N° d'ordre :

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES
ACULTE DES SCIENCES EXACTES
SIDI BEL ABBÈS

THESE DE DOCTORAT DE 3ème CYCLE

Présentée par

BAGHOR Soufiane

Domaine : Mathématique Informatique
Filière : Informatique
Intitulé de la formation : Réseaux et sécurité de l'information

Intitulé

Sur l'authentification et l'échange de clés sécurisé

Soutenue le : 13/02/2019
Devant le jury composé de :

Président : Dr. TOUMOUH Adil (MCA, UDL-SBA)

Examinateurs : Pr. AMAR BENSABER Djamel (MCA, ESI-SBA)
Dr. BOUCHIBA Djelloul (MCA, CU de Naâma)
Dr. ALI CHERIF Moussa (MCA, UDL-SBA)
Directeur de thèse : Pr. FARAOUN Kamel Mohamed (Pr, UDL-SBA)

Année universitaire : 2018/2019

Remerciements

Mes premiers remerciements vont naturellement au Professeur Faraoun Kamel Mohamed directeur de cette thèse pour m'avoir fait confiance et pour les heures de travail très stimulantes et surtout si enrichissantes. Je me souviendrai longtemps de votre passion inépuisable pour la recherche et merci de m'avoir poussé à me surpasser quand il le fallait.

Je tiens à remercier très chaleureusement tous les membres du jury pour leur lecture minutieuse de la thèse ainsi que leurs commentaires et conseils ont contribué grandement à améliorer la qualité de cette thèse.

Au plan personnel, je dois remercier mes parents, à qui je dédie cette thèse, et qui y sont certainement pour beaucoup dans tout ça.

A tous un grand Merci.

Résumé

Dans le cadre de cette thèse, nous proposons de nouvelles méthodes de protocole d'authentification de l'entité avec un accord de clé de session, en utilisant dans la première approche un mécanisme d'automates cellulaires unidimensionnels de second ordre et dans la seconde approche, un mécanisme d'automates cellulaires bidimensionnels . Les systèmes utilisent des opérations simples pour gérer le problème de l'authentification mutuelle entre deux parties communicantes et permettent le partage d'une clé de session sécurisée commune. La sécurité des schémas proposés est élevée, elle résiste aux attaques les plus fréquentes dans les systèmes d'authentification. En outre, les performances d'exécution du protocole d'authentification sont très compétitives et surpassent celles des schémas similaires existants. Les schémas proposés garantissent les trois exigences d'authentification, à savoir la mutualité, l'authenticité et l'accord clé en utilisant uniquement des opérations élémentaires et parallèles, sans nécessiter des fonctions cryptographiques additionnelles.

Mots clés : Authentification à distance, clés authentifiés à base de mot de passe, authentification mutuelle, accord de clé de session, automates cellulaires.

Abstract

Within the framework of this thesis, we propose novels approaches of remote user mutual authentication protocol with session key agreement, using in the first approach a mechanism of one-dimensional second order cellular automata, and in the second approach a mechanism of two-dimensional cellular automata. The schemes use simple and elementary operations to handle the problem of mutual authentication between two communicating parties, and enable the sharing of a common secure session key. Security of the proposed schemes proven to be high, while it resist most common attacks on authentications schemes. Besides, runtime performances of the authentication protocol are very competitive, and outperform those of existing similar schemes. The proposed schemes ensure the three authentication requirements, namely the mutuality, the authenticity and the key agreement using only elementary and parallelizable operations, without requiring any addition cryptographic functions.

Keywords: Remote user authentication, password authenticated key exchange (PAKE), mutual authentication, session key agreement, cellular automata.

Table des matières

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Introduction générale

Avec la croissance rapide des applications distribuées, des services et des ressources partagées sur Internet et dans les environnements distribués informatiques, le besoin de protocoles d'authentification de l'utilisateur devient de plus en plus crucial pour assurer un contrôle d'accès sécurisé et efficace. L'authentification des utilisateurs distants est essentielle pour empêcher les adversaires non autorisés d'accéder aux ressources et de services des systèmes et de valider l'identité des utilisateurs légitimes dans un réseau distribué, un réseau de capteurs sans fil ou un système de base de données client-serveur. Lors d'une tentative d'authentification, les serveurs authentifient d'abord les utilisateurs distants, et seulement après que l'authentification réussit, le serveur accorde l'accès aux ressources / services l'utilisateur autorisés ; Alors que les entités non autorisées ou malveillantes visant à gâcher la sécurité du réseau sont négligées.

L'authentification de l'utilisateur à distance et les accords clés sécurisés sont deux blocs de construction importants de tout système distribué sécurisé multi-utilisateurs. Bien que le premier vise à prouver l'identité d'un utilisateur donné et à lui permettre d'accéder à un service donné, le but ultérieur est d'établir une clé cryptographique qui permet une communication sécurisée entre l'utilisateur et le fournisseur du service pendant une session de communication prédéterminée. L'utilisateur possède des informations personnelles (qui peuvent être soit un mot de passe ou une carte à puce) qui permet son authentification auprès d'un serveur et permet d'établir une clé cryptographique valide de manière sécurisée.

Un attaquant passif ou un homme au milieu ne devrait pas être en mesure d'obtenir suffisamment d'informations permettant de casser ou de deviner le mot de passe, et aucun utilisateur non autorisé ne doit pouvoir se faire passer pour le serveur et prendre l'identité d'un utilisateur légitime.

La combinaison de l'authentification des utilisateurs et de l'accord clé sécurisé conduit à la définition d'un protocole d'accord clé authentifié par mot de passe permettant de réaliser une communication sécurisée entre deux parties ou plus en fonction uniquement de la connaissance du mot de passe unique.

Dans les scénarios spécifiques d'une application, un besoin supplémentaire existe pour effectuer une authentification dans les deux sens entre un utilisateur et un serveur. Les protocoles permettant cette possibilité sont nommés protocoles d'authentification mutuelle ou protocoles d'authentification bidirectionnelle. Il s'agit d'un utilisateur s'authentifiant sur un serveur et ce serveur s'authentifiant à l'utilisateur de telle sorte que chaque partie soit assurée de l'identité de l'autre. L'authentification mutuelle est généralement utilisée lorsqu'un niveau supplémentaire de sécurité est nécessaire, en particulier dans les transactions financières entre organisations.

Dans cette thèse, nous présentons de nouveaux systèmes d'authentification et d'accord de clé en se basant sur le modèle de calcul d'automate cellulaire (CA). Contrairement aux modèles existants basés sur CA, le premier schéma proposé n'utilise pas de routines cryptographiques supplémentaires (fonctions de hachage / cryptage) et garantit plusieurs exigences de sécurité en utilisant uniquement des mécanismes et des propriétés d'automates cellulaires. Ainsi, le système offre un haut niveau de sécurité et de performances avec un coût de calcul optimisé. Ce schéma gère l'authentification des utilisateurs distants, l'accord de clé de session et l'authentification mutuelle entre l'utilisateur et le serveur à l'aide d'un modèle de calcul simple et élégant, et il est résistant aux attaques communes.

Le premier schéma ne fournit pas la confidentialité persistante, puisque la clé de session est dérivée de la clé secrète partagée à long terme. En outre, le schéma n’est effectué seulement avec une clé partagée à long terme et nécessite un enregistrement préalable chez le fournisseur du service afin que l’utilisateur reçoive une carte à puce contenant la clé secrète protégée par le mot de passe d’utilisateur ; l’avantage est que cette approche fournit une authentification forte à deux niveaux ; mais dégrade la fluidité de la mise en œuvre du protocole. Pour ces raisons nous avons conçu une deuxième approche où l’authentification de l’entité et l’établissement de clé ne s’effectuent que par la connaissance d’un petit mot de passe partagé entre deux participants pour qu’ils puissent établir une session de communication en toute sécurité tout en fournissant la confidentialité persistante.

Le reste du document est organisé comme suit :

Partie I : Nous introduisons dans le premier chapitre de la première partie une description exhaustive du problème d'authentification de l'entité et l'établissement de clé cryptographique. Nous abordons ensuite dans le chapitre qui suit l'outil mathématique (CA) utilisé afin de répondre aux exigences d'implémentation en assurant un niveau de sécurité élevé.

Partie II : Nous décrivons dans la deuxième partie d'une façon détaillée la première approche proposée après avoir présenté un état de l'art sur l'authentification d'utilisateur distant en utilisant les automates cellulaires unidimensionnels de second ordre.

Partie III : De la même façon que la deuxième partie, nous décrirons la deuxième approche proposée après avoir présenté un état de l'art sur l'échange de clé authentifié en utilisant les automates cellulaires bidimensionnels.

Nous conclurons cette thèse, par une vision des apports des travaux réalisés ainsi qu'un ensemble de perspectives futures.

Partie I :
Problématique d'authentification et outil mathématique

Chapitre I :
La cryptographie et le problème d'authentification

Partie I : Problématique d'authentification et outil mathématique.

Chapitre I : La cryptographie et le problème d'authentification

Sommaire

1. Introduction

L'authentification et l'établissement de clés sont des éléments fondamentaux pour la sécurisation des communications électroniques. Les algorithmes cryptographiques pour le chiffrement et l'intégrité ne peuvent pas remplir leur fonction, sauf si des clés sécurisées seront établies et les utilisateurs savent telles parties partagent telles clés. Il est essentiel que les protocoles d'authentification et d'établissement de clés soient adaptés à leur usage.

2. La cryptographie

2.1. Définition et utilité

L'art et la science de garder un secret sont appelés cryptographie. De nos jours, ce sont les mathématiciens et les physiciens qui étudient la cryptologie. La cryptographie historique est plus un art qu'une science. Ces dernières décennies, la cryptographie s'est développée en une science.

Cette science est exploitée par les informaticiens pour des applications. Les schémas sont conçus et analysés de façon plus systématique, avec comme objectif ultime de donner des preuves rigoureuses qu'une construction est sûre.

Une bonne approche du problème est de comprendre quelle notion de sécurité est demandée pour une telle application, et de trouver un schéma cryptographique qui satisfasse la notion. Une conséquence de cette approche est la modularité : un concepteur peut éliminer un schéma de chiffrement et le remplacer par un autre (qui satisfasse aussi la définition nécessaire de sécurité) sans avoir à s'inquiéter de comment la sécurité est affectée sur l'application globale.

2.2. Les objectifs de la cryptographie

Bien que l'on reporte des définitions mathématiques aux sections suivantes, on décrit ici des définitions pour les non-mathématiciens. Van Oorschot et Vanstone [I.31] identifient quatre objectifs fondamentaux qui peuvent être atteints par des algorithmes cryptographiques.

2.2.1. La confidentialité des données

La confidentialité est le problème le plus ancien auquel la cryptographie ait tenté de trouver une réponse. Le problème à résoudre est le suivant : deux personnes veulent communiquer et utilisent pour cela un canal public qui peut être espionné par un adversaire. La solution consiste à transformer les données de sorte à les rendre illisibles à l'adversaire tout en assurant que le destinataire pourra appliquer la transformation inverse pour retrouver les données originelles (seul le destinataire peut connaître le contenu des messages qui lui sont transmis).

2.2.2. L'intégrité des données

Le problème de l'intégrité est également un problème ancien qui s'est accentué avec l'émergence de l'écrit sous forme électronique. En effet, la copie, la

modification puis la diffusion du document sont des opérations facilement réalisables.

De même, l'intégrité de certaines données de sécurité est sensible. C'est par exemple, le cas du mot de passe qui doit être stocké sur la carte à puce ou dans l'ordinateur. Si ce mot de passe venait à être modifié, la personne ne pourrait plus s'authentifier sur le système. En outre, lorsque l'on télécharge sur Internet un document, on veut être certain que le document reçu correspond bien à celui que le site web présentait (le destinataire doit pouvoir s'assurer que le message n'a pas été modifié).

De manière plus générale, le problème de l'intégrité peut apparaître dans le cas du partage de ressources et dans le cas de l'envoi de messages sur un canal non sécurisé.

2.2.3. L'authentification d'un point de vue général

La cryptographie apporte des techniques permettant d'authentifier l'émetteur d'un document ou d'un écrit électronique (le destinataire d'un message doit pouvoir s'assurer de son origine).

En cryptographie, l'authentification regroupe l'authentification de données qui permet de garantir que les données transmises proviennent bien d'un émetteur désigné, et l'identification de personne qui consiste à prouver son identité.

Il existe d'autres systèmes d'identification permettant par exemple l'entrée sur un système informatique ou l'accès à certaines informations sur un site web. Dans ce cas, l'utilisateur entre son nom et son mot de passe. En tapant son login, l'utilisateur déclare une identité. Le système acceptera l'utilisateur si ce dernier peut fournir une preuve de cette identité en révélant par exemple un mot de passe correct. Le mot de passe sera alors comparé à celui conservé par le système soit de manière centralisée, soit de manière locale. Lors d'une authentification centralisée, un adversaire espionnant le réseau peut intercepter le login et le mot de passe et se faire passer pour cette personne. Ultérieurement, afin de protéger l'identification d'un utilisateur, la cryptographie a apporté une réponse originale en proposant le concept de preuve d'identité par preuve à divulgation nulle de connaissance "zero knowledge". Il s'agit de mécanismes qui permettent à une personne de prouver son identité sans avoir à divulguer son secret. On peut prouver que le système qui authentifie ne peut apprendre aucune information sur le secret qui

représente la personne lors d'une instance de ce protocole. Ceci est fondamental dans de nombreux protocoles cryptographiques.

2.2.4. La non répudiation

L'émetteur et le destinataire ne peuvent nier avoir respectivement envoyé et reçu le message. Ce service sera détaillé dans la section (3.2.6).

Ces exigences doivent être assurées si l'on souhaite effectuer une communication sécurisée. Il n'existe pas une méthode simple et sûre pour permettre de telles exigences, mais une multitude de techniques permettent, en les combinant, de satisfaire ces besoins de sécurité. Cependant, une sécurité absolue basée sur les théories mathématiques est impossible. En effet, tout secret peut être découvert. Il s'agit donc de déterminer quelles seraient les conséquences et les dégâts engendrés si le secret était amené à être découvert. A partir de cette analyse, il faut définir les degrés de sécurité et la complexité des algorithmes pour protéger ce secret. Plus la complexité de l'algorithme utilisé est grande, plus le temps pour casser cet algorithme sera long.

De manière générale, "Si le coût nécessaire pour casser un algorithme dépasse la valeur de l'information chiffrée, alors cet algorithme peut être considéré comme sûr".

2.3. Les objets cryptographiques

Nous considérons maintenant les définitions des mécanismes cryptographiques qui sont généralement utilisés pour fournir les principaux services cryptographiques. Le tableau I.1 résume la notation que nous utiliserons pour ces mécanismes tout au long de ce chapitre.

Tableau I. 1 - Résumé de la notation pour les algorithmes cryptographiques

$E_A(M)$	Cryptage par la clé publique de l'entité A du message M.
$\{M\}_K$	Cryptage symétrique du message M avec la clé partagée K.
$MAC_K(M)$	Code d'authentification du message M en utilisant la clé partagée K.
$Sig_A(M)$	Signature numérique du message M généré par l'entité A.

Définition I.1 : Un système cryptographique est composé de trois ensembles finis : un ensemble de clés ***K***, un ensemble de messages ***M*** et un ensemble de texte chiffré ***C*** avec trois algorithmes.

2.3.1. Les algorithmes de chiffrement

Sont les principes, les moyens et les méthodes de transformation de données en vue de dissimuler leur contenu pour empêcher leur modification ou leur utilisation illégale.

Le chiffrement est le processus de transformation d'une donnée de telle manière à la rendre incompréhensible. Le résultat du chiffrement est appelé texte chiffré ou cryptogramme.

Définition I.2 : Un algorithme de cryptage, qui prend en entrée un élément $m \in M$ et une clé de cryptage $k \in K$ et émet en sortie un élément $c \in C$ *défini comme* $c = E_k\{m\}$. *L'algorithme de chiffrement peut être randomisé de* sorte qu'un c différent donne le même m.

2.3.2. Les algorithmes de déchiffrement

Tandis que le déchiffrement est le processus inverse du chiffrement. Il correspond au procédé de reconstruction de la donnée origine à partir de son cryptogramme.

Définition I.3 : Une fonction de décryptage, qui prend en entrée un élément $c \in C$ et une clé de décryptage $k^{-1} \in K$ et émet en sortie un élément $m \in M$ défini comme $m = D_{k^{-1}}\{c\}$. Nous exigeons que $D_{k^{-1}}\{E_K\{m\}\} = m$.

Figure I. 1 - Procédé de la cryptographie

2.3.3. Les algorithmes de génération de la clé

Les deux opérations (chiffrement et déchiffrement) s'effectuent par le biais d'un paramètre secret dans l'algorithme de chiffrement, qui lui, est public (principe de Kirchhoff). Il s'agit d'une suite binaire aléatoire de taille prédéfinie appelée la clé cryptographique. La sécurité sera donc basée sur la protection et le non divulgation de la clé secrète.

Définition I.4 : Un algorithme de génération de clé, qui génère une clé de cryptage valide $k \in K$ et une clé de déchiffrement valide $K^{-1} \in K$.

Suivant la gestion des clés, le chiffrement moderne est divisé en deux classes, le chiffrement symétrique et le chiffrement asymétrique. Comme le montre le tableau I.1, nous utilisons une notation différente pour distinguer entre le cryptage symétrique et asymétrique.

2.3.4. Le chiffrement symétrique

Les algorithmes dits "symétrique" sont des algorithmes utilisant une seule clef, cette clef est appelée clef secrète. En général, cette clef secrète va jouer le rôle de clef de chiffrement et de déchiffrement. Une fois qu'un message sera chiffré à l'aide de la clef secrète, il pourra même être envoyé sur un réseau non-sécurisé.

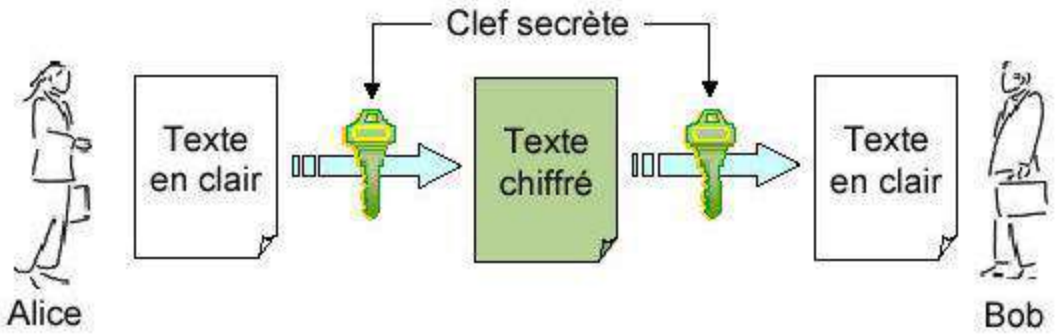


Figure I. 2 - Procédé du chiffrement symétrique

Définition I.5 : Un schéma de cryptage est dit symétrique si $K = K^{-1}$.

La faiblesse de cet algorithme est qu'il repose uniquement sur le non divulgation de cette clef secrète. Si la clef secrète est dévoilée, alors n'importe qui pourra consulter les messages chiffrés avec celle-ci. De plus, le transport de la clef secrète entre Alice et Bob reste problématique.

Il existe deux types d'algorithmes à clef secrète. Certains traitent le message en clair un bit à la fois, ceux-ci sont appelés stream-cipher pour algorithmes de chiffrement continu. D'autres opèrent sur le message en clair par groupe de bits.

Ces groupes sont appelé blocs, ces algorithmes sont appelés block-cipher ou algorithme de chiffrement par bloc.

2.3.5. Le chiffrement asymétrique

Les algorithmes dits "asymétrique" sont des algorithmes utilisant une paire clef : une clef publique et une clef privée. Dans ce cas, la clef de chiffrement est différente de la clef de déchiffrement. De plus, on ne peut retrouver une clef à partir de l'autre clef ce qui garantit l'unicité de la paire de clef.

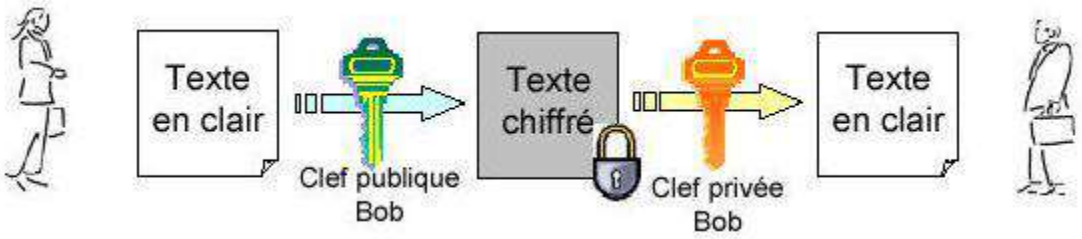


Figure I. 3 - Procédé du chiffrement asymétrique

Définition I.6 : Dans un algorithme de cryptage asymétrique ou à base de clé publique $K \neq K^{-1}$ et il est difficile de calculer la clé privée K^{-1} à partir de la clé publique K.

Cet algorithme semble beaucoup plus adapté que les algorithmes à clef secrète pour sécuriser les communications. Cependant, cet algorithme ne peut pas se substituer aux algorithmes à clef secrète essentiellement pour une raison : Les algorithmes à clef publique sont généralement beaucoup plus lents à traiter que les algorithmes à clef secrète.

