 809 bits) début 2003, soit un entier bien plus grand que le dernier record RSA en date.

G.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 :

- 25 septembre 1996, Weber, Denny et Zayer, **281** bits (85 chiffres décimaux)
- 26 mai 1998, Joux et Lercier, **298** bits (90 chiffres décimaux)
- 2000, 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, **432** bits (130 chiffres décimaux)

où la taille en bits désigne la taille du nombre premier *p*.

G.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) sont :

- 1992, Gordon et McCurley, GF($2^{\mathbf{401}}$) soit **401** bits
- 25 septembre 2001, Joux et Lercier, GF($2^{\mathbf{521}}$) soit **521** bits
- 23 février 2002, Thomé, GF($2^{\mathbf{607}}$) soit **607** bits
- 22 septembre 2005, Joux et Lercier, GF(2^{613}) soit **613** bits

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

G.1.5. Records de calcul de logarithme discret dans GF(p^n)

Les principaux records en termes de calcul de logarithme discret dans un corps fini à p^n éléments GF(p^n), pour *p* supérieur à 2, sont les suivants :

- 28 juin 2005, Lercier et Vercauteren, GF($370801^{\mathbf{18}}$) soit **336** bits (101 chiffres décimaux)
- 24 octobre 2005, Joux et Lercier, GF($65537^{\mathbf{25}}$) soit **400** bits (120 chiffres décimaux)
- 9 novembre 2005, Joux et Lercier, GF($370801^{\mathbf{30}}$) soit **556** bits (168 chiffres décimaux)
- août 2006, Joux, Lercier et Vercauteren, GF(p^3) soit **400** bits (120 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 d'une difficulté intermédiaire entre la difficulté dans GF(2^n) et dans GF(p).

G.1.6. Records de calcul de logarithme discret sur courbe elliptique

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

La société Certicom a publié le 6 novembre 1997 une liste de challenges[15] concernant le problème du logarithme discret sur courbe elliptique. Les challenges sont

[15] Voir `http://www.certicom.com`

de trois types : courbes elliptiques « quelconques » définie sur GF(p) ou sur GF(2^n) et courbes de Koblitz également définie sur 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.

Nous avons reproduit ci-dessus les résultats annoncés jusqu'à aujourd'hui ainsi que les prochains challenges non-résolus et le nombre d'opérations qui ont été nécessaires.

G.2. Article de Lenstra et Verheul [2]

Arjen Lenstra et Eric Verheul ont publié en 2001 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.

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. Le fait de le citer et de reproduire quelques-uns des résultats dans cette annexe ne constitue cependant pas une caution de l'article pris dans son intégralité. Nous laissons en particuliers aux auteurs la responsabilité de certaines affirmations.

G.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 disponible pour un prix donné double tous les 18 mois. Le graphique ci-dessous indique l'évolution des tailles de clés nécessaires afin de demeurer équivalent à la sécurité du DES (56 bits) en 1982. Il se lit de la façon suivante. L'année est donnée en abscisse. L'ordonné indique la taille de clé en bit offrant une sécurité équivalente à la sécurité du DES en 1982.

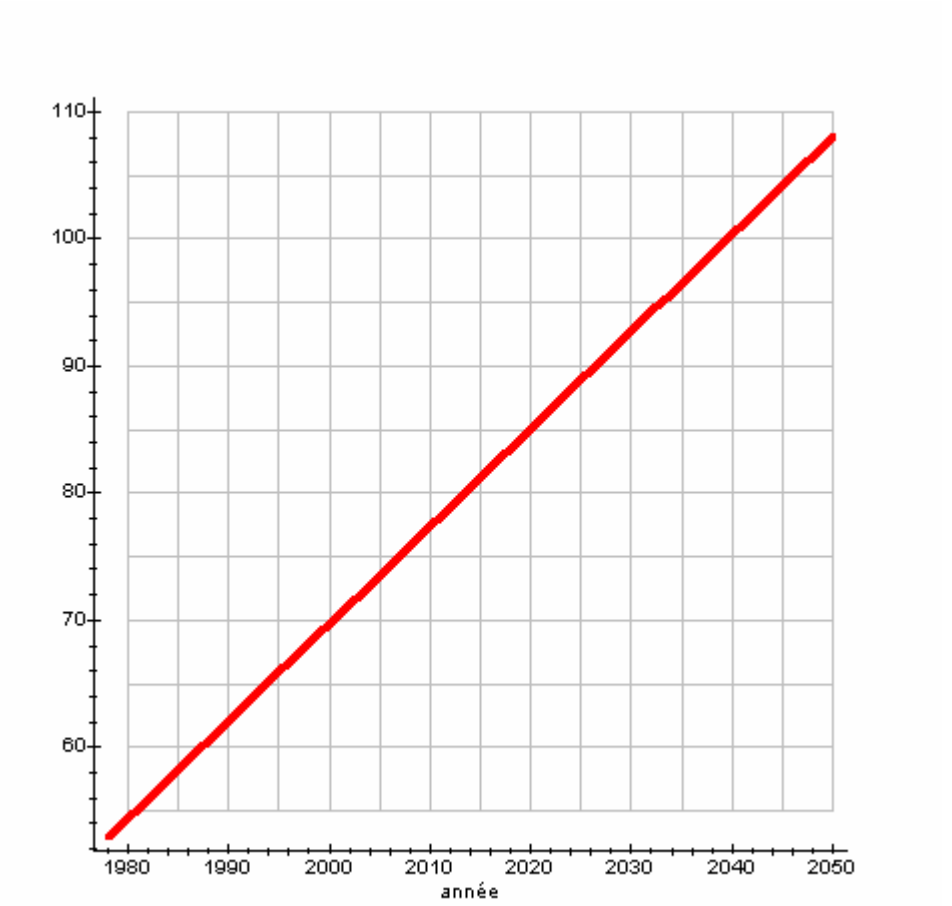


Figure 6. Taille de clé symétrique offrant une sécurité équivalente à celle du DES en 1982

Remarques :

- 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.

G.2.2. Évolution des tailles de modules en cryptographie asymétrique

Le même genre d'évaluation peut être réalisé pour des modules asymétriques. Aucune distinction n'est faite ici entre problème de factorisation et de logarithme discret dans GF(p) car cette distinction aurait peu de sens en pratique. En considère cependant que le problème du logarithme discret est légèrement plus difficile que celui de la factorisation ; en pratique, on peut considérer que dans les gammes de taille qui nous intéressent, 100 à 200 bits les séparent, cette approximation étant très imprécise. Le graphique ci-dessous se lit de la façon suivante. L'année est donnée en abscisse. L'ordonné indique la taille de module en bit offrant une sécurité équivalente à la sécurité du DES en 1982.