2.3.6. Les fonctions de hachage

Une fonction de hachage dérive une donnée de taille fixe appelée empreinte depuis un document de taille variable, de façon à ce que cette donnée représente le contenu d'origine du document dans une forme compressée et sans possibilité de le reconstruire.

Définition I.7 : Une fonction $f : X \rightarrow Y$ est unidirectionnelle si on peut facilement calculer $y = f(x)$ étant donné $x \in X$, mais difficilement calculable pour trouver tout x avec $f(x) = y$ pour presque toutes les valeurs de $y \in Y$.

Les fonctions de hachage sont nombreuses, nous citons : MD4, MD5, SHA-1… Ces fonctions doivent impérativement vérifier certaines propriétés :

• **Fonction à sens unique :** étant donné ***y***, il est difficile de trouver un ***x*** tel que ***H(x) = y*** en un temps raisonnable (Il doit être impossible de pouvoir reconstruire le document d'origine à partir d'un résultat de hachage). Cette propriété permet d'empêcher une personne malintentionnée de modifier le contenu du disque dur ou d'un fichier envoyé.

• **Fonction résistante aux collisions :** il est difficile de trouver ***x*** et ***x' (x ≠ x')*** tels que ***H (x) = H (x')*** en temps raisonnable. Cette propriété permet d'empêcher l'émetteur de créer des fichiers ayant la même "empreinte". Un émetteur pourrait renier le premier document en disant qu'il a envoyé le second. Cette notion est aussi importante dans le modèle hash-and-sign qui consiste à signer uniquement une empreinte du fichier. Ceci permet en outre de construire des schémas de signature qui prennent leur entrée dans un espace fixe et permettent de signer des documents de taille variable.

• **Fonction résistante à un deuxième antécédent :** étant donné ***y = H(x)***, il est difficile de trouver en temps raisonnable ***x' ≠ x*** tel que ***H (x') = y***. Dans le modèle de signature hash-and-sign (deux documents différents ne doivent jamais résulter sur une même valeur de hachage), ceci empêche un attaquant de trouver un deuxième message ayant même signature qu'un message signé donné.

Une fonction de hachage doit prendre en considération le volume des données (les données volumineuses peuvent être aussi traitées) et la rapidité de calcul.

3. La problématique de l'authentification

3.1. La définition du problème

Ce sont les scientifiques de la communauté de la cryptographie et la communauté de la sécurité informatique qui ont produit un grand nombre de protocoles. Cela a conduit à de nombreuses difficultés pour les chercheurs et les praticiens qui souhaitent naviguer dans cette vaste littérature. La procédure que nous allons utiliser pour expliquer les idées est de concevoir un protocole d'établissement de clés à partir des premiers principes.

Les problèmes avec le protocole seront révélés dans les étapes à travers la présentation des attaques légitimes afin que chaque problème puisse être résolu à son tour. Éventuellement un bon protocole est atteint.

Avant de commencer la conception de tout protocole l'architecture de communication doit être établie. Notre scénario à un ensemble d'utilisateurs et un serveur, chaque utilisateur peut établir une nouvelle clé avec le serveur afin de sécuriser leurs communications ultérieures grâce à la cryptographie. Une telle clé est connue comme une clé de session. Il est important de comprendre que la réussite de l'établissement de clés et l'authentification de l'entité, n'est seulement le début d'une session de communication sécurisée : (la protection des données réelles).

Tous les utilisateurs font confiance au serveur. En outre, le serveur est digne de confiance pour générer la nouvelle clé et de le faire d'une manière telle qu'elle est suffisamment aléatoire pour empêcher un attaquant d'obtenir toute information utile à ce sujet.

Nos scénarios impliquent donc deux entités. Un utilisateur que l'on note **U** et le serveur **S**. L'objectif du protocole est pour **U** et **S** d'établir une nouvelle clé secrète $\mathbf{K_S}$ qu'ils peuvent l'utiliser pour sécuriser leurs communications ultérieures. Le rôle de **S** est de générer $\mathbf{K_S}$ et la transporter à **U**. Le but du protocole peut se résumer comme suit.

• A la fin du protocole la valeur de $\mathbf{K_S}$ devrait être connue par **A**, mais pas par les autres parties, à l'exception de **S**.

• **U** et **S** doivent savoir que $\mathbf{K_S}$ est nouvellement générée.

3.1.1. La méthode naïve

Commençons avec une perspective complètement naïve. Un protocole pour réaliser le transport d'une nouvelle clé de session $\mathbf{K_S}$ est représenté sur la figure I.4.

Le protocole sur la cette figure pourrait être spécifié comme suit :

1. U → S : U

2. S → U : K_s

Protocole I.1 - Première tentative du protocole en notation conventionnelle

Le protocole est constitué de deux messages. Tout d'abord, l'utilisateur **U** contacte l'utilisateur **S** en lui envoyant son identité ; d'autre part, **S** retourne la clé $\mathbf{K_s}$ à **U**. Examinons le (manque de) la sécurité de ce protocole.

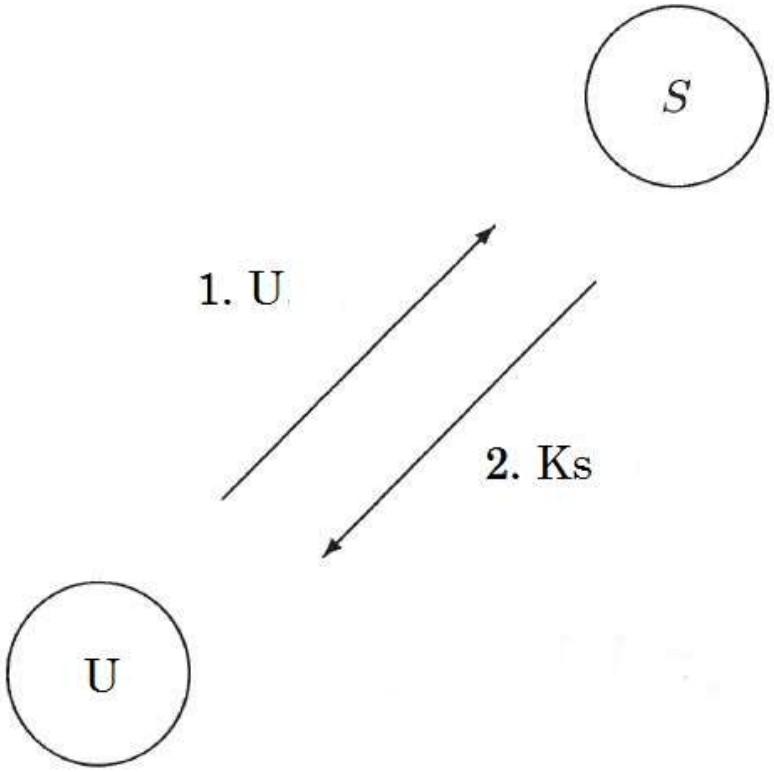


Figure I. 4 - Première tentative du protocole

3.1.2. Le problème de confidentialité de la clé de session

Le problème est que la clé de session $\mathbf{K_S}$ doit être transportée à **U** mais pas à d'autres entités. C'est une hypothèse que l'adversaire peut écouter tous les messages envoyés ou reçus. Ceci est une hypothèse réaliste dans les systèmes de communication typiques tels que l'Internet et les réseaux d'entreprise. En effet, si cette possibilité peut être écartée alors il n'y a probablement pas une nécessité d'appliquer la sécurité.

Hypothèse de sécurité 1 : L'adversaire est capable d'écouter sur tous les messages envoyés dans un protocole cryptographique.

Afin de garantir la confidentialité, il est nécessaire d'utiliser un algorithme cryptographique et une clé associée à chaque utilisateur. Pour l'instant, nous allons tout simplement faire l'hypothèse que le serveur **S** partage initialement une clé secrète $\mathbf{K_{US}}$ avec chaque utilisateur du système.

La nécessité de garder $\mathbf{K_S}$ confidentielle conduit immédiatement à notre deuxième tentative de protocole représenté sur la figure I.5.

Ce protocole commence comme notre première tentative, **U** envoie à **S** son identité. **S** génère la clé de session K_S, puis la chiffre par la clé K_{US} puis envoie le résultat à **U**. Ce protocole est tout aussi non sécurisé dans un environnement ouvert comme notre première tentative, mais pour une toute autre raison.

Un espion passif ne peut pas voir K_S puisque les messages cryptés ne peuvent être lus que par les destinataires légitimes qui ont les clés nécessaires pour le déchiffrement.

Cela nous amène à la question des algorithmes cryptographiques. Une compréhension des algorithmes cryptographiques est essentielle pour la conception correcte des protocoles. Cependant, les détails de l'algorithme cryptographique utilisé sont souvent hors de propos et ces détails sont souvent évités pendant la conception et l'analyse de protocole. Toutes les attaques décrites dans cette section sont indépendants des algorithmes cryptographiques utilisés.

Dans de nombreuses analyses, l'hypothèse d'une «cryptographie parfaite» est faite, ce qui signifie qu'il n'y a rien à gagner par l'adversaire dans l'observation d'un message crypté. Naturellement cette hypothèse pratique apporte avec elle certaines responsabilités pour le concepteur du protocole.

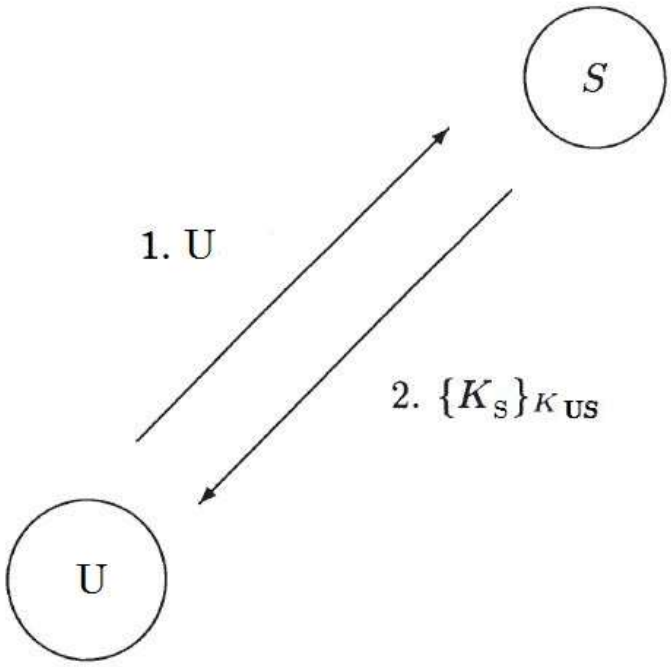


Figure I. 5 - Deuxième tentative du protocole

Afin de faire une conception pratique il doit y avoir des algorithmes cryptographiques appropriés disponibles qui satisfont aux exigences de sécurité.

En outre, ces exigences doivent être explicites dans le cadre de la spécification du protocole. Une autre approche consiste à inclure un modèle abstrait des algorithmes cryptographiques dans le cadre de la spécification du protocole. Une comparaison de ces deux points de vue différents a été réalisée par Abadi et Rogaway [I.2].

3.1.3. Le problème d'authentification de l'entité

Proposons un autre scénario où deux utilisateurs **A** et **B** veulent établir une nouvelle clé de session $\mathbf{K_{AB}}$ en passant maintenant par un serveur de confiance **S**.

Le problème avec le protocole sur la Figure I.6 est la difficulté d'obtention des informations sur qui d'autre à la clé de session. Nous devons prendre en compte que l'adversaire est non seulement capable d'écouter les messages envoyés, mais peut également capturer des messages et de les modifier.

Nous pouvons résumer la situation en disant que l'adversaire a le contrôle complet du canal sur lequel les messages de protocole découlent.

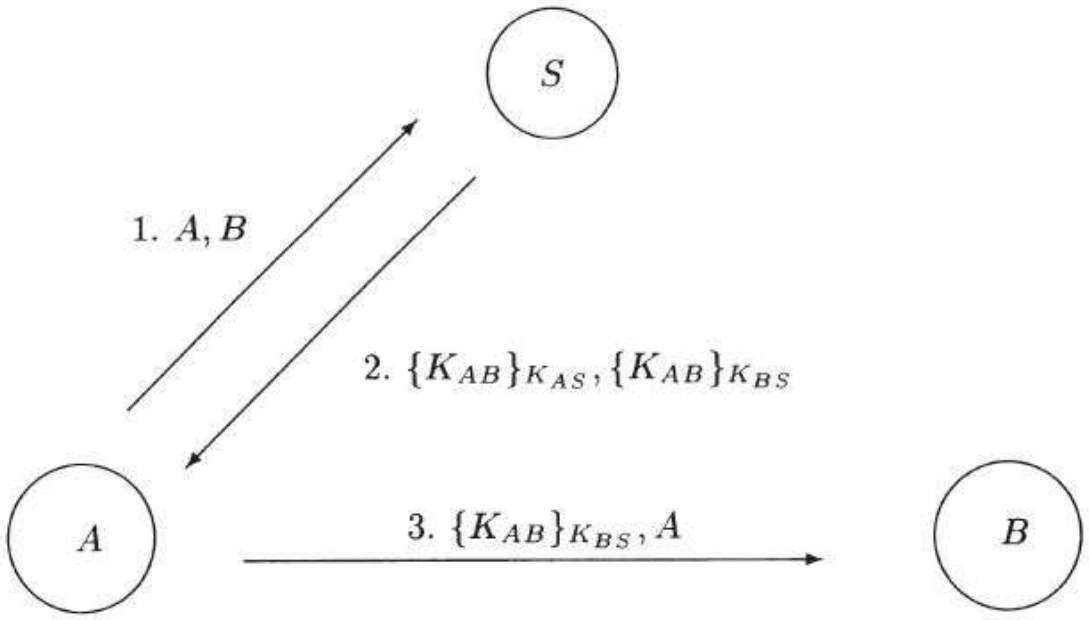


Figure I. 6 - Extension de la deuxième tentative du protocole

Hypothèse de sécurité 2 : L'adversaire est en mesure de modifier tous les messages envoyés dans un protocole cryptographique en utilisant toutes les informations disponibles. En outre, l'adversaire peut réacheminer tout message à tout autre participant. Cela inclut la possibilité de générer et d'insérer complètement de nouveaux messages.

Une attaque sur notre deuxième tentative du protocole est représentée sur la Figure I.7. L'attaque est très simple. L'adversaire **C** intercepte simplement le message de **A** à **B** et substituts d'identité de **A** par **D** (où **D** pourrait être toute identité propre, y compris **C**). La conséquence est que **B** croit qu'il partage la clé avec **D** alors qu'il est en fait le partage avec **A**.

Les résultats ultérieurs de cette attaque dépendront du scénario dans lequel le protocole est utilisé, mais peuvent inclure des actions telles que **B** partage des informations avec **A** qui aurait dû être les partagé seulement avec **D**. Bien que **C** n'obtienne pas $\mathbf{K_{AB}}$ nous considérons toujours le protocole comme rompu car il ne satisfait pas notre exigence que les utilisateurs devraient savoir qui d'autre connaît la clé de session.

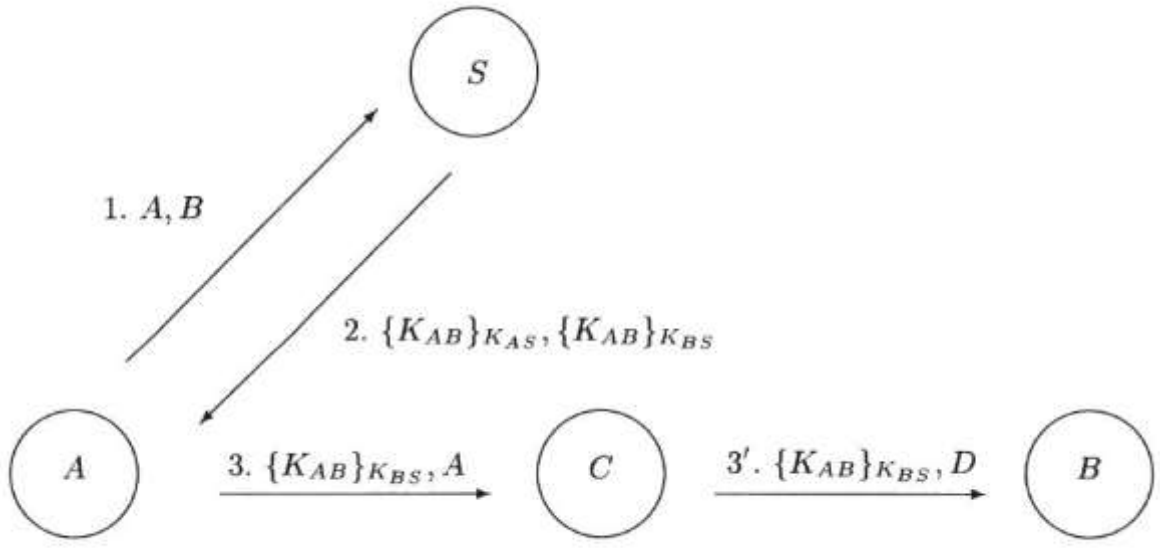


Figure I. 7 - Attaque sur la deuxième tentative du protocole

Cependant, une autre attaque possible sur le protocole permettra à **C** d'obtenir la clé de session comme le montre la Figure. I.8.

Dans cette seconde attaque **C** modifie le message de **A** vers **S** de sorte que **S** crypte la clé K_{AC} avec la clé de **C** au lieu d'avec celle de **B**. Puisque **A** ne peut pas distinguer entre les messages cryptés destinés à d'autres participants, il ne détectera jamais la modification. Notez que K_{AC} est tout simplement un nom formel d'une suite de bits qui représente la clé de session qui sera donc acceptée par **A**. De même **C** collecte le message de **A** destiné à **B** de telle sorte que **B** ne sera pas en mesure de détecter toute anomalie.

Le résultat de cette attaque est que **A** croit que le protocole a été complété avec succès avec **B** alors qu'en fait **C** connait K_{AC} et peut donc passer pour **B** et connaître toutes les informations envoyés de **A** vers **B**.

Notez que, contrairement à l'attaque précédente, celle-ci ne réussira que si **C** est un utilisateur légitime connu par **S**. Cela, encore une fois, est une hypothèse tout à fait réaliste - il est largement admis que les participants internes sont souvent plus une menace que les étrangers. Cela nous amène à une autre hypothèse de sécurité.

Hypothèse de sécurité 3 : L'adversaire peut être un participant légitime de protocole (un interne), ou une partie externe (un étranger), ou une combinaison des deux.

Pour surmonter l'attaque, les noms des utilisateurs qui doivent partager K_{AB} doivent être liés cryptographiquement à la valeur de K_{AB}. Ceci conduit au protocole représenté sur la Figure. I.9 ; où les noms de **A** et **B** sont inclus dans les messages cryptés reçus de **S**. Il peut facilement être vérifié que, dans ce protocole, aucune des deux attaques précédentes ne réussira. L'une des propriétés nécessaires d'un algorithme de chiffrement (utilisé par **S**) est qu'il soit impossible de modifier la valeur des messages chiffrés.