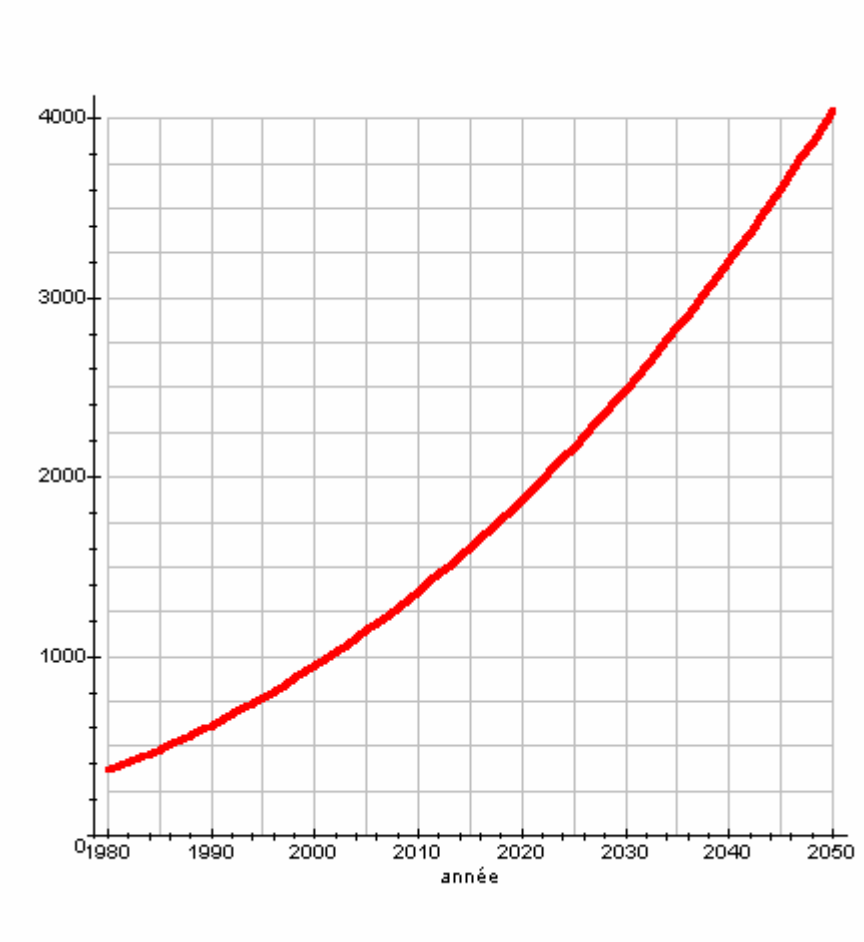


Figure 7. Taille de module offrant une sécurité équivalente à celle du DES en 1982

Remarques :

- 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.

Une comparaison de cette courbe et des records annoncés publiquement s'impose. Il va de soi que l'on ne risque pas d'observer une concordance entre ces deux approches puisque la courbe de la Figure 7 indique des tailles minimales de modules à utiliser afin d'assurer une sécurité cryptographique alors que les records ne traduisent que le savoir-faire académique. Il existe nécessairement une différence, que certains jugerons importante, entre ces deux approches.

À titre de comparaison, rappelons que dans le cas de clés symétriques, il ne viendrait pas à l'idée d'utiliser des clés de 65 bits parce qu'un effort public sur Internet a permis de retrouver une clé de 64 bits. De même, utiliser des modules de 674 bits parce qu'un module de 664 bits a été factorisé, 674 bits étant obtenu en augmentant 664 bits d'un 64^{ième}, semble une approche bien risquée…