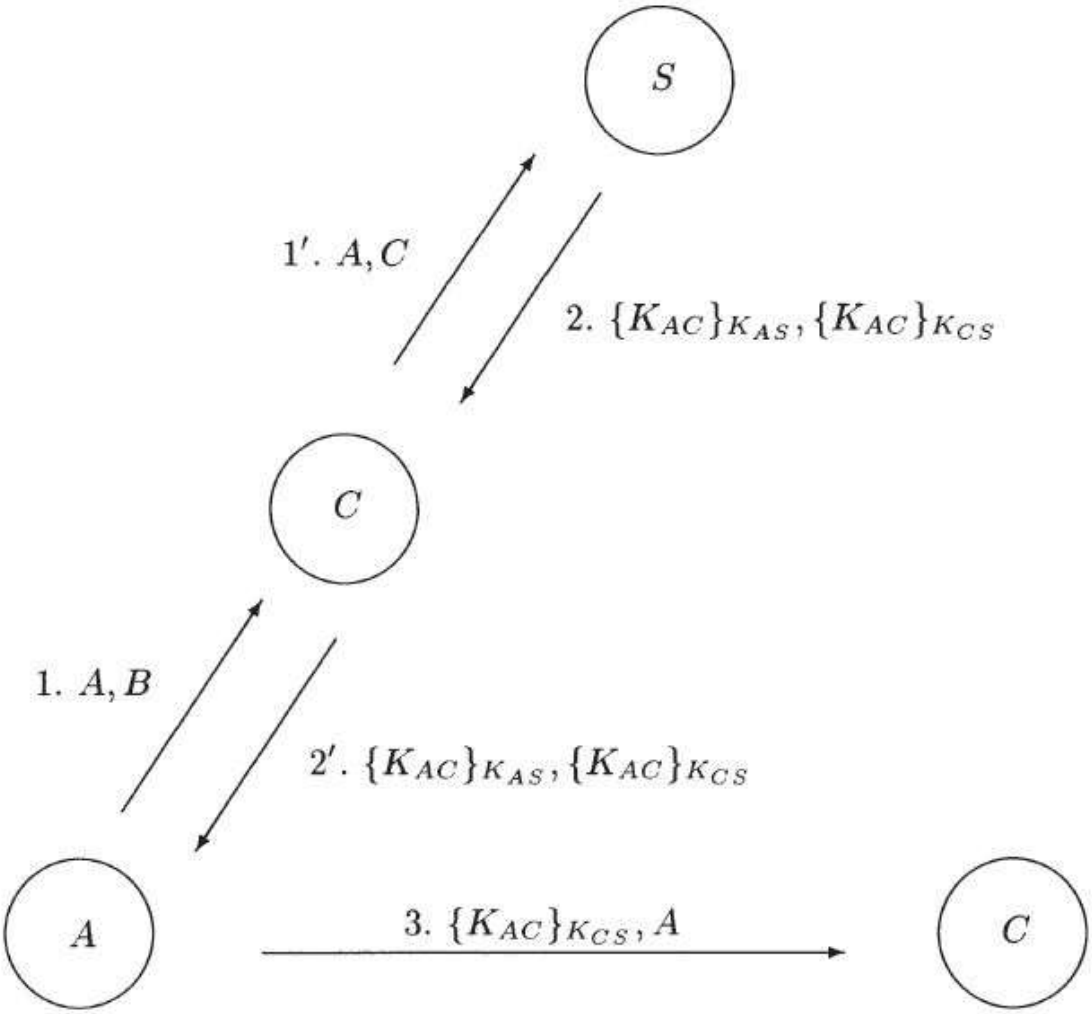


Figure I. 8 - Attaque alternative sur la deuxième tentative du protocole

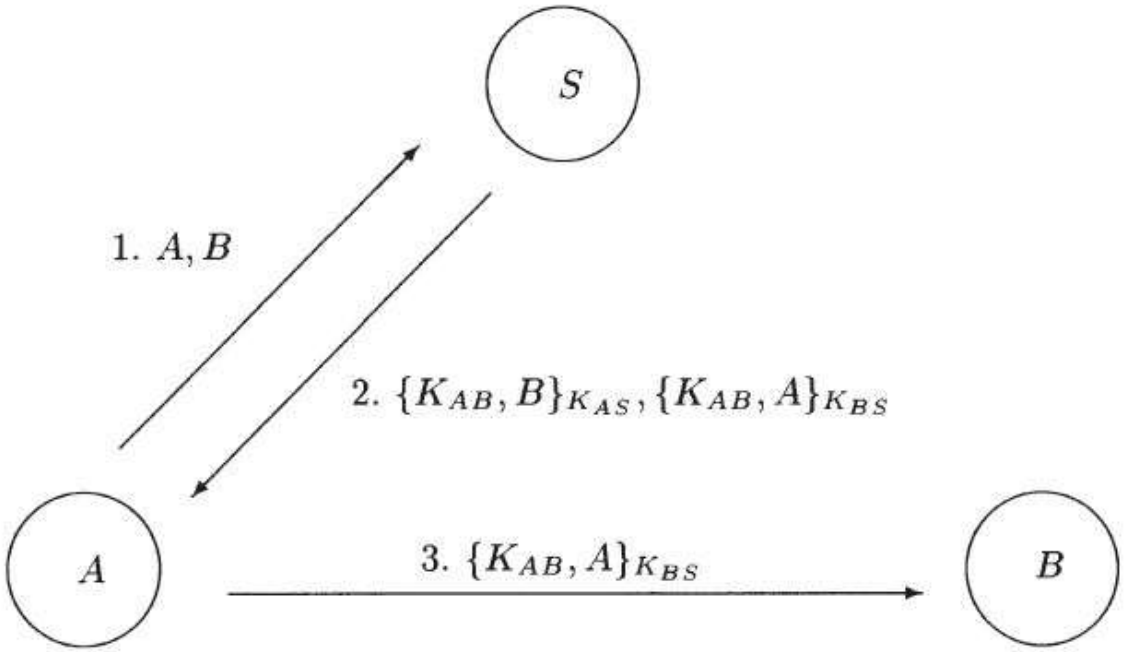


Figure I. 9 - Troisième tentative du protocole

3.1.4. Le problème de rejeu

Jusqu'à présent, notre protocole est amélioré au point où un adversaire est incapable d'attaquer ni par l'écoute ni par la modification des messages envoyés entre les utilisateurs légitimes. Cependant, même à ce stade le protocole ne suffit pas pour assurer la sécurité dans des conditions normales de fonctionnement.

Le problème provient de la différence de qualité entre les clés de cryptage partagées (à long terme) initialement avec **S**, et les clés de session $\mathbf{K_{AB}}$ générées pour chaque session.

La raison pour laquelle une nouvelle clé est générée pour chaque session est que les clés de session sont censées être vulnérables aux attaques. Elles sont susceptibles d'être utilisées avec une variété de formats de données régulières, ce qui les rend cibles pour la cryptanalyse ; aussi elles peuvent être stockées sur des périphériques de stockage relativement incertains et pourraient facilement être mises au rebut négligemment après la fermeture d'une session.

La deuxième raison de l'utilisation de nouvelles clés de session est que les communications dans les différentes sessions doivent être séparées. En particulier, il ne devrait pas être possible de rejouer les messages des sessions précédentes. Pour ces raisons, tout type d'attaque devient possible en se basant sur la notion que les anciennes clés peuvent être rejouées dans une session ultérieure. Notez que même si **A** soit prudent dans la gestion des clés de session utilisées précédemment, la compromission d'une clé de session prise par **B** peut encore permettre des attaques de rejeu lorsqu'**A** communique avec **B**.

Hypothèse de sécurité 4 : Un adversaire est en mesure d'obtenir la valeur de la clé de session $\mathbf{K_{AB}}$ utilisé dans toute session précédente du protocole.

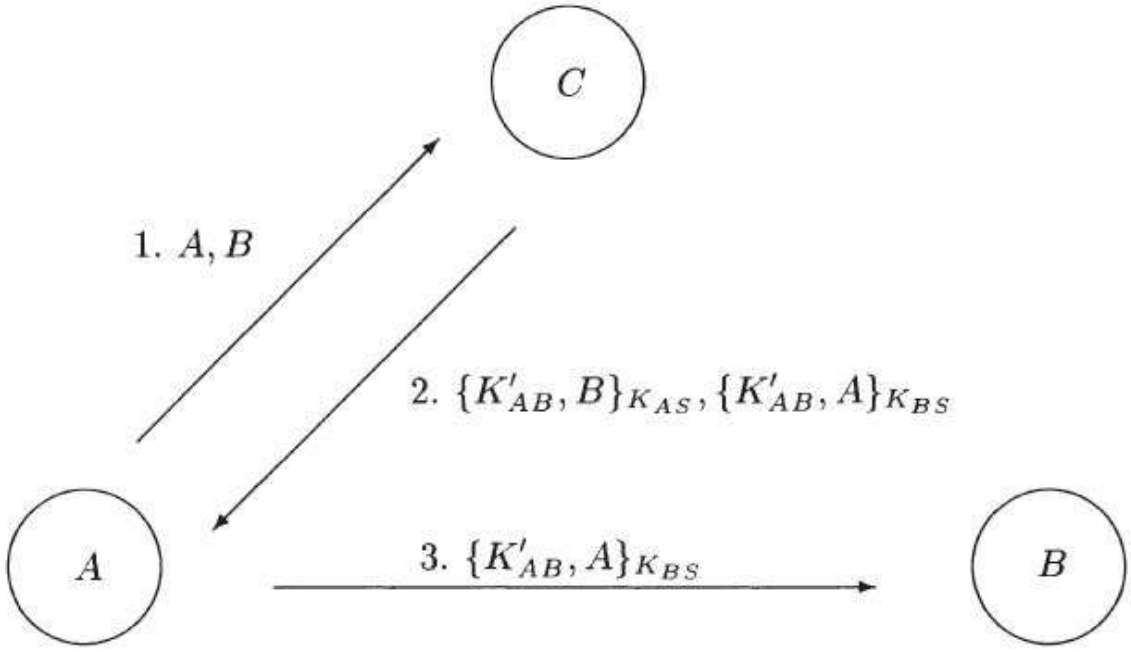


Figure I. 10 - Attaque sur la troisième tentative du protocole

Figure I.10 montre une attaque de rejeu sur notre troisième tentative de protocole. Cette fois **C** intercepte le message de **A** à **S** - en effet **S** ne figure nul part dans le protocole. La clé $\mathbf{K_{AB}}$ est une ancienne clé de session utilisée par **A** et **B** dans une session précédente ; cela peut être attendu par l'hypothèse de sécurité 1. Par l'hypothèse de sécurité 4, **C** peut connaître la valeur de $\mathbf{K_{AB}}$. Ainsi, lorsque **A** accompli le protocole avec **B**, **C** sera capable de déchiffrer des

informations ultérieures chiffrées avec $\mathbf{K'_{AB}}$ ou insérer/modifier des messages dont l'intégrité est protégée par $\mathbf{K'_{AB}}$.

Notez que l'attaque de rejeu à la Figure. I.10 peut encore être considérée comme réussie, même si **C** n'a pas obtenu la valeur de $\mathbf{K_{AB}}$. En effet, **C** a réussi à faire **A** et **B** acceptent une vieille clé de session. Une telle attaque peut être utile de **C**, car elle lui permet de rejouer des messages protégés par $\mathbf{K_{AB}}$ qui ont été envoyés à la session précédente. En outre, elle permet au **C** d'obtenir plus de cryptogrammes avec la même clé qui pourrait être utile à la cryptanalyse.

Il existe divers mécanismes qui peuvent être utilisés pour permettre aux utilisateurs de vérifier que les clés de session ne sont pas rejouées. Ceux-ci sont examinés en détail dans la section (I.5), mais pour l'instant, nous allons améliorer notre protocole en utilisant la méthode la plus populaire appelée défi-réponse. Dans cette méthode, **A** va générer une nouvelle valeur aléatoire $\mathbf{N_A}$ connu comme un nonce (A number used only once).

Définition I.8 : Un nonce est une valeur aléatoire générée par l'une des parties et est retournée à cette partie pour montrer qu'un message est nouvellement généré.

Le participant **A** envoie son nonce $\mathbf{N_A}$ à **S** au début du protocole en même temps que la demande d'une nouvelle clé. Si cette même valeur est reçue avec la clé de session alors **A** peut déduire que la clé n'a pas été rejouée. Cette déduction sera valable tant que la clé de session et le nonce sont liés ensemble cryptographiquement d'une manière telle que seule **S** aurait pu former un tel message. Puisque **B** ne communique pas directement avec le serveur **S**, il est gênant pour lui d'envoyer son nonce à **S**. Comment, alors, est-il capable de gagner la même assurance comme **A** que $\mathbf{K_{AB}}$ n'a pas été rejouée ?

Si le message chiffré pour **B** est inclus dans la partie chiffrée du message de **A**, alors **A** peut obtenir l'assurance que le message est frais. Il est tentant de croire que **A** peut passer cette assurance à **B** dans un Handshake supplémentaire : **B** va générer un nonce $\mathbf{N_B}$ et l'envoyer à **A**, lui-même protégé par $\mathbf{K_{AB}}$. Alors **A** peut utiliser la clé de session pour envoyer une réponse connexe à **B**. Ceci conduit à une quatrième tentative de protocole représenté sur la Figure I.11.

Le protocole que nous avons atteint par une série d'étapes, est l'un des plus célèbres dans le sujet des protocoles cryptographiques. Il a été publié par Needham et Schroeder en 1978 [I.3] et a été à la base de toute une classe de

protocoles connexes. Malheureusement, le protocole Needham Schroeder d'origine est vulnérable à une attaque presque aussi célèbre en raison de Denning et Sacco [I.4].

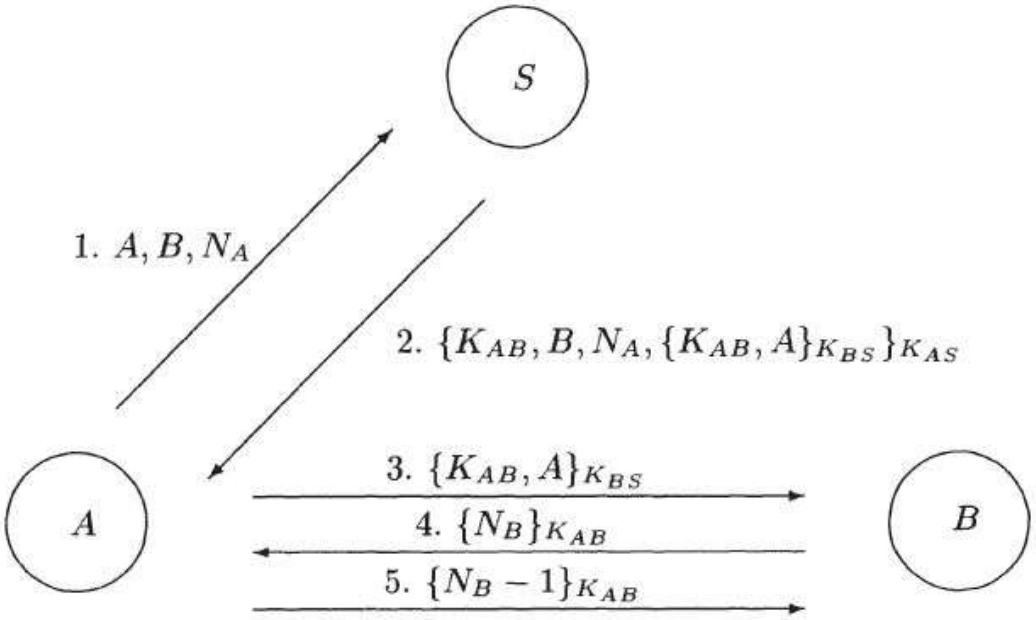


Figure I. 11 - Quatrième tentative du protocole (Needham- Schroeder)

Leur attaque illustre qu'il y avait une faille dans l'argument ci-dessus utilisé pour justifier la conception du protocole. Cela peut être localisé à une hypothèse que seul **A** sera en mesure de former une réponse correcte au message 4 de **B**. Puisque l'adversaire **C** peut connaître la valeur d'une ancienne clé de session, cette hypothèse est irréaliste. Lors de l'attaque sur la Figure I.09, C se déguise comme **A** et est donc en mesure de persuader **B** d'utiliser l'ancienne clé $\mathbf{K'_{AB}}$.

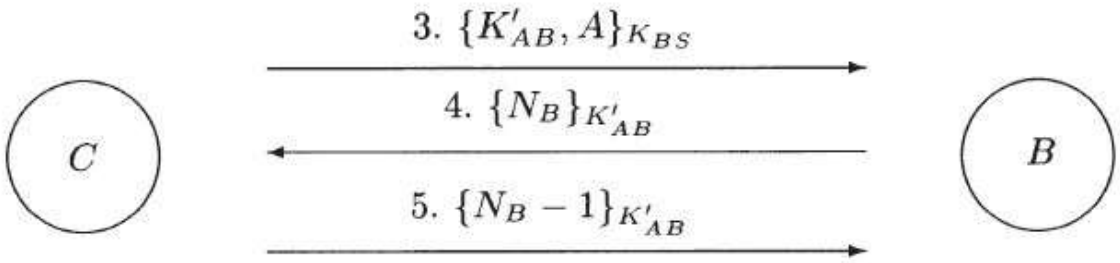


Figure I. 12 - Attaque sur le protocole de Needham-Schroeder

Comme d'habitude, une fois qu'une attaque a été repérée, il est relativement facile de suggérer des moyens pour la surmonter. La méthode que nous choisissons ici est de jeter l'hypothèse selon laquelle il est gênant pour les

participants **A** et **B** d'envoyer leurs défis à **S**. Cela conduit à notre protocole représenté sur la figure. I.13.

Il serait téméraire de prétendre que ce protocole est sécurisé avant de donner un sens précis à ce terme. Pourtant, nous pouvons dire qu'il évite toutes les attaques que nous avons rencontrées jusqu'à présent, tant que l'algorithme cryptographique utilisé fournit les propriétés de la confidentialité et l'intégrité, et le serveur **S** agit correctement. La sécurité d'un protocole doit toujours être considérée par rapport à ses objectifs ; les différents objectifs possibles sont examinés en détail dans la section (I.7).

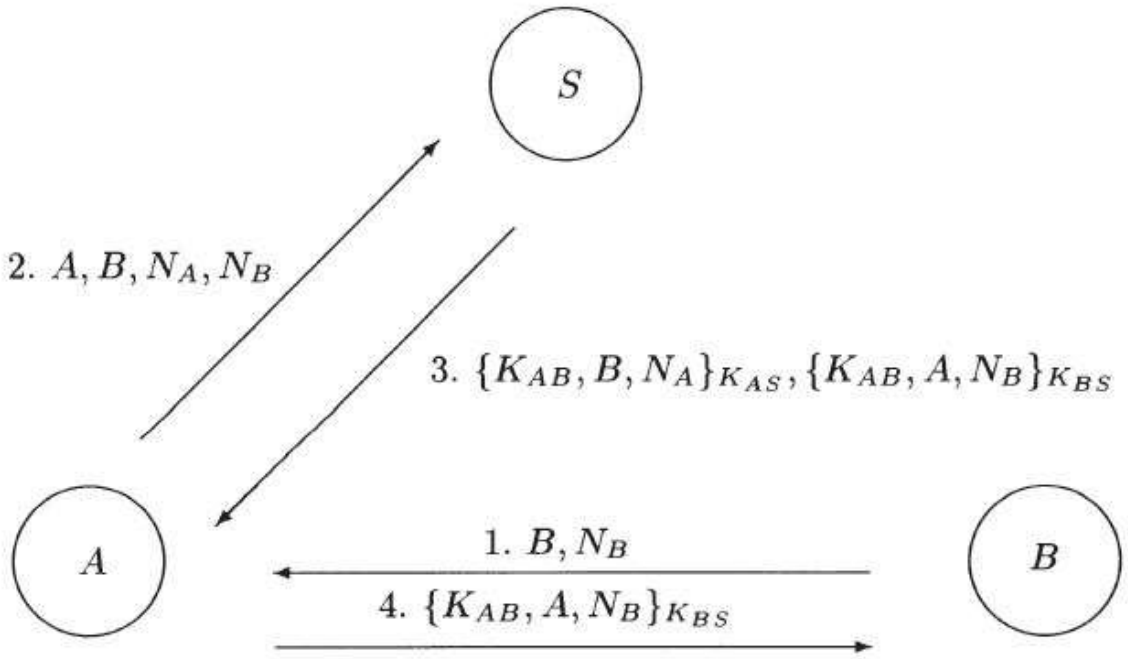


Figure I. 13 - Cinquième tentative du protocole

Pour permettre à la fois aux utilisateurs d'envoyer leur nonce à **S**, le protocole est maintenant initié par **B** qui envoie son nonce, $\mathbf{N_B}$, d'abord à **A**. **A** ajoute son nonce, $\mathbf{N_A}$, et envoie les deux nonces à la fois à **S** qui est maintenant en mesure de retourner $\mathbf{K_{AB}}$ dans des messages séparés pour **A** et **B**, qui peuvent chacun être vérifiés leurs fraicheurs par leurs destinataires respectifs. Bien que cela semble que nous avons obtenu plus que le protocole sur la figure. I.11 en utilisant moins de messages, en effet **A** a obtenu moins dans ce protocole. A cet effet, sur la figure. I.11 **A** pourrait vérifier non seulement que la clé est nouvelle et connue seulement par **A**, **B** et **S**, mais aussi que **B** a en effet reçu la clé de session.

Cette propriété de confirmation de clé est obtenue grâce à l'utilisation de la clé dans le message 4, en supposant que $\{N_B\}_{K_{AB}}$ ne peut être formé sans la connaissance de K_{AB}. Nous allons discuter de confirmation de clé plus loin dans section (I.7). Dans le protocole de la figure. I.13, ni **A** ni **B** ne peuvent déduire à la fin d'une exécution réussite du protocole que l'autre participant a effectivement reçu K_{AB}.

3.2. Les différentes solutions du problème d'authentification

Les différentes solutions citées dans cette section comprennent l'authentification de l'entité ainsi que l'authentification des données.

3.2.1. Authentification par mot de passe

C'est la forme la plus simple, la plus ancienne et la plus courante des méthodes d'authentification. Sa mise en œuvre a évolué avec les découvertes successives de failles de sécurité.

3.2.1.1. Authentification par mot de passe statique

Sur les premiers systèmes Unix, un programme de connexion demandait à l'utilisateur de donner son identifiant puis son mot de passe qu'il comparait à des valeurs stockées à l'avance (très vite, on a découvert qu'il fallait hacher les mots de passe avant de les stocker et n'a guère fallu de temps pour se rendre compte que même cette précaution était inopérante et qu'il fallait trouver une solution de sécurité plus efficaces).

Il est à cet égard intéressant de voir un problème simple de sécurité et les méthodes mises en œuvre pour le résoudre. La tendance à l'économie de l'être humain ainsi que son manque de mémoire amenaient les utilisateurs à choisir des mots de passe faibles, trop courts ou en rapport avec leur nom, prénom, ceux de leurs proches ou à des dates aisées à connaître. Il devenait alors simple de se connecter par essais successifs pour peu que l'on pût en tenter suffisamment. Une réponse technique a alors consisté à exiger des mots de passe plus longs, combinant lettres et chiffres, en majuscules et en minuscules augmentant ainsi le nombre de mots de passe possibles. Parallèlement, on a limité le nombre d'essais autorisés. Mais à ces éléments techniques on a souvent ajouté des éléments psychologiques par exemple en expliquant aux utilisateurs qu'ils pouvaient construire des mots de passe difficiles à deviner en utilisant par exemple les premières lettres des mots d'une phrase simple à mémoriser.

Mais cela n'a pas empêché des intrus d'établir des listes de mots de passe préalablement hachés et de parcourir le fichier de mots de passe du système pour trouver des correspondances. Elles sont d'autant plus importantes que les mots de passe sont faciles à deviner comme expliqué précédemment. Pour remédier à ce problème, Morris et Thompson (1979) ont décrit une technique où le système choisi un nombre aléatoire à chaque nouvelle identification, le concatène au mot de passe rentré par l'utilisateur et applique la fonction de hachage sur le résultat.

Ce procédé, utilisé pour une authentification locale, a été ensuite repris dans la conception des protocoles d'authentification à distance.

3.2.1.2. Authentification par mot de passe dynamique (OTP)

Toujours afin de résoudre les problèmes de sécurité de l'authentification présentés ci-dessus, certains systèmes demandent aux utilisateurs de changer fréquemment de mot de passe. Dans le cas extrême, un mot de passe différent peut être demandé à l'utilisateur à chaque nouvelle authentification. Il est utilisé une seule fois et immédiatement invalidé. Une liste de ces mots de passe est remise à l'utilisateur lors d'une entrevu face à face au moment de l'ouverture de ses droits. Cependant, l'existence d'une telle liste qui peut être égarée, perdue ou volée est en elle-même une faille de sécurité.

Mais la technique du mot de passe à utilisation unique (en anglais One Time Password ou OTP) – aussi mot de passe dynamique – peut être mise en œuvre en remettant à l'utilisateur lors d'une entrevue face à face consécutive à l'ouverture de ses droits un objet, appelé clé OTP. Cet objet, qui n'est pas reproductible, affiche pendant une durée assez courte un code que l'utilisateur doit employer pendant cette durée comme élément de son authentification. Ainsi pendant chaque tranche de temps l'utilisateur sait quel est l'élément d'authentification en cours.

De cette façon, le système authentifiant n'a pas besoin d'enregistrer la liste de mots de passe de chaque utilisateur. De plus un adversaire ne peut pas deviner le prochain mot de passe à utiliser. L'objet OTP a été adopté par de grandes entreprises ou des administrations.

3.2.2. Authentification à base de clé secrète partagée

La majorité des protocoles d'établissement de clés et d'authentification de l'entité qui ont été proposées dans la littérature se concentrent sur le cas où il y

a exactement deux utilisateurs qui souhaitent communiquer ou d'établir une clé de session. Ceci est communément appelé le cas de deux parties. Dans cette section, nous discutons des protocoles d'authentification et d'établissement de clés de deux parties basés sur des algorithmes symétriques. Tandis que la section suivante traite des protocoles de deux parties en utilisant des algorithmes à clé publique.

On peut classer les protocoles d'établissement de clés de deux parties à l'aide des deux critères suivants :

1. Les clés cryptographiques disponibles a priori.
2. La méthode de génération des clés de session.

Si seulement des clés partagées sont disponibles pour établir une nouvelle clé de session, il existe essentiellement deux cas à considérer en ce qui concerne le critère 1 :

a. Les deux participants partagent au préalable une clé secrète.

b. Chaque participant partage une clé avec un serveur de confiance.

Critère 2 concerne les méthodes de génération de clés de session, pour lesquels il existe trois possibilités différentes en général : transport de clé, accord de clé et hybride.

Si un protocole n'a que deux participants sans le biais d'un serveur de confiance, comme dans le cas (a) ci-dessus, on ne peut pas distinguer entre le transport de clé et la génération hybride de clé.

Les critères mentionnés ci-dessus conduisent à la classification de 1 x 2 + 1 x 3 = 5 classes différentes de protocoles.

La reconnaissance des critères permet de classer les protocoles à deux parties dans la littérature dans l'une des cinq classes ci-dessus. Cependant, dans cette section, nous insistons principalement sur les protocoles sans l'intermédiaire d'un serveur de confiance, les protocoles basés sur serveur de confiance seront discutés dans la section (3.2.8).

3.2.2.1. Protocoles concernant l'authentification de l'entité seulement

Les protocoles qui visent à fournir une authentification de l'entité sans établissement de clé sont relativement rares dans la littérature. C'est peut-être parce que la variété des approches est assez limitée.

Les protocoles Bird-Gopal-Herzberg-Janson-Kutten-Molva-Yung

Un article des chercheurs d'IBM en 1993 [I. 37] a été l'un des premiers à démontrer une large classe d'attaques sur plusieurs protocoles d'authentification, y compris un projet de protocole proposé par l'ISO. Sur la base des attaques contre ces protocoles, les auteurs développent un ensemble de critères de sécurité pour éviter de telles attaques et proposent des protocoles qui répondent à leurs critères. Ils commencent par un protocole de base qu'ils ne considèrent pas comme sûr, et l'améliorent en une série d'étapes à travers l'examen de diverses attaques et d'autres exigences de conception. Finalement, un bon protocole est atteint.

Les protocoles ISO/lEC 9798-2

La norme internationale ISO / IEC 9798 Partie 2 [I.38] spécifie six protocoles utilisant des algorithmes de cryptage symétriques. Quatre de ces protocoles sont destinés à fournir l'authentification de l'entité seulement, tandis que deux sont destinés à fournir l'établissement de clés ainsi que l'authentification de l'entité. Deux des quatre protocoles visant uniquement l'authentification d'entité concernent l'authentification unilatérale, tandis que les deux autres concernent l'authentification mutuelle.

3.2.2.2. Protocoles concernant l'établissement de clé

Les protocoles considérés exigent que les deux utilisateurs partagent déjà une clé secrète à long terme et peuvent exiger qu'un utilisateur génère la clé établie (transport de clé) ou que les deux utilisateurs apportent une partie de la clé établie (accord de clé).

Le protocole de Andrew Secure RPC

Le protocole d'Andrew Secure RPC [I.39], est encore un exemple largement utilisé dans la littérature. Burrows et al. [I.24] ont souligné un problème majeur avec le protocole. L'initiateur n'a aucune assurance de la fraicheur de la clé établie. Un autre problème a été signalé par Clark et Jacob [I.40]. Il s'agit d'une attaque par mauvaise interprétation du message (Typing attack).

Le protocole de Janson-Tsudik 2PKDP

Janson et Tsudik [I.41] ont proposé un protocole qui s'étend du protocole d'authentification de Bird et al. [I.37] pour assurer l'établissement de clé. Un aspect distinctif de 2PKDP est qu'il emploie deux algorithmes cryptographiques

: un algorithme qui assure la confidentialité et un autre algorithme qui fournit l'authentification.

Le protocole de Boyd Two-Pass

Le protocole proposé par Boyd [I.42] permet à deux participants de contribuer par une partie de la clé établie. Les messages envoyés sont simplement des nombres aléatoires choisis par les deux participants.

La nouvelle clé générée est $\mathbf{K = f\ (N_A,\ N_B,\ K_{AB})}$, où f est une fonction de combinaison telle qu'il ne soit pas possible de trouver $\mathbf{f\ (.,.,\ K_{AB})}$ sans la connaissance de $\mathbf{K_{AB}}$, même après une utilisation répétée. Cette propriété est nécessaire pour assurer l'authentification des clés. Une autre propriété requise de f est qu'elle doit être unidirectionnelle dans les deux premières entrées. Cette propriété est nécessaire pour assurer la fraîcheur des clés.

Les protocoles d'ISO/lEC 11770-2

La norme internationale ISO/lEC 11770 Partie 2 [I.46] spécifie 13 protocoles utilisant des algorithmes de cryptage symétrique. Six de ces protocoles sont sans l'intermédiaire d'un serveur de confiance, tandis que les sept autres s'appuient sur un serveur de confiance. Certains protocoles utilisent une fonction de dérivation de clé $\mathbf{f\ (\cdot)}$ pour former la nouvelle clé de session à partir de deux entrées ou plus. La fonction f peut être : une opération de XOR bit à bit des entrées ou l'application d'une fonction de hachage à la concaténation des entrées.

3.2.3. Authentification à base de clé publique

Il est généralement considéré qu'il y a deux principaux avantages potentiels des techniques à clé publique sur la cryptographie symétrique. Tout d'abord les systèmes à clés publiques permettent la définition directe des signatures numériques, permettant ainsi le service de non-répudiation qui est si utile dans les applications commerciales. La seconde est la simplification de la gestion des clés, car il n'y a pas d'obligation de présence de tierce en ligne qui fait partie des protocoles typiques basés sur la cryptographie symétrique.

Le premier de ces avantages n'est pas vraiment notre préoccupation dans cette thèse. Cependant, le deuxième avantage a conduit à une grande variété de nouveaux protocoles d'établissement de clé depuis l'invention de la cryptographie à clé publique. Dans les environnements de communication modernes distribués illustrés par l'Internet, les protocoles à base de clés

publique sont devenus beaucoup plus importants que les protocoles basés sur la cryptographie symétrique.

Il y a deux coûts qui doivent être payés en échange de ces avantages. Le premier est le coût de calcul élevé qui vient avec tous les systèmes cryptographiques à clé publique connue. Malgré les progrès réalisés dans la cryptographie à clé publique et les avantages de la courbe elliptique de cryptologie [I.34], les algorithmes à base de clé publique nécessitent deux ou trois ordres de grandeur plus de calculs que les algorithmes symétriques.

Avec l'augmentation de puissance de calcul sur un ordinateur de bureau typique. Les opérations à base de clés publiques ne causeront pas un retard de plus d'une fraction de seconde. Par contre, les frais généraux pour les serveurs à plusieurs clients et des dispositifs informatiques à faible puissance, tels que les cartes à puce, restent encore significatifs.

Il est donc essentiel pour les concepteurs de protocoles à base de clés publiques de réduire au maximum le nombre d'opérations chaque fois qu'il est possible.

Une autre question à considérer est de savoir si le protocole nécessite plus d'opérations de clés privées (générations de signature et décryptages) ou plus d'opérations de clés publiques (vérifications de signature et cryptages). RSA et les algorithmes connexes sont beaucoup plus efficaces pour les opérations de clés publiques que pour les opérations de clés privées, tandis que pour la plupart des algorithmes basés sur les logarithmes discrets, le contraire est prouvé vrai. En outre, bien que les algorithmes basés sur des logarithmes discrets (y compris les algorithmes de courbe elliptique) soient plus efficaces que RSA dans l'ensemble, un protocole qui nécessite principalement des opérations de clé publique peut être beaucoup plus efficacement mis en œuvre en utilisant RSA avec un petit exposant public. Généralement, il n'est pas aisé de comparer l'efficacité de différents protocoles lorsqu'ils sont implémentés en utilisant différents algorithmes.

Le deuxième coût supplémentaire de la cryptographie à clé publique est que les clés publiques doivent encore être gérées. La meilleure solution pour une infrastructure à clé publique (PKI) fait encore l'objet de recherches considérables. Bien que les clés publiques ne soient pas tenues confidentielles, elles doivent être maintenues, normalement au moyen de certificats signés par des tiers de bonne réputation. La question de savoir comment traiter les clés privées compromises est délicate, mais la solution la plus établie consiste à

utiliser les listes de révocation de certificats pour vérifier si une clé publique est toujours valide, tout comme une liste noire est vérifiée avant d'accepter une carte de crédit. Dans les descriptions des protocoles donnés dans cette section, nous ignorerons les certificats ou tout autre moyen utilisé pour garantir la validité d'une clé publique et supposer simplement que les clés publiques sont à la disposition des parties qui en ont besoin et sont garanties qu'elles sont correctes . Ce que signifie correct ici est que le propriétaire revendiqué de la clé publique est la seule entité qui est capable d'utiliser la clé privée correspondante. Nous expliquons le fonctionnement des PKI dans la section (3.2.7).

Anderson et Needham [I.45] ont proposé un ensemble de ce qu'ils appellent des principes de robustesse pour les protocoles à clé publique. Ceux-ci peuvent être considérés comme des cas plus spécifiques des principes généraux pour la conception de protocoles proposés plus tôt par Abadi et Needham. Ils forment une liste de contrôle qui peut être utilisée par les concepteurs de protocole pour éviter les erreurs les plus courantes.

Plus tard, Syverson [I.47] s'est interrogé sur l'applicabilité de certains des principes proposés par Anderson et Needham en montrant des exemples lorsqu'ils ne sont pas appropriés. Néanmoins, Syverson conclut que les principes sont encore utiles mais devraient être utilisés de manière intelligente et critiqués par le concepteur de protocole.

Dans cette section nous examinons brièvement les protocoles qui réalisent seulement l'authentification de l'entité.

3.2.3.1. Protocoles dans ISO/IEC 9798-3

Cinq protocoles sont inclus dans l'ISO / IEC 9798-3 pour l'authentification entre une paire de participants. Deux d'entre eux sont pour l'authentification unilatérale d'une partie à une autre, et trois sont pour l'authentification mutuelle des deux parties l'une à l'autre. Deux des protocoles sont également inclus dans la norme américaine FIPS 196 [I.48], qui comprend des indications supplémentaires sur l'utilisation de champs facultatifs.

3.2.3.2. Protocoles dans ISO / IEC 9798-5

La norme internationale ISO / IEC 9798-5 [I.49] est consacrée aux mécanismes d'authentification d'entités utilisant des techniques de preuve à divulgation nulle de connaissance «zero knowledge».

Trois protocoles sont spécifiés. Les deux premiers portent essentiellement sur les protocoles bien connus d'identification de la preuve à divulgation nulle de connaissance de Fiat-Shamir [I.50], Guillou-Quisquater [I.51] et Schnorr [I.52]. Ces protocoles dépendent fortement de mécanismes cryptographiques spécifiques et sont conçus pour prouver la connaissance des clés privées correspondant aux clés publiques connues. Le troisième protocole fournit également une authentification unilatérale sans connaissance de l'entité homologue.

3.2.3.3. Le protocole SPLICE/AS

Le protocole connu sous le nom de SPLICE / AS a été proposé en 1990 par Yamaguchi et al. [I.53] pour fournir une authentification mutuelle entre le client et le serveur. Un certain nombre de documents différents ont progressivement trouvé des attaques et des améliorations proposées. Le protocole complet comprend la récupération de clés publiques certifiées à partir d'un serveur d'authentification. Hwang et Chen [I.54] ont montré que le format original du certificat était erroné, ce qui permet à un adversaire de modifier la clé publique apparente du client ou du serveur.

3.2.4. Authentification à base de mot de passe et échange de clé « PAKE »

L'authentification cryptographique repose sur la possession d'une clé par la partie afin d'être authentifié. Une telle clé est généralement choisie au hasard dans son domaine et peuvent avoir des longueurs d'environ 100 bits jusqu'à plusieurs milliers de bits, en fonction de l'algorithme utilisé et le niveau de sécurité souhaité. L'expérience a montré que les humains trouvent qu'il est difficile de se souvenir des secrets sous la forme de mots de passe de mêmes sept ou huit caractères. Mais si toutes les lettres majuscules et minuscules sont utilisées conjointement avec les chiffres 0 à 9 puis un mot de passe de huit caractères aléatoires représente moins de 48 bits aléatoires.

Les clés cryptographiques sont souvent stockés dans une mémoire sécurisée dans les ordinateurs ou à l'aide de dispositifs spéciaux tels que les serveurs cryptographiques inviolables ou les cartes à puce. Cependant, il y a des situations où cela est gênant et coûteux. Pas tous les périphériques ne sont inviolables, et la mémoire nécessaire pour les clés publiques peuvent être rare.

Il est donc souhaitable de pouvoir mettre en place des communications sécurisées reposant uniquement sur un court secret qui peut être rappelé par les humains.

À première vue, il peut sembler impossible de parvenir à l'établissement de clés en utilisant seulement un court secret (un mot de passe à faible entropie) de telle sorte que la recherche par force brute pour trouver le secret est impossible. Cette intuition peut expliquer pourquoi il n'a pas été jusqu'en 1989 que les premiers protocoles à base de mot de passe sont apparus dans la littérature. Ces premiers protocoles, en raison de Lomas et al. [I.35] ont utilisé l'hypothèse supplémentaire que le client (dans une application client-serveur) a la connaissance de la clé publique du serveur, en plus le mot de passe partagé avec le serveur. Plus tard Bellovin et Merritt [I.36] ont introduit une classe de protocoles qui ne nécessitent pas cette hypothèse.

L'idée de Bellovin et de Merritt des protocoles EKE (Encrypted Key Exchange) est que l'initiateur du protocole choisira une clé publique éphémère et utilise le mot de passe partagé afin de crypter cette clé. Le répondeur peut déchiffrer la clé publique et l'utiliser pour envoyer en retour la clé de session en toute sécurité à l'initiateur. Dans l'hypothèse (pas toujours raisonnable) que les clés publiques sont des chaînes aléatoires, un adversaire qui tente une recherche de mots de passe par force brute ne sera pas en mesure de distinguer quelle clé publique éphémère a été utilisée ; En outre, même lorsque la clé publique correcte a été trouvée, elle ne peut pas être utilisée pour découvrir la clé de session, car il est impossible d'obtenir la clé privée à partir de la clé publique. Il est intéressant de comparer les protocoles à base de mot de passe avec les protocoles fournissant la confidentialité persistante.

Nous pouvons résumer les propriétés particulières et les exigences pour les protocoles d'établissement de clés à base de mot de passe comme suit.

• Les utilisateurs ne possèdent qu'un secret de faible entropie. Plus précisément, il est possible pour l'adversaire de rechercher dans tous les secrets possibles dans un délai raisonnable.

• Les attaques par dictionnaire hors ligne ne devraient pas être possibles. Cela signifie qu'un indiscret passif qui enregistre la transcription d'une ou plusieurs sessions ne peut pas éliminer un nombre important de mots de passe possibles.

• Les attaques par dictionnaire en ligne ne devraient pas être possibles. Cela signifie que l'adversaire actif ne peut pas abuser le protocole de façon à éliminer

un nombre important de mots de passe possibles. Un adversaire actif peut toujours éliminer au moins un mot de passe par tentative de se faire passer pour l'utilisateur légitime en utilisant ce mot de passe. Idéalement, cela devrait être tout ce que l'adversaire pourrait gagner.

D'autres objets, tels que (La confidentialité persistante), sont souvent considérées comme souhaitables [I.33].

Il y a eu de nombreuses variantes de protocole d'EKE de Bellovin et Merritt, ainsi que de nombreux autres protocoles. Récemment il a été démontré que certaines d'entre elles ont des propriétés de sécurité éprouvées.

Le protocole EKE original ne précise pas l'algorithme de chiffrement à utiliser ou comment transformer le mot de passe à une clé correspondante. Certains protocoles ultérieurs ont utilisé des algorithmes spécifiques qui nécessitent généralement un moyen pour mapper le mot de passe partagé à un élément du groupe dans lequel un échange Diffie-Hellman a lieu.

La plupart des protocoles ont des variantes dans lesquelles le serveur stocke seulement une image du mot de passe du client plutôt que le mot de passe lui-même. Cela signifie que la compromission du serveur ne compromet pas automatiquement le mot de passe ; cependant, il permettra une attaque de recherche par force brute donc ce n'est pas nécessairement un avantage significatif. Certains auteurs ont exploré davantage la situation dans laquelle le client a accès à une clé publique du serveur et il y a maintenant des protocoles avec une sécurité prouvée dans une telle situation.

La majorité des protocoles proposés utilisent l'accord de clé basée sur Diffie-Hellman avec le mot de passe partagé utilisé à des fins d'authentification.

3.2.5. Code d'authentification de message «MAC»

En cryptographie, un code d'authentification de message (MAC) est une courte information utilisée pour authentifier un message, c'est-à-dire pour confirmer que le message provient bien de l'expéditeur déclaré (son authenticité) et que ce message n'a pas été modifié en transit (son intégrité).

Cependant, on peut souhaiter de l'authentification sans pour autant rechercher la confidentialité. Dans ce cas, le message devrait être envoyé en clair pour que tous puissent voir le contenu, et le chiffré devrait être envoyé au destinataire spécifique que l'on veut convaincre de l'authenticité. La longueur totale du

message authentifié est donc le double de la longueur du message initial. Ceci peut être un véritable inconvénient lorsqu'il s'agit d'un gros document, tels un livre ou un film.

Les MAC permettent de résoudre ce problème de façon beaucoup plus efficace en produisant un certificat d'authenticité très court (voir figure I.4) : ∑ prend en entrée un message de longueur quelconque, ainsi qu'une clé k, typiquement sur 64 ou 128 bits, et retourne un certificat d'authenticité sur 64 ou 128 bits.