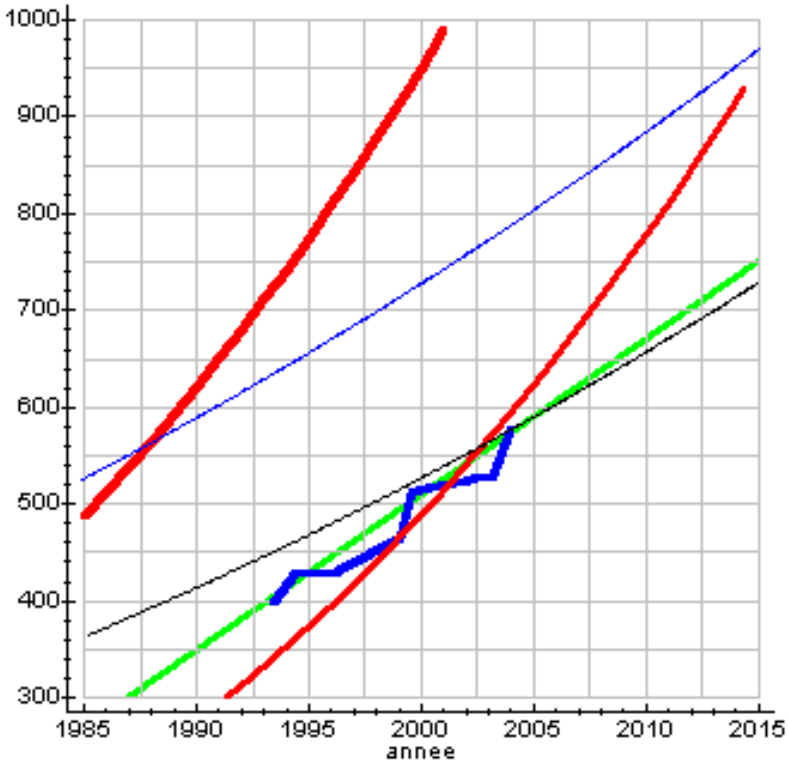


Figure 8. Comparaison entre les records de factorisation et les tailles théoriques nécessaires pour une sécurité équivalente à celle du DES en 1982

Ces précautions étant prises, observons les records publiés et comparons les aux courbes précédentes : dans la figure ci-dessus,

- la courbe bleue épaisse correspond aux records de factorisation publiés,
- la courbe rouge épaisse en haut à gauche correspond à la courbe théorique de la Figure 7,
- la courbe rouge fine correspond à cette même courbe décalée horizontalement de 15 ans,
- la courbe verte est une tentative d'interpolation linéaire des points de la courbe bleue,

- la courbe noire fine correspond à l'extrapolation du dernier record de factorisation en utilisant la complexité théorique de l'algorithme employé (le crible algébrique),
- la courbe bleue fine correspond à la même courbe que la précédente mais en supposant que l'on dispose de 1000 fois plus de puissance de calcul.

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. Ceci dit, la courbe théorique décalée de 15 ans peut être considérée comme une interpolation des records. Dans ce cas, on peut grosso modo dire que suivre les tailles minimales de module préconisées par Lenstra-Verheul assure une protection en pratique face aux attaques publiques d'une quinzaine d'années au-delà de la date de fin d'utilisation prévue.

D'un autre côté, il semble que graphiquement, i.e. en oubliant toute autre considération de nature théorique, les records puissent aussi bien être interpolés par une simple droite…

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 de Lenstra et Verheul mais également de la plupart des experts du domaine. Les observations que nous venons de faire peuvent cependant pousser certains concepteurs à assumer de 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.

G.2.3. Évolution des tailles de courbes elliptiques

Le même travail est réalisé dans le cas de courbes elliptiques. Selon que l'on tient compte d'hypothétiques progrès algorithmiques (courbe rouge) ou pas (courbe bleue), on obtient les deux courbes d'évolution ci-dessous. Le graphique se lit de la façon suivante. L'année est donnée en abscisse. L'ordonné indique la taille de module en bit offrant une sécurité équivalente à la sécurité du DES en 1982.