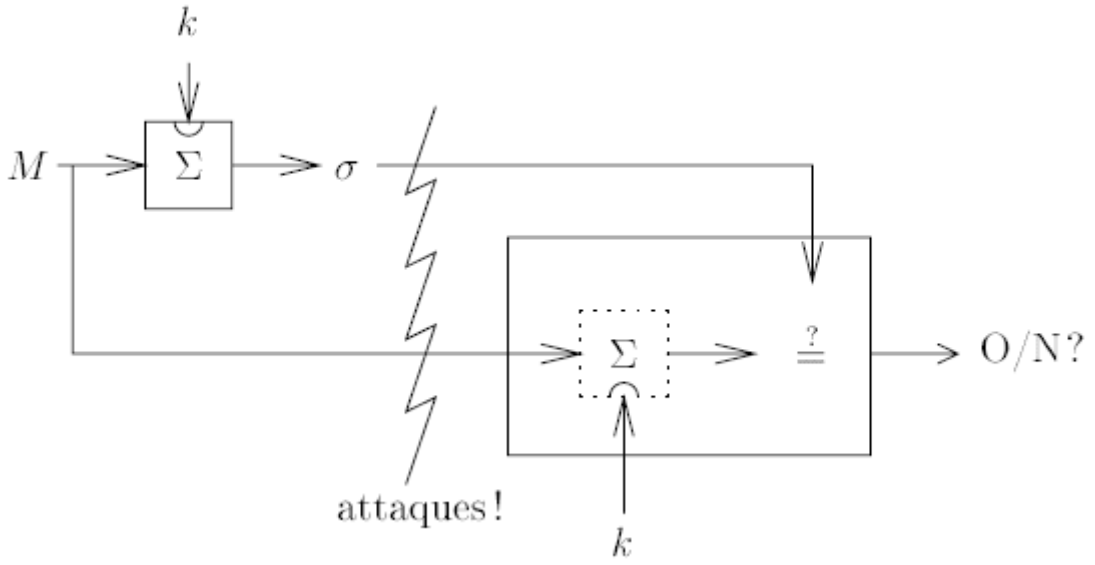


Figure I. 14 - Code d'authentification de message (MAC)

Définition I.9 : Le code d'authentification de message (***MAC***) est une famille de fonctions paramétré par une clé ***k*** tel que ***MAC*** $_k$ ***(m)*** prend un message **m** de longueur arbitraire et délivre une valeur de longueur fixe en satisfaisant :

1. il est informatiquement facile de calculer ***MAC*** $_k$ ***(m)*** étant donné ***k*** et ***m*** ;
2. étant donné un certain nombre de valeurs ***MAC*** pour un ***K*** donné, il est informatiquement difficile de trouver une valeur ***MAC*** valable pour tout nouveau message.

À la réception, le destinataire fait subir la même transformation au message reçu, et vérifie l'égalité avec le certificat. L'authenticité est effectivement garantie si le MAC est sûr contre les « falsifications existentielles à messages choisis ». [I.44]

3.2.6. Authentification par signature électronique

L'authentification par signature électronique est souvent souhaitable lorsqu'on veut assurer le service de la non-répudiation qui est souhaitable souvent dans les applications commerciales.

Les signatures électroniques diffèrent des MAC car les valeurs MAC sont générées et vérifiées à l'aide de la même clé secrète. Cela implique que l'expéditeur et le récepteur d'un message doivent s'entendre sur la même clé avant de lancer des communications, comme c'est le cas avec le cryptage symétrique. Tandis que les valeurs de la signature numérique sont générées par la clé privée et vérifiées par la clé publique. Pour la même raison, les MAC ne fournissent pas la propriété de non-répudiation offerte par les signatures électroniques ; En particulier dans le cas d'une clé secrète partagée à l'échelle d'un réseau : tout utilisateur qui peut vérifier un MAC est également capable de générer des MAC pour d'autres messages. En revanche, une signature électronique est générée à l'aide de la clé privée d'une paire de clés. Étant donné que cette clé privée n'est accessible qu'à son titulaire, une signature numérique prouve qu'un document n'a été signé que par le titulaire de la clé et non pas par un autre.

La signature est infalsifiable car c'est la clef privée qui l'a généré au moment de la signature. La signature n'est pas réutilisable car elle fait partie intégrante du document. Le document est immuable car la moindre falsification sur le document provoquerait une erreur lors du déchiffrement de celui-ci. Cependant, au lieu de signer le document lui-même, il est plus préférable de signer l'empreinte du document (après avoir effectué un hash) car elle est de taille fixe et comme les chances d'avoir deux documents différents ayant la même empreinte est très faible, signer l'empreinte est aussi fiable que de signer le document tout entier. Pour vérifier l'identité de l'émetteur, il suffit de calculer, à partir du document original, l'empreinte par la fonction de hachage et de déchiffrer la signature de l'émetteur avec sa clef publique. Si les deux empreintes sont identiques, alors nous avons vérifié l'identité de l'émetteur et par la même occasion l'intégrité du document.

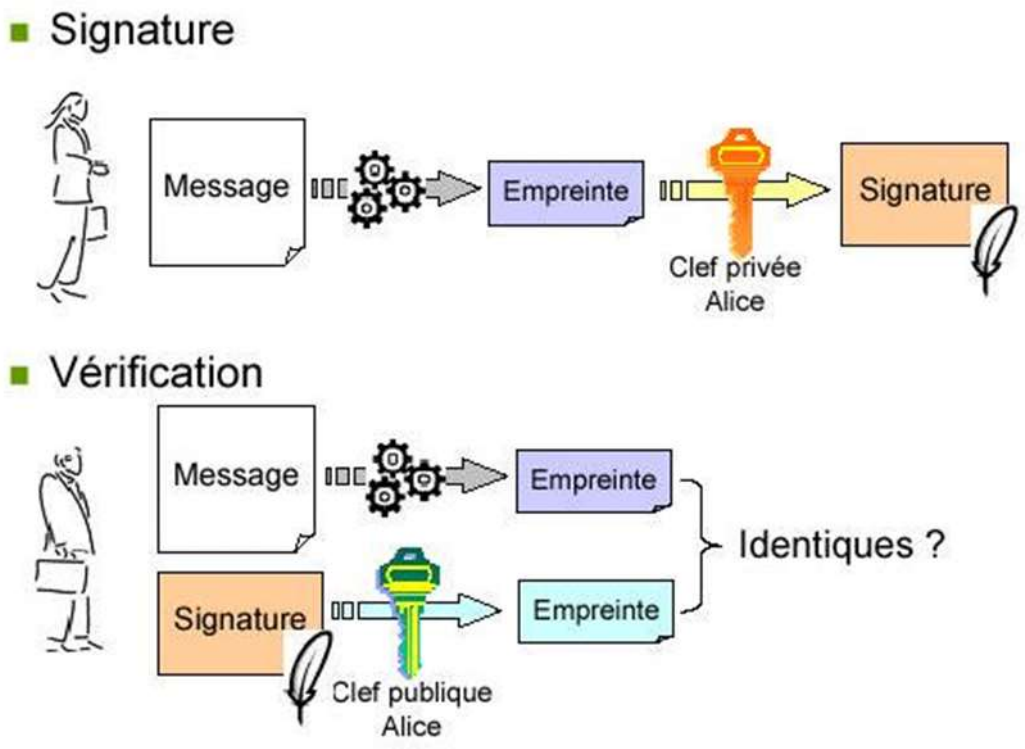


Figure I. 15 - Le procédé de la signature pour l'authentification

Définition I.10 : Un algorithme de signature numérique se compose de trois ensembles : un ensemble de clés ***K***, un ensemble de messages ***M*** et un ensemble de signatures ***S*** avec trois algorithmes :

1. Un algorithme de génération de clé, qui délivre une clé de signature valide $k \in K$ et une clé de validation valide $k^{-1} \in K$;
2. Un algorithme de génération de signature, qui prend en entrée un élément $m \in M$ et une clé de signature $k \in K$ et sort un élément $s \in S$. Nous écrirons $s = Sig_A\{m\}$ où k est la clé de génération de signature de l'entité A. L'algorithme de génération de signature peut être randomisé de sorte qu'une sortie différente se produise avec le même m.
3. Une fonction de vérification, qui prend une signature $s \in S$, un message $m \in M$, et une clé de vérification $k^{-1} \in K$ et émet en sortie un élément $v \in \{0, 1\}$. Si $v = 1$ alors on dit que la signature est valide sinon $v = 0$ on dit que la signature est invalide.

Mais en réalité, Bob ne peut pas être sûr que le message provient bien d'Alice (voir figure I.15). Tout ce dont il est certain, c'est qu'à la clé publique jointe au message correspond bien la clé privée qui a servi à signer le message. En effet, le nom d'Alice ne figure nulle part.

C'est pourquoi, il est nécessaire d'introduire les certificats numériques. Nous allons voir comment ces derniers vont permettre d'instaurer un état de confiance ; c'est-à-dire en prouvant l'identité de l'expéditeur.

3.2.7. Authentification à base de certificat numérique et l'infrastructure « PKI »

Un certificat permet d'associer une clé publique à une entité afin d'en assurer la validité. De façon plus imagée il correspond au passeport ou à la carte d'identité que nous possédons. Elle est délivrée par un organisme appelé autorité de certification (souvent notée CA). Le standard le plus employé pour les certificats numériques est le standard X.509.

3.2.7.1. Structure d'un certificat X.509

Ce certificat est divisé en deux parties :

• La partie contenant les informations d'identité sur l'émetteur et sur l'autorité de certification (CA) ;

• La partie contenant la signature de l'autorité de certification. Comme illustré ci-dessous (Figure I.16)

• Version du protocole X509 ;

• Numéro de série : Valeur entière donnée par l'autorité de certification, elle est unique pour chaque certificat ;

• Identificateur de l'algorithme : Ce champ désigne l'algorithme utilisé par l'autorité de certification pour signer le certificat, ainsi que tous les paramètres de l'algorithme. Il s'avère peu utile car le type d'algorithme de signature est contenu dans la partie signature ;

• Délivreur du certificat : Permet d'identifier l'autorité de certification qui a délivré le certificat. Il existe un formalisme bien défini pour attribuer un nom à chaque entité sans ambigüité (la position géographique entre en compte) ;

• Période de validité : Période durant laquelle le certificat est valide (une date de début et une date de fin) ;

• Sujet du certificat : L'objet de l'utilisation de la clé publique ;

• Clé publique du sujet : Clé publique de l'entité à authentifier. De plus il identifie l'algorithme à utiliser et les paramètres associés à ce dernier ;

• Extensions (facultatif) : Ce champ a été introduit dans la version 3 du standard X509. Il permet aux autorités de certification de rajouter leurs propres informations aux certificats qu'elles délivrent.

• Contient l'identifiant de l'algorithme (fonction de hachage) utilisé par l'autorité de certification pour signer le certificat, et la valeur de signature sur l'ensemble des champs précédents.

L'avantage majeur des certificats X509 est qu'ils présentent une utilisation très flexible par ajout de champs si besoin.

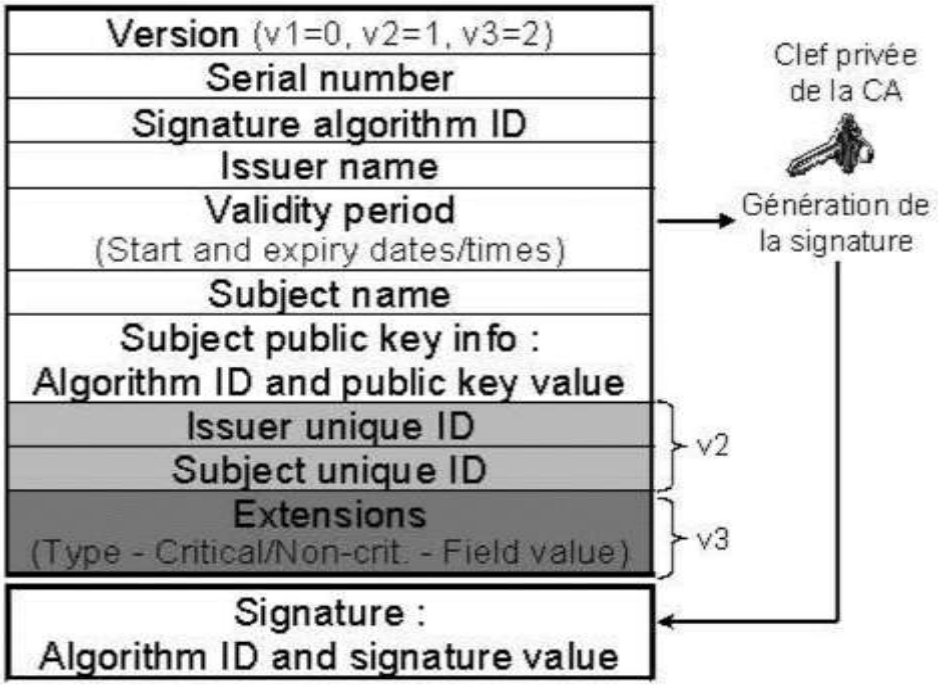


Figure I. 16 - Structure d'un certificat X.509

3.2.7.2. La tierce de confiance (Autorité de Certification)

De manière générale, un tiers de confiance est un organisme habilité à mettre en œuvre des certificats. Dans notre exemple d'Alice et Bob, au lieu de communiquer directement entre eux, Alice et Bob vont faire confiance à une tierce personne. Cette tierce personne va leur assurer que tous les messages qui transiteront par lui seront confidentiels, identifiés et immuables. Cependant, Alice et Bob devront lui faire entièrement confiance. Pour cela, ils devront par eux même vérifier l'identité de cette tierce personne.

Un émetteur fait intervenir une tiers de confiance (nommée CA) qui, sur réception d'une demande de certificat, vérifie que la clef publique de l'émetteur correspond bien à son identité. Ensuite, le CA crée un certificat, puis signe ce certificat en condensant les informations qu'il contient et en les chiffrant à l'aide de sa clef privée (cette signature qui lui donnera toute crédibilité aux yeux des utilisateurs). A la réception de ce certificat, l'entité voulant vérifier l'authenticité du certificat applique la clef publique du CA au condensé. Si le déchiffrement fonctionne, cela signifie que le certificat provient bien du CA. Il applique ensuite à la partie lisible du certificat le même algorithme de condensation et compare le résultat avec le condensé reçu. S'il y a identité, cela signifie que le certificat n'a pas été modifié en cours de transmission. Cette authentification de l'émetteur est complétée d'un traitement similaire appliqué au document ou au message, afin de garantir qu'il n'a pas été modifié non plus en cours de transmission.

Ceci nous amène aux Infrastructures de Gestion de Clefs (ICP) ou encore (PKI) qui est un ensemble de composants et de procédures humaines en vue de gérer le cycle de vie des certificats numériques.

3.2.7.3. Public Key Infrastructure (PKI)

La gestion des clefs

La gestion des clefs de l'infrastructure doit être rigoureuse. En effet, il a été démontré dans les faits qu'il est beaucoup plus facile de s'introduire dans un système et de se procurer illicitement les clefs plutôt que de casser un algorithme. Et le moment le plus propice pour espérer se procurer les clefs est sans conteste le moment où l'échange des clefs a lieu. C'est pourquoi, l'échange des clefs doit être fait avec la plus grande prudence car il représente le point de vulnérabilité de tout le système.

La gestion des clés proprement dite se compose des opérations suivantes :

• **Génération :** Les clefs doivent être générées de manière aléatoirement, de manière à ce qu'elles soient non prédictibles. La prédictibilité dans le processus de création de clef peut compromettre tout le système de sécurité.

• **Distribution :** La distribution est l'action de déplacer une clef de cryptage. Un exemple de distribution est la clef de session, on va créer une clef qui va permettre le transport d'une autre clef.

• **Stockage :** La clef doit être protégée et doit garder à tout prix son intégrité et sa confidentialité. Le contrôle d'accès peut assurer l'intégrité et l'authenticité,

mais seul un stockage sur support hardware peut assurer la confidentialité de la clef.

• **Suppression :** La suppression de clefs intervient quand la clef à atteint sa fin de validité ou lorsqu'un doute subsiste sur sa confidentialité. La suppression signifie la destruction des toutes les copies de la clef symétrique ou de la clef publique. Cependant, si le système permet l'archivage des clefs, alors celles-ci seront archivées plutôt que supprimer.

• **Archivage :** L'archivage des clefs permet de conserver une copie des clefs même si elles ne sont plus utilisées, le but est de pouvoir valider des données qui ont été précédemment protégées par cette clef. Dans tous les cas, une clef archivée ne peut pas être remise en service dans un environnement d'application.

• **Recouvrement :** Le recouvrement des clefs est une procédure délicate qui permet de retrouver la clef privée d'un client. Par exemple lorsqu'un utilisateur a perdu sa clef privée. Ce principe permet aussi le recouvrement des données chiffrées par cette clef. En effet, si cette clef est perdue, alors toutes les données seront perdues par la même occasion. C'est pourquoi, le recouvrement des clefs peut être une solution pour permettre le recouvrement des données.

Toutes ces étapes doivent être minutieusement effectuées et contrôlées pour que la PKI ne soit pas sujette à diverses attaques.

Les composants d'une PKI

Une PKI contient plusieurs composants principaux essentiels à son bon fonctionnement :

- **Autorité d'enregistrement**

(Registration Authority - RA) Son principal rôle est de vérifier la demande d'enregistrement (Certificate Signing Request - CSR) d'un nouvel utilisateur dans l'infrastructure. Les méthodes de vérification de cette étape sont définies en fonction de la politique de certification choisie pour l'infrastructure. En effet, cela peut aller du simple échange de courriel pour valider la demande ou à une vérification de l'identité de la personne (carte d'identité, passeport, etc.). Si l'autorité d'enregistrement valide la demande d'enregistrement, alors la requête de certificat passera entre les mains de l'autorité de certification (cette requête est transmise au format standard PKCS#10). L'autorité d'enregistrement peut

être contenue dans l'autorité de certification. Cependant, l'avantage de séparer ces deux entités de la répartition de charge de chaque entité.

- **Autorité de Certification**

(Certificate Authority - CA) Son principal rôle est de générer un certificat pour l'utilisateur. Le certificat contiendra des informations personnelles sur l'utilisateur mais surtout sa clef publique et la date de validité. L'autorité de certification signera ce certificat avec sa clef privée, ainsi ce certificat sera certifié authentique par lui-même. C'est pourquoi on parle de chaîne de confiance dans une PKI car il s'agit de faire confiance à cette autorité de certification qui sera lui-même certifié par une autorité supérieure et ainsi de suite. L'autorité de certification aura aussi le rôle de mettre à jour la liste des certificats qu'il a signé afin de connaître les dates de validité de ses certificats. En effet, pour vérifier si un certificat est valide, il faudra demander à l'autorité de certification qu'il l'a généré si le certificat en question est toujours valide ou s'il a été révoqué.

- **Stockage des certificats dans un Annuaire**

L'annuaire est indépendant de la PKI cependant la PKI en a besoin. Les seules contraintes de l'annuaire sont qu'il doit accepter le protocole X.509 pour le stockage des certificats révoqués et le protocole LDAP. Son rôle est comme dit précédemment de stocker les certificats révoqués et par la même occasion, les certificats en cours de validité afin d'avoir un accès rapide à ces certificats. De plus, l'annuaire peut stocker les clefs privées des utilisateurs dans le cadre du recouvrement de clef. Sachant que les certificats sont largement distribués, l'annuaire est une solution pour les mettre à disposition.

Dans le cadre des révocations de certificat, il subsiste un problème. En effet, les autorités de certification doivent constamment mettre à jour leurs listes de certificat en cours de validité et les utilisateurs doivent constamment interroger l'autorité de certification pour vérifier la validité d'un certificat. Si les demandes ne sont pas synchronisées, alors un certificat révoqué pourra être considéré comme valide. Pour contrer ce problème, les utilisateurs doivent disposer de la liste de révocation en temps réel, en vérifiant ces informations directement dans la base de données de l'autorité de certification. Cette vérification est possible par l'intermédiaire d'un élément OCSP (Online Certificate Status Protocol) qui se

chargera d'interroger l'autorité de certification sur la validité d'un certificat. Autrement dit, OCSP est considéré comme un répondeur afin de connaître les listes de révocation. C'est pourquoi, la liste de révocation de la PKI est le seul élément devant disposer d'un service d'annuaire obligatoirement connecté à Internet.

Un exemple d'utilisation de la PKI (carte à puce)

L'utilisateur demande à l'autorité d'enregistrement, une demande de certificat. Lorsque l'autorité d'enregistrement reçoit la demande de l'utilisateur, il vérifie l'identité de l'utilisateur et valide sa demande s'il est apte à recevoir un certificat. Par la suite, il passe cette demande à l'autorité de certification qui va appliquer les procédures. Une fois que ces procédures sont faites, l'autorité de certification va générer une paire de clef. Dans notre exemple, le support du certificat est une carte à puce (d'autre cas d'utilisations comme une clef USB, la biométrie, etc. sont possibles). L'autorité de certification va signer un certificat contenant la clef publique de l'utilisateur avec sa clef privée. Par la suite, le certificat et la clef privée de l'utilisateur seront insérés dans la carte à puce. De plus, le certificat sera ajouté dans l'annuaire étant été signé par l'autorité de certification (la clef privée pourrait aussi être ajoutée dans l'annuaire dans le cadre du recouvrement de clef). La carte à puce sera remise à l'utilisateur.

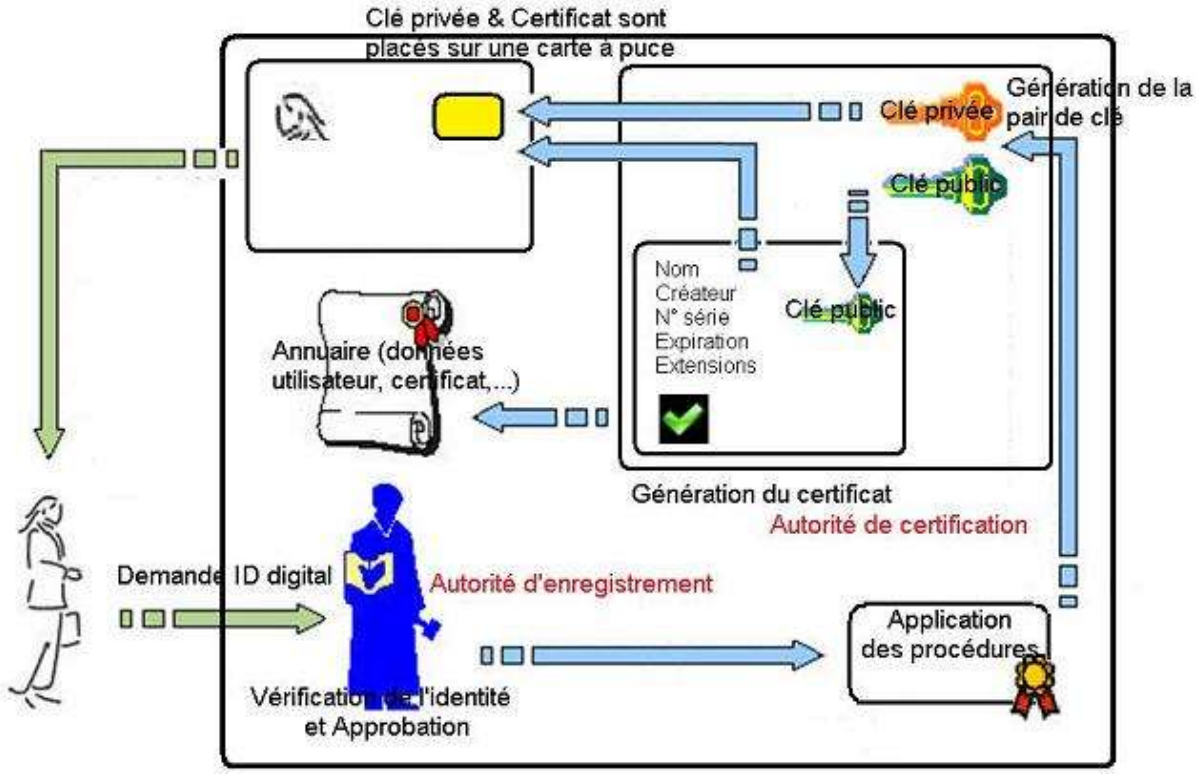


Figure I. 17 - Demande de certificat

3.2.8. Authentification à base de centre de distribution de clé «KDC»

En cryptographie, un centre de distribution de clés (KDC) fait partie d'un système cryptographique destiné à réduire les risques inhérents à l'échange de clés. KDC fonctionnent souvent dans des systèmes dans lesquels certains utilisateurs peuvent avoir l'autorisation d'utiliser certains services à certains moments et pas à d'autres.

Une opération typique avec un KDC implique une demande d'un utilisateur souhaite utiliser un certain service. Ce KDC va utiliser des techniques cryptographiques pour authentifier les utilisateurs. Il sera également vérifié si un utilisateur individuel a le droit d'accéder au service demandé. Si l'utilisateur authentifié remplit toutes les conditions prescrites, le KDC peut délivrer un ticket d'accès.

- KDC fonctionnent principalement avec le chiffrement symétrique ;
- Dans la plupart des cas (mais pas toujours), le KDC partage une clé avec chaque parti du protocole ;
- Le KDC produit un ticket basé sur une clé d'un serveur ;
- Le client reçoit le ticket et le soumet au serveur approprié ;

- Le serveur peut vérifier le ticket présenté et accorder l'accès à l'utilisateur.

Il existe plusieurs crypto systèmes basés sur les KDC nous citons le plus populaire (Kerberos).

3.2.8.1. Kerberos version 4

Le protocole Kerberos 4 est largement basé sur le protocole de Needham-Schroeder [I.3], avec deux changements majeurs.

Le premier changement au protocole de Needham-Schroeder est de réduire le nombre de messages envoyés entre le client et le serveur d'authentification. Le protocole Needham-Schroeder ne dépend pas d'une source de temps sur le réseau, mais le coût était un supplément de deux échanges de messages.

La seconde, plus significative, le changement de protocole de base crée le concept d'un ticket d'octroi de ticket « Ticket Granting Ticket » ; ce qui permet aux utilisateurs de s'authentifier auprès de plusieurs serveurs d'applications en entrant leurs secret d'authentification qu'une seule fois. Si l'original du protocole de Needham-Schroeder a été mis en œuvre tel quel, l'utilisateur devra entrer son mot de passe à chaque fois qu'il souhaite se connecter à un serveur d'applications. L'un des objectifs majeurs de la conception de Kerberos était de créer un système d'authentification unique « SSO » dans laquelle les utilisateurs doivent entrer leurs informations d'identification une fois par jour, et toutes les demandes d'authentification futures sont gérées de manière transparente, sans intervention de l'utilisateur.

De ce fait, le protocole Kerberos 4 est divisé en deux composants logiques : le serveur d'authentification (AS) et le serveur d'octroi de tickets (TGS). Notez qu'il y a un affrontement malheureux dans la terminologie ici. Le serveur d'authentification Kerberos ne doit pas être confondu avec le serveur d'authentification de Needham-Schroeder, le premier effectue une partie de services de ce dernier, comme nous le verrons dans un instant. Bien que ces composants soient généralement mis en œuvre dans un seul programme qui s'exécute sur le KDC, ils sont logiquement des processus séparés.

Encore d'autres changements reflètent les réalités de la sécurité des réseaux informatiques d'aujourd'hui. Le protocole original de Needham-Schroeder suppose que tous les secrets concernées, y compris la clé de l'utilisateur, la clé du serveur d'application et la clé de session générée aléatoirement par le KDC,

sont toujours tenues secrètes. En réalité, les machines sont compromises et que les gens donnent leurs mots de passe. En outre, la capacité de SSO prévu dans Kerberos 4 signifie que les postes de travail des utilisateurs auront les informations d'identification «credentials» mises en cache, si elles sont laissées sans surveillance, elles peuvent être utilisées par un attaquant pour usurper l'identité d'un utilisateur.

Par conséquent, Kerberos 4 introduit une durée de vie limitée pour les informations d'identification. Sans expiration du ticket, un utilisateur peut s'authentifier une seule fois, et ne jamais s'authentifier à nouveau (à condition que leurs credentials ne soient jamais supprimés du poste de travail). Ces durées de vie assurent que les utilisateurs doivent vérifier leur identité périodiquement, disons une fois par jour en entrant leur mot de passe. Elles ont aussi fermé la fenêtre de vulnérabilité dans le cas où les informations d'identification seront volées.

Serveur d'authentification et serveur d'octroi de ticket

Le serveur d'authentification effectue une fonction : recevoir une requête d'authentification contenant le nom d'utilisateur du client demandeur et retourner un TGT crypté pour cet utilisateur.

Ensuite, le client peut utiliser ce (TGT) pour demander des tickets supplémentaires pour d'autres services.

Le premier message du client envoyé au KDC Kerberos est le message de demande d'authentification du serveur, également connu comme un message AS_REQ.

Ce message est envoyé en clair contenant l'identité du client, l'heure locale du client, et le nom principal du TGS du serveur.

Une fois que le KDC reçoit le message AS_REQ, il vérifie que le principal demandeur existe, et que timestamp du client est proche de l'heure locale du KDC (en général, cela signifie moins de cinq minutes). Cette vérification n'est pas faite pour détecter les Reply, après tout, le message AS_REQ est envoyé entièrement en texte clair. Au lieu de cela, il effectue cette vérification de telle sorte que les clients peuvent fournir aux utilisateurs un message tôt dans le processus d'authentification dans le cas d'un décalage de temps entre le client et le KDC. Si une de ces vérifications échoue, un message d'erreur est envoyé au client et donc le client n'est pas authentifié.

Ensuite, le serveur d'authentification génère une clé de session aléatoire. Cette clé de session sera partagée entre le client et le TGS. Elle sécurise les demandes de tickets que le client apporte au TGS tard pour les services Kerberos spécifiques. Le KDC fait deux copies de cette clé de session : une pour le client, et l'autre pour le TGS.

Le KDC répond avec un message (Authentication Server Reply : AS_REP). Ce message contient les informations ci-dessus, une copie de la clé de session, la vérification de l'identité du client chiffré avec la clé du TGS, une autre copie de la clé de session chiffrée avec la clé de l'utilisateur. Tant que le client a connaissance de la clé de l'utilisateur, alors le client peut déchiffrer le message et acquérir la clé de session d'aujourd'hui partagée entre le client et le TGS.

La connaissance de cette clé de session et la possession du TGT (l'autre copie de la clé de session, chiffrée avec la clé du TGS) permettent au client d'obtenir des tickets pour les services Kerberos supplémentaires sans obliger l'utilisateur à entrer de nouveau leur mot de passe.

Ceci implique que la sécurité du système Kerberos est fortement tributaire des mots de passe que les utilisateurs choisissent. Le seul moyen de défense contre les attaques est un mot de passe fort. Les échanges AS_REQ et AS_REP sont résumés dans la (figure I.18).

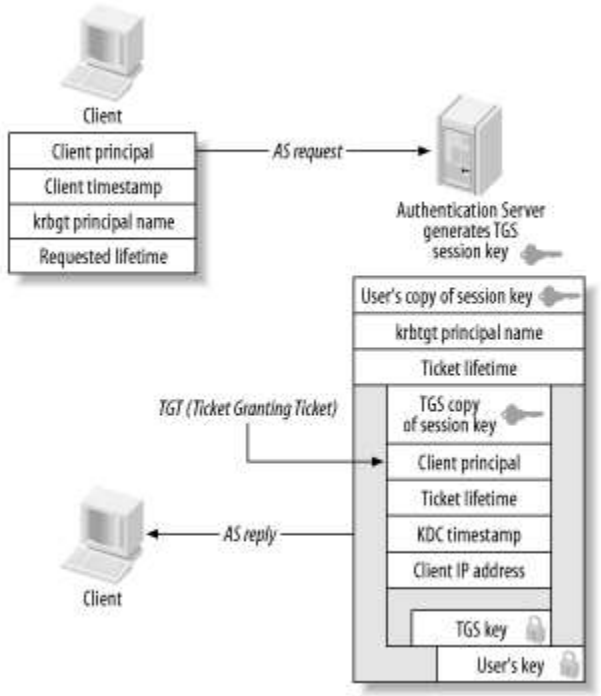


Figure I. 18 - Serveur d'authentification, l'échange requête-réponse

Une fois la transaction du serveur d'authentification terminée, le client dispose d'une clé de session chiffrée par la clé de l'utilisateur, ainsi que d'un TGT, d'abord chiffré avec la clé du TGS. Le client tente de déchiffrer le message par la clé de l'utilisateur (mot de passe). Si ce déchiffrement est réussi, alors le mot de passe est correct et le client enregistre le TGT ainsi que sa copie de la clé de session dans le credential. Rappelez-vous que même si le client a retiré la couche externe de cryptage, le TGT est toujours crypté avec la clé du TGS. Par conséquent, le client ne peut pas lire son contenu ; mais il le stocke simplement dans le credential.

Lorsque l'utilisateur souhaite s'authentifier auprès d'un service de Kerberos plus tard, il y a un protocole d'échange distinct avec le TGS. Le client prépare un message au serveur d'octroi de tickets constitué de trois parties : une demande TGS, une copie du ticket d'octroi de tickets acquis plus tôt, et un authentificateur «authenticator» qui sert à contrecarrer les attaques par rejeu.

L'authentificateur est constitué d'un timestamp chiffré avec la clé de session acquise de l'échange avec le serveur d'authentification. L'authentificateur garantit que chaque paquet de demande de ticket est unique et prouve aussi que le client à connaissance de la clé de session partagée mise en place lors de l'échange du serveur d'authentification. Sans l'authentificateur, un attaquant pourrait se contenter d'écouter sur le réseau pour un ticket envoyé par le KDC pour un client légitime, faire une copie du ticket et de la rejouer.

Lors de la réception de la demande de ticket du client, le KDC formule un message de réponse qui inclut un nouvel ensemble de clés de session, partagées entre le client et le serveur d'applications. La nouvelle copie du client est chiffrée avec l'ancienne clé de session, cela garantit que seul le client valide peut lire la nouvelle clé de session pour une utilisation avec le serveur d'application. L'autre copie du service est incorporée à l'intérieur d'un ticket de service, chiffré par la clé du service. Les clients ajoutent ce ticket de service et la nouvelle clé de session à leurs credentials pour une récupération ultérieure. Lorsque le client contacte le serveur d'application, la connaissance de ce ticket de service et de la clé de session révélera au service que le client est authentique. (Figure I.12) illustre les messages de protocole impliqués lors des échanges avec le TGS.

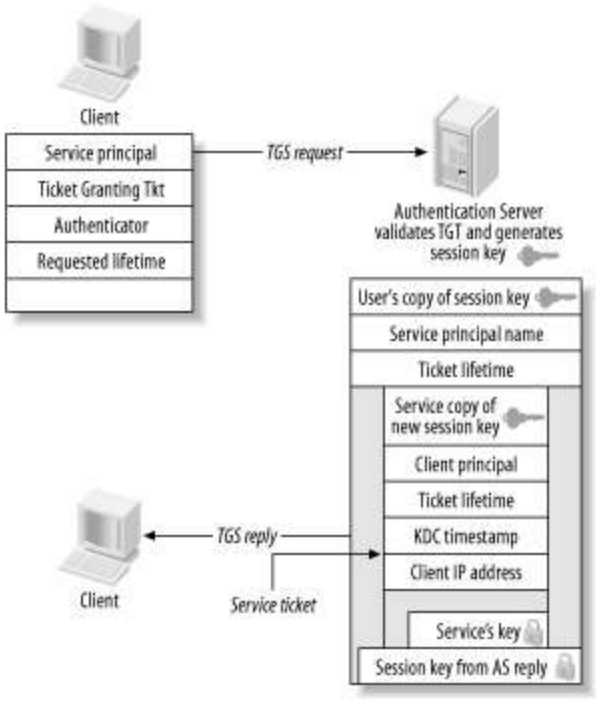


Figure I. 19 - TGS, messages échangés

Les trois principaux messages cryptographiques échangés avec l'AS et le TGS sont : le ticket de service, le ticket d'octroi de ticket et l'authentificateur. La (figure I.20) illustre le contenu de ces messages chiffrés.

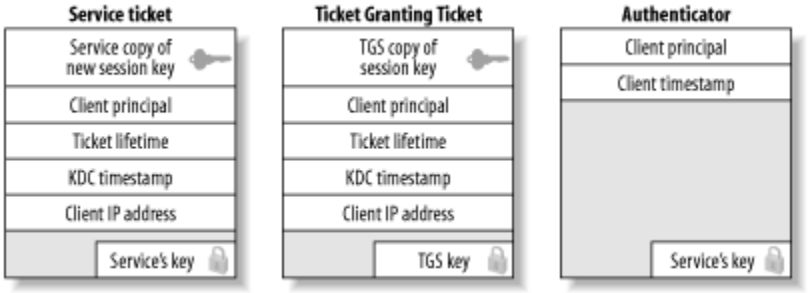


Figure I. 20 - Ticket de service, ticket d'octroi de ticket et l'authentificateur

Notez que dans le système Kerberos, le KDC ne précise pas la politique sur ce que les utilisateurs sont autorisés à accéder à un service donné. Le KDC après s'être assuré de la validité du ticket d'octroi de tickets envoyés par le client émet un ticket pour un service qu'il connaît. Kerberos laisse les décisions d'autorisation aux serveurs d'applications individuels. La possession d'un ticket de service valable pour un service donné ne signifie pas que l'utilisateur aura accès à ce service.

Les détails de la dernière étape d'authentification comment le client envoie le ticket de service à un serveur d'application sont différents pour chaque serveur

d'application. Kerberos ne définit pas une norme pour cela, au contraire, c'est à chaque application de définir le procédé permettant au client d'envoyer son ticket de service et un authentificateur pour l'authentification. Alors que Kerberos fournit des bibliothèques de fonctions qui peuvent formuler et lire des messages pour effectuer ces tâches, la transmission et la réception des messages est totalement dépendante de l'application.

Transformation String-to-Key

Kerberos 4 nécessite une transformation entre les mots de passe textuels dont les gens se souviennent et la clé DES de 56 bits qui est effectivement utilisée pour le chiffrement et le déchiffrement des messages transmis dans les deux sens entre le client, le KDC et serveur d'application.

Cette transformation est appelée fonction string-to-key, habituellement raccourcie simplement à string2key. La transformation est très similaire aux fonctions utilisées pour «chiffrer» les mots de passe dans la norme Unix.

Il s'agit d'une fonction de hachage à sens unique qui, dans le cas de Kerberos 4, utilise le mot de passe comme entrée principale, et délivre une clé DES de 56-bit au format hexadécimal.

Toutefois, étant donné que cet algorithme est public (il doit être largement connue pour que les différentes implémentations de Kerberos soient interopérables), une attaque par force brute ou dictionnaire peut être utilisé pour essayer de trouver les mots de passe d'une clé DES correspondant pour un message particulier.

3.2.8.2. Quoi de neuf dans la version 5 ?

Si vous regardez strictement l'ensemble des fonctionnalités, Kerberos 5 est une évolution de Kerberos 4. Le protocole Kerberos 5 contient toutes les fonctionnalités présentes dans le protocole Kerberos 4, mais avec de nombreuses extensions.

Nous allons examiner quelques nouvelles fonctionnalités présentes dans Kerberos 5 ainsi que la nouvelle infrastructure fournie par le protocole. Il y avait deux couches de cryptage dans Kerberos 4 soit dans un AS_REP ou TGS_REP. Le ticket chiffré avec la clé du service, est contenue dans un message de réponse chiffré avec la clé de l'utilisateur.

Ce double cryptage s'avère non nécessaire, afin que Kerberos 5 concatène les deux messages chiffrés les uns après les autres au lieu de l'imbrication du ticket à l'intérieur du message de réponse. Cette étape améliore l'efficacité et la performance, et ne réduit pas la sécurité des messages.

La phase de la pré-authentification

L'original du protocole Kerberos 4 est vulnérable aux attaques par dictionnaire et force brute. Cette vulnérabilité provient du fait que le KDC délivre une réponse chiffrée à n'importe quel client (étant donné que le client existe dans la base de données Kerberos). Du moment que le KDC fournit une réponse à tout demandeur, une attaque hors connexion peut être effectuée afin de découvrir la clé secrète du client.

Pour rendre cette attaque plus difficile, Kerberos 5 introduit la pré-authentification (voir la figure I.21). La pré-authentification requiert que les demandeurs doivent prouver leurs identités avant que le KDC ne leur délivre une réponse particulière.

Si un utilisateur tente d'acquérir des tickets initiaux à travers l'échange d'AS, mais le KDC nécessite la pré-authentification, le KDC renvoie un message KRB_ERROR au lieu d'un AS_REP en répondant à la demande du client. Ce message KRB_ERROR indique au client que la pré-authentification est requise. Le client génère les données requises de la pré-authentification et les renvoie. Si les données de la pré-authentification sont acceptées par le KDC, il répond par un message AS_REP qui inclut le ticket initial du client. Dans le cas contraire, le KDC renvoie un autre message KRB_ERROR qui indique que la pré-authentification a échoué et le client ne recevra pas de ticket initial.

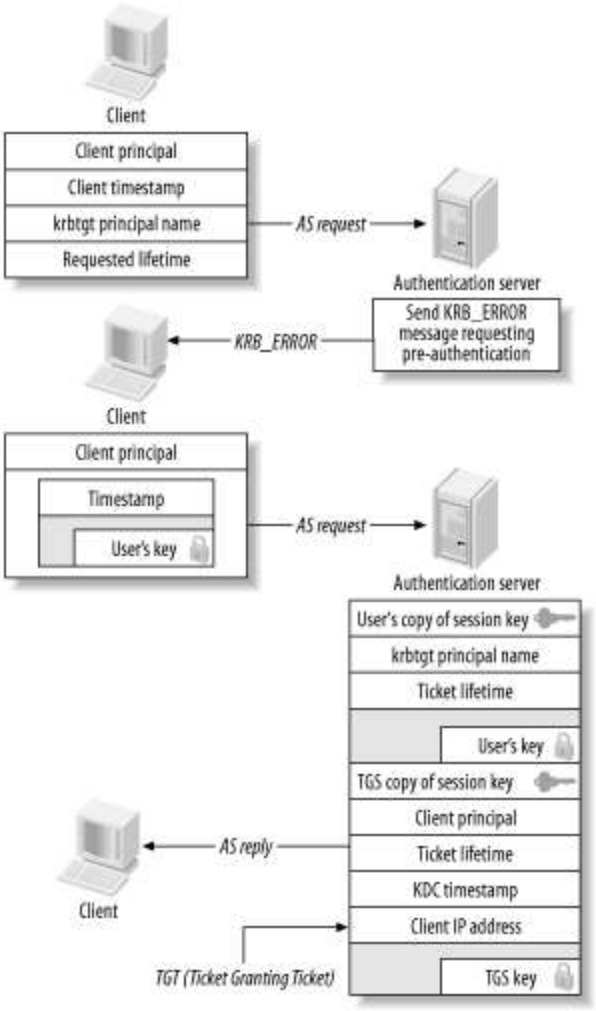


Figure I. 21 - Protocole Kerberos V5

3.2.8.3. L'authentification initiale (PKINIT) du protocole Kerberos

Une situation où la cryptographie à clé publique peut bénéficier du protocole Kerberos dans l'échange initial avec le serveur d'authentification. Au lieu de partager une clé secrète entre le client et le KDC, le client possède une paire de clé signé par une autorité de certification, qui est utilisée pour s'authentifier sur le domaine.