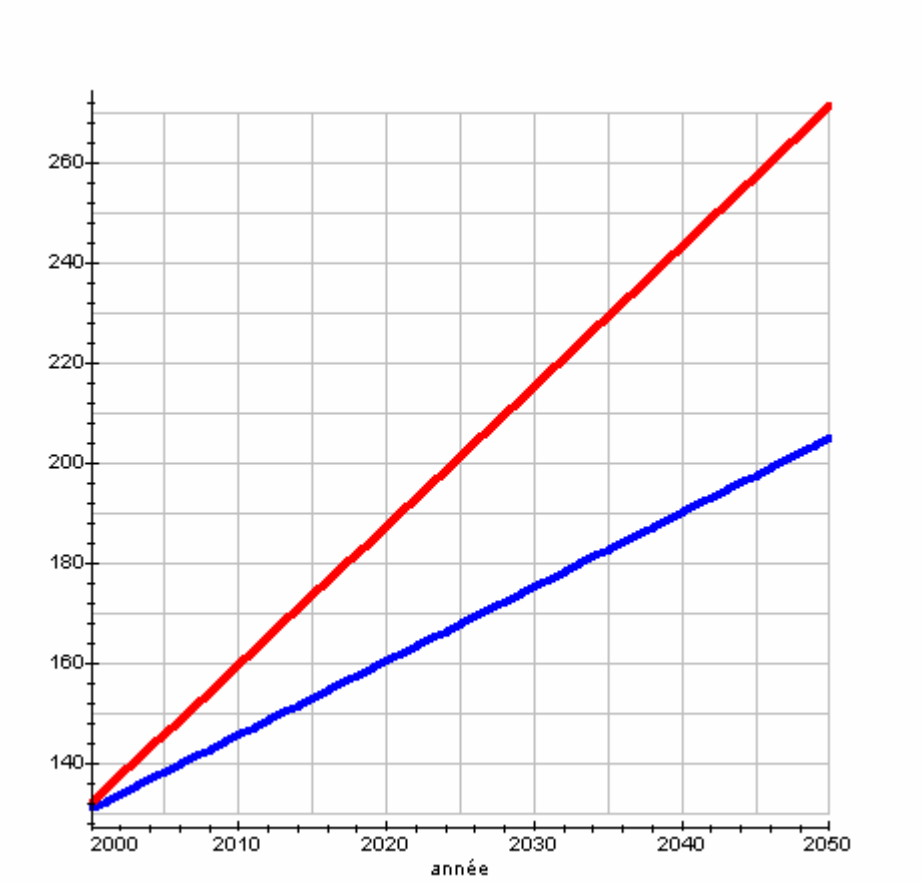


Figure 9. Taille de courbes elliptiques offrant une sécurité équivalente à celle du DES en 1982

G.2.4. Équivalence de sécurité entre taille de module asymétrique et taille 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 la courbe ci-dessous 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'ordonné indique la taille en bits du module asymétrique offrant une sécurité équivalente.

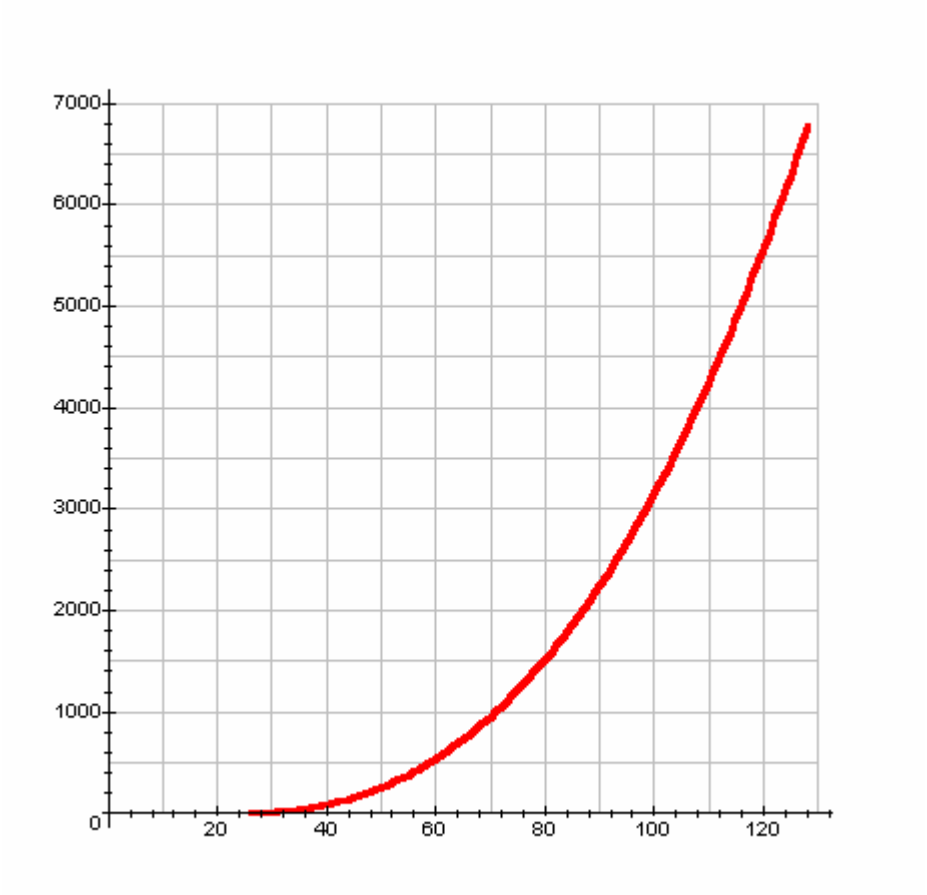


Figure 9. Équivalence entre taille de module asymétrique et taille de clé symétrique

Remarques :

- 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 9. On peut par conséquent énoncer en termes courants que :

- « pour des clés longues d'environ 512 bits, un bit symétrique équivaut à 34 bits asymétriques »,
- « pour des clés longues d'environ 1024 bits, 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* ».

G.2.5. Équivalence de sécurité entre taille de courbe elliptique et taille de clé symétrique

Le même procédé peut être appliqué pour comparer clés symétriques et cryptosystèmes à base de courbes elliptiques. Selon que l'on tient compte d'hypothétiques progrès algorithmiques (courbe rouge) ou pas (courbe bleue), on obtient les deux courbes d'évolution ci-dessous. Le graphique se lit de la façon suivante. Le nombre de bits des clés symétriques est donné en abscisse. L'ordonné indique la taille en bits de la courbe elliptique offrant une sécurité équivalente.

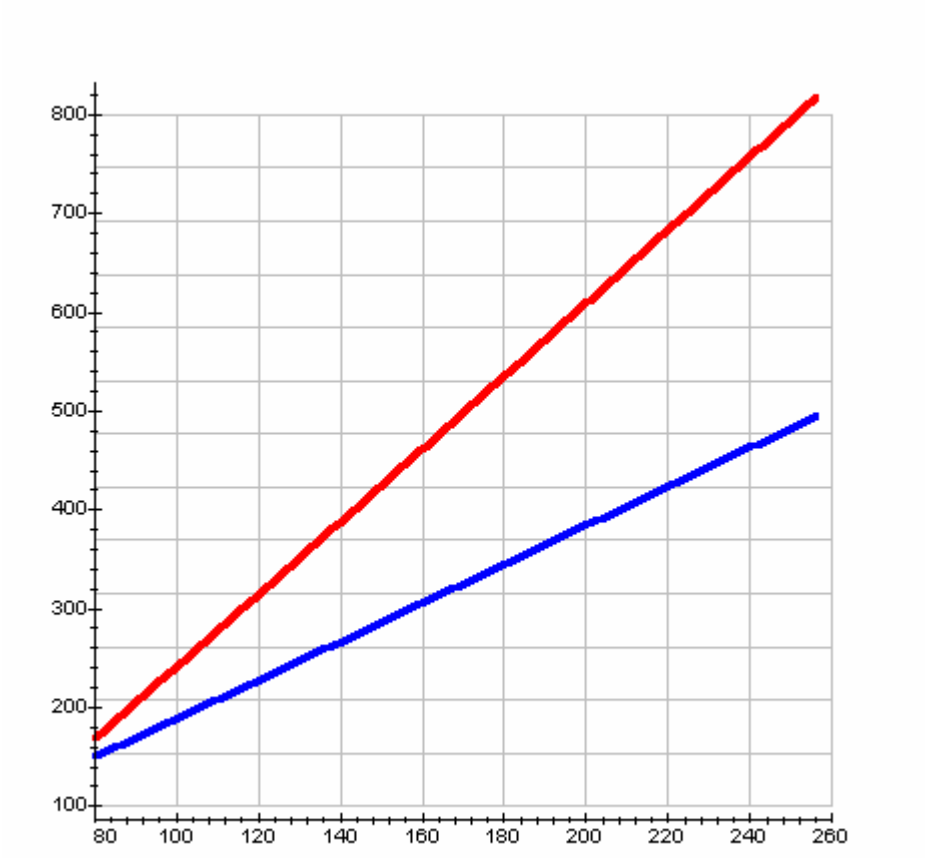


Figure 10. Équivalence entre dimensionnement de cryptosystème à base de courbe elliptique et taille de clé symétrique

Fin du Document