Le système de tiers de confiance utilisé par Kerberos est impliqué seulement lors d'une signature d'une clé publique associée à un utilisateur. Le reste du temps, l'autorité de certification fonctionne offline. Donc, elle n'a pas besoin d'être impliquée à chaque fois qu'une demande d'authentification est effectuée.

L'utilisation typique de PKINIT est de fournir à chaque utilisateur dans le domaine Kerberos un certificat signé. Chaque utilisateur sur son poste de travail génère

une paire de clés publique/privée et la clé publique est décernée à une autorité de certification pour la validation. Une fois que la clé est signée, le certificat est renvoyé à l'utilisateur et stocké sur son support.

Le seul changement au protocole Kerberos est l'échange initial des messages entre le client et le serveur d'authentification du KDC. Dans le protocole traditionnel à clé symétrique de Kerberos, le client demande un TGT au KDC, et le KDC répond avec un ticket chiffré avec la clé secrète de l'utilisateur. Si le domaine nécessite la pré-authentification, l'utilisateur doit fournir un timestamp chiffré avec le mot de passe secret de l'utilisateur (ou un autre authentificateur) au KDC, cela donne une preuve raisonnable que le client est authentique.

PKINIT s'étend du protocole Kerberos 5 en utilisant le champ de la pré-authentification pour identifier son certificat. Quand un utilisateur souhaite s'authentifier auprès d'un KDC qui prend en charge PKINIT, l'utilisateur place son certificat dans le champ de la pré-authentification de la requête AS. Le KDC reçoit la demande et renvoie le ticket validé crypté avec la clé publique contenue dans le certificat de l'utilisateur.

Une fois l'échange initial AS requête/réponse fini, les autres échanges du protocole Kerberos restent inchangées. Le TGT contient une clé de session symétrique conventionnelle qui peut ensuite être utilisée pour obtenir des tickets pour d'autres ressources Kerberos.

3.2.9. Authentification forte par la combinaison de plusieurs facteurs d'authentification

Il existe plusieurs façons pour authentifier une personne (ou plus généralement, une entité) ; on peut se baser sur :

« Ce qu'il sait » est tout simplement un mot de passe qui va permettre, par comparaison ou par décryptage, par exemple, de vérifier que la personne est bien celle qu'elle prétend être.

« Ce qu'il a » est une alternative moins fréquente mais plus sûre car elle fait appel à un objet que la personne cherchant à s'authentifier a en sa possession. Un exemple de ce type d'authentification est la carte à puce.

« Ce qu'il est » est justement une méthode encore peu utilisée pour le moment, et fait appel à des mécanismes de biométrie (empreinte digitale, rétine...). Cette technique n'est pas encore utilisée, entre autres par le manque de logiciels l'utilisant réellement, ainsi que du peu de matériels encore onéreux. L'intérêt de

ce système est qu'il supprime la duplication, le vol, l'oubli et la perte. Cependant, à long terme, cette technique risque de ne pas être exempte de piratage.

Bien sûr, un système d'authentification peut combiner ces facteurs, pour réaliser une authentification forte. [I.43]

L'authentification permet de vérifier l'identité d'un utilisateur sur l'une des bases suivantes (Figure I.22) :

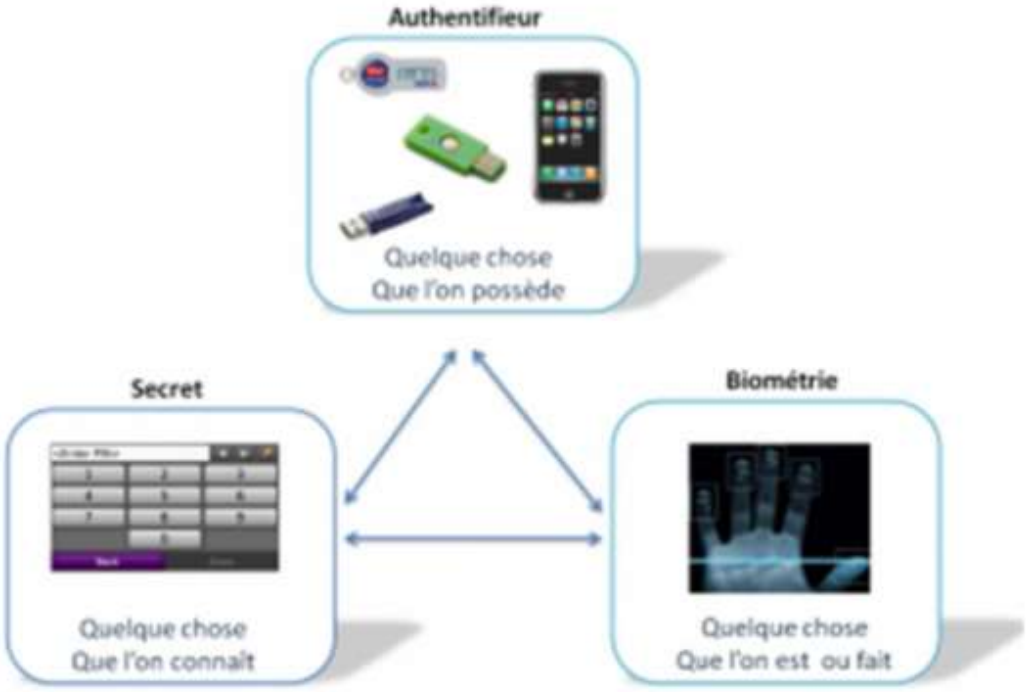


Figure I. 22 - Les facteurs d'authentification

Lorsque plusieurs facteurs d'authentification sont combinés, nous parlons d'authentification forte par opposition à l'authentification simple.

L'authentification forte

Le couple ≪Identifiant/mot de passe≫ est la méthode d'authentification la plus utilisée. Malgré son manque de sécurité (Une étude faite par le groupe NTA en 2002 sur 500 sujets montre qu'il y a approximativement 21 mots de passe par utilisateur, 81% d'entre eux sont communs et 30% d'entre eux sont écris dans des fichiers), sa facilité d'implémentation explique sa notoriété. Afin de pallier les faiblesses de l'authentification unique par mot de passe, de plus en plus d'entreprises se tournent vers l'authentification forte.

On appelle authentification forte, tout système permettant un accès informatique après une vérification multiple. Ça consiste à mixer au moins deux

stratégies d'authentification, de préférence de nature différente, par exemple : une empreinte digitale et un mot de passe. Ces informations sont ensuite mises en relation avec une solution de gestion des identités et des accès :
• Exemples de système d'authentification à 2 facteurs : Carte à puce + code PIN, Carte à puce + biométrie, Biométrie + mot de passe.
• Exemple de système d'authentification à 3 facteurs : Carte à puce + code PIN + biométrie.
En matière d'authentification forte, trois types de solutions sont courantes :

• Le principe de calculette, qui se présente sous la forme d'une petite calculatrice générant un code automatique à durée de vie limitée (principe des OTP : One Time Password).

• La clé USB, capable de stocker des mots de passe cryptés ou de gérer les certificats électroniques d'accès aux documents.

• Et enfin les systèmes de cartes à puce contenant des codes PIN et des clés de chiffrement numérique. Ces derniers offrent une solution plus sécurisée pour le stockage des clés cryptographiques utilisées dans les transactions réseaux.

3.2.10. Authentification mutuelle

L'authentification mutuelle ou l'authentification à deux sens, se réfère à deux partis authentifiant l'un auprès de l'autre en même temps.

En termes de technologie, l'authentification mutuelle se réfère à un client/utilisateur qui s'authentifie auprès d'un serveur et que ce serveur lui-même s'authentifie à son tour auprès de l'utilisateur de manière à ce que les deux parties sont assurés de l'identité de l'autre. L'authentification mutuelle est généralement utilisée uniquement lorsqu'un niveau de sécurité supplémentaire est nécessaire, en particulier dans les transactions financières entre les organisations.

4. Architecture des protocoles

Dans la section (3.1), l'architecture de protocole se compose d'un serveur de confiance en ligne, qui génère la clé de session, et un/deux autre(s) participant(s).

Nous considérons ici des architectures alternatives. Il y a trois caractéristiques que nous considérons comme des critères architecturaux pour classer les différents protocoles : quelles clés sont déjà établies, comment la clé de session

est générée, et le nombre d'utilisateurs du protocole conçu pour servir. Notez que le deuxième critère n'est applicable uniquement aux protocoles concernés par l'établissement de clés.

4.1. Les clés cryptographiques existantes

En principe, il est impossible d'établir une clé de session authentifiée sans canaux sécurisés existants déjà disponibles. En fait, ce principe peut être déclaré officiellement avéré correct [I.5]. Par conséquent, à l'exception de la possibilité de mise en place physique sécurisée des clés, il est essentiel que les clés soient déjà partagées entre les différents participants ou bien que les clés publiques certifiées soient disponibles.

Nous pouvons considérer que l'utilisation des certificats à base de clé publique comme équivalente à l'utilisation d'un serveur hors ligne. Par conséquent, si deux participants souhaitent établir une nouvelle clé de session, il y a essentiellement trois possibilités.

1. Les participants partagent au préalable une clé secrète.

2. Un serveur hors-ligne est utilisé. Cela signifie que les participants possèdent la certification des clés publiques. Afin de vérifier l'authenticité d'une clé publique, il peut être nécessaire de vérifier une chaîne de certificats.

3. Un serveur en ligne est utilisé. Cela signifie que chaque participant partage une clé avec un serveur de confiance. Afin de transmettre des informations entre les parties, il peut être nécessaire de passer par l'intermédiaire d'une chaîne de serveurs en ligne.

4.2. Les méthodes de génération de clés de session

Il y a plusieurs façons qui peuvent être utilisés pour générer des clés de session dans un protocole d'établissement de clé.

Définition I.11 : Un protocole de transport de clé est un protocole d'établissement de clé dans laquelle l'un des participants génère la clé et la transfère à tous les utilisateurs du protocole.

Définition I.13 : Un protocole hybride est un protocole d'établissement de clé dans lequel l'établissement de la clé de session est une fonction de contribution par plus d'un participant, mais pas par tous les utilisateurs. Cela signifie que le protocole est un protocole d'accord de clé pour les uns, et un protocole de transport de clé pour les autres.

Dans la littérature, les protocoles de transport de clés sont généralement associés à des protocoles utilisant un serveur en ligne, alors que ceux d'accord de clé sont associés à des protocoles utilisant un serveur hors ligne. Cependant, cette restriction n'est pas nécessaire et il y a des exemples de protocoles d'accord de clés dans lesquels un serveur en ligne a une contribution à la clé de session, et les protocoles de transport de clés avec un serveur hors ligne.

4.3. Le nombre d'utilisateurs

Un dernier élément de l'architecture de protocole est le nombre de participants qui y participent. La plupart des protocoles d'établissement de clés se sont concentrés sur le cas où deux utilisateurs souhaitent établir une clé de session pour les communications point-à-point.

Les trois critères mentionnés ci-dessus peuvent être utilisés pour classer les protocoles d'établissement de clés ; pour un nombre donné d'utilisateurs les deux premiers critères donnent lieu à (3 X 3 = 9) différentes classes de protocoles.

5. La fraîcheur d'une valeur (Freshness)

Nous avons déjà vu que l'une des exigences des protocoles utilisés pour établir une clé de session, est que chaque utilisateur de la clé doit être en mesure de vérifier que la clé est nouvelle et pas rejoué à partir d'une ancienne session. Cette propriété s'étend à divers autres types de protocoles. Par exemple, des protocoles conçus pour réaliser l'authentification en temps réel doivent également veiller à ce que les messages envoyés ne sont pas rejoués.

Avant d'envisager des mécanismes détaillés nous notons qu'il y a essentiellement deux façons différentes pour que la fraîcheur d'une valeur puisse être garantie à un utilisateur particulier. La première est là où l'utilisateur dispose une partie dans le choix de la valeur (qui sera souvent une nouvelle clé de session), tandis que le second est là où l'utilisateur doit compter sur quelque chose reçue avec une valeur connue comme fraiche.

Voici deux utilisateurs **A** et **B** les deux choisissent une entrée, $\mathbf{N_A}$ et $\mathbf{N_B}$ respectivement, à une nouvelle clé de session $\mathbf{K_{AB}}$. La clé de session sera formée en choisissant une fonction des entrées.

$$\mathbf{K_{AB} = f\ (N_A, N_B)}$$

Une propriété souhaitable de la fonction **f** est qu'il ne devrait pas être possible ni pour **A** ni pour **B** de forcer une ancienne valeur de $\mathbf{K_{AB}}$ même si l'entrée de l'autre est connue. Cela signifie que chaque utilisateur dispose une assurance indépendante de la fraîcheur de $\mathbf{K_{AB}}$. Cette propriété est obtenue par **A** si, une fois que le $\mathbf{N_A}$ est choisi, **B** ne peut pas choisir $\mathbf{N_B}$ d'une manière telle que $\mathbf{K_{AB}}$ soit une ancienne valeur.

Passons maintenant au deuxième cas, où la fraîcheur dépend de quelque chose reçu avec le message. Supposons qu'un participant **A** souhaite vérifier la fraîcheur d'une clé de session $\mathbf{K_{AB}}$ qui a été généré par certains **S** (qui peut être un serveur ou le participant B qui partage $\mathbf{K_{AB}}$). A doit faire confiance à S de telle sorte que la clé $\mathbf{K_{AB}}$ doit être fraîchement générée, mais il doit être sûr que le message reçu n'est pas un ancien message qui a été rejoué par un adversaire.

Supposons que A reçoit le message **F ($\mathbf{K_{AB}}$, N)**, qui est une fonction de K_{AB} et une valeur de fraîcheur **N**. Quelles sont les propriétés requises de **F** afin de permettre au destinataire d'être sûr que le message composé est frais ? **F** doit fournir l'authentification de l'origine des données et l'intégrité des données afin que le destinataire puisse en déduire que **F ($\mathbf{K_{AB}}$, N)** été généré par **S** et n'a pas été modifié. Si A peut être sûr que N est fraîche donc il peut aussi être sûr que **F ($\mathbf{K_{AB}}$, N)** est frais. Comme **S** est digne de confiance pour générer et authentifier que de nouvelles clés, **A** peut donc en déduire que $\mathbf{K_{AB}}$ est fraîche. Nous examinerons ensuite les différentes formes que la valeur **N** peut prendre.

Une valeur fraiche veut tout simplement dire que celle-ci n'a pas été utilisée auparavant. Il existe trois types de valeur de fraîcheur utilisés : horodatages (timestamp), nonces et les compteurs. Gong [I.6] a classé les différentes façons que ces types peuvent être utilisés dans un protocole.

5.1. Le mécanisme d'horodatage (Timestamps)

L'expéditeur du message ajoute l'heure actuelle au message lors de l'envoi. Ceci est contrôlé par le destinataire lors de la réception du message en faisant la comparaison avec l'heure locale. Si l'horodatage est reçu dans un intervalle acceptable, le message est considéré comme frais.

La difficulté d'utiliser horodatages est que les horloges synchronisées sont nécessaires et doivent être maintenus en toute sécurité. Gong [I.7] a fait remarquer que si l'horloge d'un participant est avancée au-delà du temps dans le reste du système, une vulnérabilité peut exister même après que l'horloge soit

corrigée. En effet, un adversaire aurait pu capturer et supprimer un message qui deviendra frais dans l'avenir (suppress relay attack).

5.2. Les défis aléatoires (Nonces)

Le destinataire **A** du message génère un nonce $\mathbf{N_A}$, et il l'envoie à l'expéditeur du message **B**. Le nonce $\mathbf{N_A}$ est ensuite retourné avec le message après le traitement avec une fonction cryptographique **f** comme indiqué dans le protocole I.2.

1. A → B : N_A

2. B → A : f (N_A, …)

Protocole I.2 - Utilisation d'un nonce (challenge aléatoire)

A vérifie le nonce à la réception et en déduit que le message est frais parce que le message ne peut pas être formé avant la génération du nonce. Un inconvénient d'utilisation de défi est la nécessité d'augmenter le nombre de messages échangés.

Une attention particulière doit également être portée à la qualité des nombres aléatoires produits, car si le nonce qui est utilisé est prévisible une réponse valide peut être obtenue à l'avance et plus tard rejoué (preplay attack).

5.3. Les compteurs synchrones

L'expéditeur et le destinataire maintiennent un compteur synchrone dont sa valeur est envoyée avec le message, puis sera incrémenté. Un inconvénient des compteurs est que les informations d'état doivent être maintenues pour chaque potentiel de communication. La gestion des compteurs peuvent également causer des problèmes en présence d'erreurs de canal. Gong [I.6] a souligné que si un compteur n'est pas synchronisé avec le récepteur, les attaques de relecture sont possibles.

Mitchell [I.8] a proposé un hybride entre les compteurs et les horodatages. Son idée est d'utiliser un compteur en fonction du temps réel.

6. Types d'attaque sur les protocoles

Nous avons déjà rencontré un certain nombre d'attaques. Le but de cette section est de les résumer, avec beaucoup d'autres qui ne sont pas rencontrées.

Tout d'abord, pour de nombreux protocoles la liste ne sera pas complète. En effet, on pourrait affirmer il n'est pas considérablement utile de savoir qu'un

protocole ne soit pas vulnérable à une certaine liste de menaces ; ce qui est vraiment nécessaire est la confiance qu'il respecte ses objectifs de sécurité suivant une liste connue d'hypothèses.

D'autre part, différents protocoles ont de différents objectifs. Dans nos exemples ci-dessus l'objectif visé était de dériver en toute sécurité une nouvelle clé de session. Cependant, certains protocoles peuvent avoir aucun intérêt pour transmettre la confidentialité, mais uniquement l'authentification en temps réel. D'autres peuvent avoir des objectifs supplémentaires, tels que la confirmation de la connaissance de la clé de session.

La question difficile pour définir des objectifs appropriés pour un protocole cryptographique est le sujet de la section I.7 et nous ne la traitera pas ici. Naturellement, si un protocole atteint ou pas des objectifs particuliers dépend de quelles attaques considérées comme possible.

Considérons maintenant les menaces souvent rencontrés dans les protocoles cryptographiques. Le tableau I.2 liste et définit les attaques connues. Il convient de noter qu'il existe d'autres façons de classer les attaques ; une liste alternative, avec des exemples, est donnée par Carlsen [I.9].

Tableau I. 2 - Les types d'attaques

EAVESDROPPING	L'adversaire capture les informations envoyées dans le protocole.
MODIFICATION	L'adversaire modifie les informations envoyées dans le protocole.
REPLAY	Les enregistrements capturés par un adversaire peuvent être ré-envoyés ultérieurement.
PREPLAY	Une réponse valide peut être obtenue à l'avance par un adversaire et plus tard rejouée.
REFLECTION	L'adversaire envoie en retour des messages de protocole au participant qui les a envoyés.
DENIAL OF SERVICE	L'adversaire empêche ou entrave les participants légitimes de compléter l'exécution du protocole.
TYPING ATTACKS	L'adversaire remplace un message de protocole (normalement encrypté) d'un premier type par un message d'un autre type.
CRYPTANALYSIS	L'adversaire gagne une certaine influence utile du protocole pour lui aider de la cryptanalyse.
CERTIFICATE MANIPULATION	L'adversaire choisit ou modifie les informations de certificat pour attaquer une ou plusieurs exécutions du protocole.
PROTOCOL INTERACTION	L'adversaire choisit un nouveau protocole pour interagir avec un protocole connu.

6.1. L'écoute clandestine (Eavesdropping)

Eavesdropping est peut-être l'attaque la plus élémentaire sur un protocole. Presque tous les protocoles traitent l'écoute en utilisant le cryptage. Il est évident que le chiffrement doit être utilisé pour protéger les informations confidentielles telles que les clés de session. Dans certains protocoles, il peut y avoir d'autres informations qui doivent également être protégées. Un exemple intéressant est que les protocoles pour l'établissement de clé dans les communications mobiles exigent généralement que l'identité de la station mobile reste confidentielle. Eavesdropping est parfois distingué comme étant une attaque passive car il n'impose pas à l'adversaire de perturber les communications des participants légitimes. Les autres attaques que nous considérons toutes impose à l'adversaire d'être actif.

6.2. La modification des messages

Si un message de protocole n'est pas redondant alors la modification de celui-ci est une attaque potentielle. L'utilisation de mécanismes d'intégrité cryptographiques est donc omniprésente dans les protocoles d'authentification et d'établissement de clés.

Tous les messages sont vulnérables à la modification. De nombreuses attaques ne modifient pas n'importe quel message connu, mais divisent et rassemblent différents messages. Cela signifie que les mesures d'intégrité doivent couvrir toutes les parties du message qui doivent être maintenu ensemble ; le chiffrement de ces champs ne suffit pas. Exemples d'attaques sur les protocoles dans lesquels le cryptage ne fournit pas les propriétés d'intégrité requises sont données par Stubblebine et Grigor [I.10] et par Mao et Boyd [I.11].

6.3. L'attaque par rejeu (Replay)

Les attaques par rejeu comprennent toute situation où l'adversaire interfère avec un protocole par l'insertion d'un message, ou une partie d'un message, qui a été envoyé précédemment. Nous pouvons considérer la réponse comme un autre type fondamental d'attaque qui est souvent utilisé en combinaison avec d'autres attaques. En variante, le rejeu peut provenir d'un protocole qui se déroule en même temps que l'attaque. Syverson [I.12] a produit une taxonomie des attaques de relecture sur la base de cette distinction.

6.4. L'attaque par pré-jeu (Preplay)

Preplay pourrait être considéré comme une extension naturelle de rejeu. Un exemple intéressant d'une attaque qui emploie des personnes est l'attaque que l'on appelle (triangle attack) de Burmester [I.13].

6.5. L'attaque par réflexion (Reflection)

La réflexion est vraiment un cas particulier important de rejeu. Un scénario typique est celui où deux participants s'engagent à un protocole de clé partagée et l'un retourne un défi qui été destiné à lui-même. Cette attaque ne peut être possible que si l'exécution parallèle du même protocole sont autorisés, mais ce qui est souvent une hypothèse réaliste. Par exemple, si un participant est un hôte Internet, il peut accepter des sessions de plusieurs autres participants tout en utilisant la même identité et un ensemble de clés cryptographiques.

Hypothèse de sécurité 5 : L'adversaire peut initier un certain nombre d'exécutions parallèles du protocole entre tous les participants, y compris différentes exécutions impliquant les mêmes participants et avec d'autres participants qui prennent les mêmes ou différents rôles de protocole.

Considérez Protocole I.3, ce qui donne un exemple très basique. Supposons que **A** et **B** partagent déjà une clé secrète **K** et choisissez des nonces respectifs $\mathbf{N_A}$ et $\mathbf{N_B}$ pour une utilisation dans le protocole. Le protocole est destiné à authentifier mutuellement les deux parties en démontrant la connaissance de **K**.

1. A → B : $\{N_A\}_K$
2. B → A : $\{N_B\}_K$, N_A
3. A → B : N_B

Protocole I.3 - Un protocole vulnérable à l'attaque de réflexion

A la réception de messages 2, **A** déduit que celui-ci devait être envoyé par **B** depuis **B** a la clé **K**. Cependant, si **A** est prêt à participer à l'exécution du protocole parallèle alors il y a une autre possibilité, à savoir que le message 2 a été initialement formé par **A**. Un adversaire **C** peut compléter avec succès deux exécutions du protocole, comme indiqué dans l'attaque I.1.

1. A → C :$\{N_A\}_K$
I'. C → A : $\{N_A\}_K$

2'. A → C : $\{N'_A\}_K$, N_A

2. C → A : $\{N'_A\}_K$, N_A

3. A → C : N'_A

3'. C → A : N'_A

Attaque I.1 – Attaque par réflexion sur le protocole I.3

Immédiatement après avoir reçu le premier message, **C** commence une autre exécution du protocole avec **A**, et reflète de nouveau le message reçu de **A**. La réponse permet à **C** de répondre au premier message, puis les deux exécutions du protocole peuvent être complétées. En fait tout le traitement cryptographique a été effectuée par **A**, tandis que A croit que deux exécutions de protocole ont été accomplies avec succès. Ces sortes d'attaques sont parfois appelés «attaques d'oracle» puisque **A** agit comme un oracle pour **C** en lui offrant le décryptage nécessaire. Un traitement poussé des attaques par réflexion est donnée par Bird et al. [I.14].

6.6. Le Déni de service (Denial Of Service)

Dans une attaque par déni de service (souvent contracté pour DoS attaque) l'adversaire empêche les utilisateurs légitimes de terminer l'exécution du protocole. Des attaques de déni de service dans la pratique ont lieu sur des serveurs qui doivent interagir avec de nombreux clients. Les attaques peuvent être divisées en celles qui visent à utiliser les ressources de calcul du serveur (resource depletion attacks) et celles qui visent à épuiser le nombre de connexions autorisées par le serveur (connection depletion attacks).

En principe, il semble qu'il est impossible d'empêcher des attaques par déni de service complètement. Toute tentative pour établir une connexion doit soit donner lieu à l'attribution d'une connexion soit d'utiliser certains calculs pour déduire que la tentative est invalide. Néanmoins, il y a certaines mesures qui peuvent être prises pour réduire l'impact des attaques de déni de service. Certains protocoles sont beaucoup plus vulnérables à ce genre d'attaque que d'autres, il est donc important de ne pas ignorer cette question.

Karn et Simpson ont proposé leurs protocoles Photuris (la plus récente version de 1999 [I.15], mais initialement publié en 1995). L'idée est d'assurer, avant d'investir des ressources importantes, que le client fait une demande unique pour la connexion en utilisant le principe des cookies. Cette technique empêche des attaques par déni de service dans lequel l'adversaire envoie des requêtes de

connexion aléatoire. En changeant le secret du serveur sur une base régulière (généralement toutes les 60 secondes), pour que le même client ne puisse pas faire des demandes de connexion illimitée.

Meadows [I.16] suggère que, pour se protéger contre l'épuisement de connexion chaque message doit être authentifié. Toutefois, pour minimiser le calcul éventuellement perdu, l'authentification faible (par les cookies par exemple) au début du protocole peut augmenter progressivement avec les messages suivants.

Jules et Brainard [I.17] ont proposé un mécanisme qu'ils appellent 'client puzzles' pour former une authentification plus forte que celle fournie par les cookies. Leur idée est que lorsque la charge sur un serveur devient élevée (peut-être à la suite d'un déni de service), le serveur va envoyer un «puzzle» de difficulté de calcul modéré à chaque client qui doit être résolu avant qu'une nouvelle connexion sont établie. Les clients légitimes ne seront que légèrement incommodés par cette demande, mais un adversaire qui essaie de faire des connexions multiples devra résoudre de nombreux puzzles.

6.7. L'attaque par mauvaise interprétation du message (Typing Attacks)

Lorsqu'un protocole est écrit sur la page ses éléments sont clairement distincts. Mais en pratique, un participant qui reçoit un message (chiffrés ou non) voit simplement une chaîne de bits qui doit être interprétée.

Cette attaque exploite cette situation en poussant un destinataire mal interpréter un message en prenant un élément pour un autre. Par exemple, un élément qui a été conçu comme un identifiant d'un participant pourrait être accepté comme une clé. Une telle attaque fonctionne généralement avec le rejeu d'un message précédent.

Un exemple peut être vu dans le protocole bien connu d'Otway et Rees [I.18] représentée dans le protocole I.4. **A** et **B** partagent des clés à long terme $\mathbf{K_{AS}}$ et $\mathbf{K_{BS}}$ respectivement, avec le serveur **S**. **S** génère une nouvelle clé de session $\mathbf{K_{AB}}$ et il la passe à la fois à **A** et à **B**. **M** et $\mathbf{N_A}$ sont deux nonces choisis par **A**, $\mathbf{N_B}$ est un nonce choisi par **B**.

Goal : Key transport

1. $A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

2. $B \rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$

3. $S \rightarrow B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$

4. $B \rightarrow A : M, \{N_A, K_{AB}\}_{K_{AS}}$

Protocole I.4 - protocole d'Otway-Rees

Cette attaque fonctionne en raison de la similarité dans les parties cryptées du premier et dernier messages - elles commencent par le même champ de message et sont cryptés par la même clé. Comme d'habitude pour ce genre d'attaque nous devons faire certaines hypothèses supplémentaires sur l'attaque et son succès. L'attaque dépend de la longueur du champ composite **M**, **A**, **B** qui est la même que celle prévue pour la clé $\mathbf{K_{AB}}$. Cela peut être une hypothèse tout à fait raisonnable ; par exemple, soit **M** de taille de 64 bits, **A** et **B** pourrait être de taille de 32 bits, de sorte que $\mathbf{K_{AB}}$ devrait être d'une longueur de 128 bits, ce qui est un choix populaire de taille de clé symétrique. Avec ces hypothèses, un adversaire **C** est capable d'exécuter l'attaque I.2. Ici, nous introduisons la notation $\mathbf{C_B}$ pour indiquer que l'adversaire **C** est déguisé en tant que participant **B**.

1. $A \rightarrow C_B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

4. $C_B \rightarrow A : \{N_A, M, A, B\}_{K_{AS}}$

Attaque I.2 –Typing attack sur le protocole Otway-Rees

C se déguise en tant que **B** et intercepte le message de **A**. **C** retourne ensuite la partie chiffrée de ce message à **A**, qui est interprété par **A** comme le message '4' du protocole. Avec les hypothèses mentionnées ci-dessus, **A** va accepter le champ composite **M**, **A, B** comme la clé $\mathbf{K_{AB}}$. Bien sûr, **C** connaît les valeurs de **M**, **A** et **B** du message 1, et est en mesure de continuer à se faire passer pour **B** pendant la durée de la session. Ces attaques peuvent être contrées par diverses mesures. Les précautions d'ad-hoc comprennent la modification de l'ordre des éléments de message à chaque fois qu'ils sont utilisés en veillant à ce que chaque clé de chiffrement ne soit utilisée qu'une et une seule fois. Des méthodes plus systématiques doivent inclure un numéro de message authentifié dans chaque message. Naturellement celles-ci engendrent un coût supplémentaire en termes de calcul et de bande passante. Aura [I.19] a examiné des méthodes systématiques pour éviter les attaques par rejeu en fonction de la mauvaise interprétation du message.

6.8. L'attaque par manipulation des certificats (Certificate Manipulation)

Dans les protocoles à clés publiques le certificat d'un participant agit comme une assurance hors ligne à partir d'une autorité de confiance que la clé publique du participant lui appartient vraiment. D'autres participants qui font usage d'un certificat sont confiants que l'autorité a correctement identifié le propriétaire de la clé publique au moment où le certificat a été délivré. Cependant, il n'est pas nécessairement prévu que l'autorité soit pourvue d'éléments de preuve que la clé privée correspondante est effectivement détenu par le participant revendiquant la propriété de la paire de clés. Cela conduit à des attaques potentielles dans lesquelles l'adversaire gagne un certificat d'une clé publique, même s'il ne connaît pas la clé privée correspondante. En choisissant la clé publique comme une fonction d'une clé publique existante, des conséquences indésirables peuvent survenir.

Un exemple d'une attaque de manipulation de certificat est donné par Menezes et al. [I.20] sur un protocole d'accord de clé de Matsumoto et al. [I.21]. Les participants **A** et **B** possèdent les clés publiques $\mathbf{g^a}$ et $\mathbf{g^b}$ respectivement, et les clés privées correspondant **a** et **b**. Ici **g** génère un groupe approprié dans lequel le problème du logarithme discret est difficile.

Chaque clé publique est certifiée et ainsi **A** et **B** possèdent des certificats **Cert(A)** et **Cert(B)**, respectivement, qui contiennent des copies de leurs clés publiques. Un protocole normal procède comme indiqué dans le protocole I.5, où **x** et **y** sont des valeurs aléatoires choisies respectivement par **A** et **B**.

Goal: Key agreement

1. A → B : g^x, Cert(A)

2. B → A : g^y, Cert(B)

Protocole I.5 - Un protocole vulnérable à la manipulation de certificat (Protocole de MTI)

La clé partagée est $\mathbf{K_{AB} = g^{ay + bx}}$, calculée par A comme $\mathbf{(g^y)^a * (g^b)^x}$ et par **B** comme $\mathbf{(g^a)^y * (g^x)^b}$. L'adversaire débute une attaque en choisissant une valeur aléatoire **c**, affirmant que $\mathbf{g^{ac}}$ est sa clé publique, et l'obtention d'un certificat pour cette clé publique (Notez que **C** ne peut pas obtenir ac la clé privée correspondante.) **C** se déguise entant que **B** dans le protocole I.5, et complète deux exécutions du protocole, l'une avec **A** et l'autre avec **B**, comme indiqué dans l'attaque I.3.

1. A → CB : g^x, Cert(A)

I'. C → B : g^x, Cert(C)

2'. B → C : g^y, Cert(B)

2. CB → A : g^{yc}, Cert(B)

Attaque I.3 – Une attaque de manipulation de certificat sur le protocole MTI

Après l'exécution de l'attaque, **A** calculera la clé $\mathbf{K_{AB} = (g^{yc})^a * (g^b)^x = g^{acy+bx}}$ et **B** calculera la clé $\mathbf{K_{CB} = (g^{ac})^y * (g^x)^b = g^{acy + bx}}$. Ainsi **A** et **B** ont trouvé la même clé, mais **A** croit que cette clé est connues seulement par **A** et **B** tandis que **B** croit qu'elle est connu que par **C** et **B**. Ce malentendu peut conduire à des problèmes dans l'utilisation ultérieure de la clé de session.

Les attaques de ce genre peuvent être évitées en exigeant de chaque participant à démontrer une connaissance de la clé privée avant la délivrance du certificat pour une clé publique. Une telle démonstration est idéalement réalisée en utilisant des techniques de preuve à divulgation nulle de connaissance « zero knowledge » afin que l'autorité de confiance ne gagne rien d'utile à trouver la clé privée. Une méthode plus pratique est que le propriétaire de la clé privée signe un message spécifique ou un défi. Plus généralement ce processus fait partie de validation de la clé publique qui fournit l'assurance que la clé publique et la clé privée correspondante ont été correctement produites comme exigé dans le protocole.

6.9. L'attaque par interaction de protocole (Protocol interaction)

La plupart des clés à long terme sont destinées à être utilisées pour un seul protocole. Toutefois, il pourrait y avoir le cas où les clés sont utilisées dans des protocoles multiples. Cela pourrait être dû à une conception négligente, mais peut être délibérée dans les cas où les appareils dotés d'une capacité de stockage limitée sont utilisées pour de multiples applications (les cartes à puce sont l'exemple évident).

Il est bien évident que les protocoles conçus indépendamment peuvent mal interagir. Par exemple, l'exploitation d'un protocole qui utilise le chiffrement pour prouver la possession d'une clé peut être utilisée par un adversaire pour déchiffrer les messages d'un autre protocole de la même clé. Kelsey et al. [I.22] donnent plusieurs exemples de la façon dont les choses peuvent aller mal, et discutent de l'attaque de (chosen protocol attack) dans lequel un nouveau protocole est conçu par l'adversaire afin d'attaquer un protocole existant.

Mis à part la limitation d'utilisation de façon unique les clés dans les protocoles, une méthode pour prévenir de telles attaques consiste à inclure les détails du protocole (comme identificateur unique et un numéro de version) dans une partie authentifiée des messages.

7. Les objectifs d'authentification et d'établissement de clés

Toute attaque sur un protocole n'est valide que si et seulement si elle viole l'une des propriétés que le protocole devait atteindre. En d'autres termes toutes les attaques doivent être considérées par rapport aux objectifs du protocole. L'expérience a montré que de nombreux problèmes de protocole surgissent lorsque les concepteurs ne sont pas clairs sur les objectifs du protocole à d'atteindre.

Gollmann [I.23] a reconnu qu'il est difficile de décider exactement ce que l'on entend par les mots couramment utilisés tels que «authentification»; même si tout le monde a une idée générale de la signification d'un tel mot, l'interprétation peut varier avec le protocole. Il se trouve que, bien que la plupart des auteurs puisse convenir de définitions générales, leurs idées divergent lorsque la précision est nécessaire.

Les définitions des différents objectifs apparaissent maintenant dans un certain nombre de normes pour les protocoles cryptographiques. Cependant, il y a un manque d'accord sur ce que sont les objectifs souhaitables d'authentification et d'établissement de clés, ainsi que les définitions de ce que ces objectifs devraient être.

7.1. Les objectifs basiques

Après avoir examiné les différents types d'adversaires, nous considérons à quel point l'établissement de clés et l'authentification de l'entité sont connectés.

Cela nous amène à considérer séparément d'abord les objectifs orientés utilisateur (user-oriented goals), puis les objectifs orientés clés (key-oriented goals).

7.1.1. Les modèles de sécurité

Certains modèles (un exemple particulier est celui utilisé par Burrows et al. [I.24]) ont supposé que les participants légitimes doivent agir honnêtement. Cela signifie que l'entité qui a été authentifié suivra la spécification du protocole, ce qui correspond à une situation dans laquelle les membres d'un groupe fermé d'utilisateurs se font mutuellement confiance pour agir correctement. Cela peut

être considéré comme un modèle faible qui ne doute pas de la corruption des participants internes. Un modèle d'attaque plus puissant est celui dans lequel les participants (authentifiés) peuvent agir malicieusement. Ce modèle a été utilisé par d'autres chercheurs, comme Bellare et Rogaway [I.25] qui supposent que l'adversaire est capable de corrompre tout participant à tout moment. Il est précisé par Gollmann [I.26] que cela correspond à la situation la plus réaliste dans des applications commerciales où la majorité des attaques du monde réel proviennent des participants internes.

7.1.2. L'établissement de clé ou l'authentification de l'entité ?

Dans la littérature au début des protocoles cryptographiques, il était courant de se référer à tous les protocoles concernés par la mise en place des clés de session en tant que «protocoles d'authentification». Cela n'est pas tout à fait satisfaisant parce que certains protocoles qui mettent en place des clés de session ne fournissent aucune authentification d'une partie à l'autre, tandis que d'autres protocoles conçus pour fournir l'authentification de l'entité ne comportent aucune clé de session.

Gollmann [I.27] a proposé un certain nombre d'options différentes pour ce qui pourrait être signifié par l'authentification. La première est la suivante.

• **Option 1** : Le protocole doit établir une clé de session fraîche, connu seulement par les participants à la session et éventuellement certains tiers de confiance.

C'est un objectif à propos de l'établissement de clés plutôt que l'authentification de l'entité.

• **Option 2** : **A** et **B** sont deux participants d'un protocole, une clé cryptographique associée à **B** est utilisé dans un message reçu par **A** (challenge ou timestamp) au cours de l'exécution du protocole.

Ceci est un objectif concernant l'authentification de l'entité. Il ne dit rien sur une nouvelle clé de session et peut être satisfaite par un protocole qui ne porte pas sur l'établissement de clés. Diffie et al. [I.28] disent qu'il est «admis que ces sujets doivent être considérés conjointement plutôt que séparément», tandis que Bellare et Rogaway [I.29] vont plus loin en affirmant :

> ... L'authentification de l'entité est rarement utile en absence d'une distribution de clé associée, tandis que la distribution des clés, par elle-même, est non seulement utile, mais elle est sensiblement plus utile quand une authentification de l'entité se produit conjointement…

7.2. Les objectifs orientés utilisateur

L'ISO Security Architecture [I.30] définit l'authentification de l'entité comme 'La corroboration qu'une entité est celle revendiquée'. Celle-ci n'est pas une définition aussi précise qu'on le souhaiterait car il ne précise pas quelle entité s'agit-il. Menezes et al. [I.31] donnent une définition plus complète de la manière suivante.

Définition I.14 : l'authentification de l'entité est le processus par lequel une partie est assurée (par l'acquisition de preuves) de l'identité d'une seconde partie impliquée dans un protocole, et que le second a effectivement participé [I.34].

Définition I.15 : l'authentification de l'entité forte de **A** à **B** est fournie si **B** a une nouvelle assurance qu'**A** est au courant que **B** est son entité homologue.

7.3. Les objectifs orientés clés

Menezes et al. [I.31] donne la définition suivante pour l'établissement de clés.

'L'établissement de clé est un processus ou un protocole par lequel un secret partagé devient disponible à deux ou plusieurs parties, pour une utilisation cryptographique ultérieure'.

Cette définition peut être étendue et rendue plus spécifique.

Définition I.16 : La clé de session partagée est une bonne clé pour **A** utilisable avec **B** si et seulement si (**A**) a l'assurance que :

• la clé est fraiche (key freshness) ;

• la clé est connue que par **A** et **B** ainsi que les tierces de confiance (key authentication).

Définition I.17 : Intégrité des clés est la propriété que la clé n'a pas été modifiée par l'adversaire (protocole de transport de clé), ou de manière équivalente n'a que les contributions des participants légitimes (protocole d'accord de clé).

7.4. Les objectifs renforcés

Il est possible d'envisager une combinaison des objectifs et sous-objectifs de l'établissement de clés et de l'authentification de l'entité. Il y a aussi des objectifs qui dépassent à la fois une bonne clé d'authentification et l'authentification de l'entité qui sont appelés ici objectifs renforcés. Quels sont les objectifs utiles à viser ? Un protocole qui prévoit seulement l'établissement de clé ne donne aucune assurance que le partenaire avec lequel la communication est souhaitée existe.

Depuis l'authentification de l'entité, comme dans définition. I.15, traite exactement cette préoccupation, il est naturel que les objectifs améliorés devraient inclure l'authentification de l'entité ainsi que l'établissement de clés.

Définition I.18 : La confirmation de clé de **A** à **B** est fournie si **B** a l'assurance que la clé **K** est une bonne clé pour communiquer avec **A**, et que le participant **A** est en possession de **K**.

Définition I.19 : Mutuelle croyance en clé **K** est prévu pour **B** que si **K** est une bonne clé pour une utilisation avec **A**, et **A** souhaite communiquer avec **B** en utilisant la clé **K** dont lequel **A** la croit bonne pour cet usage.

7.5 Objectifs concernant les clés compromises

Les objectifs considérés ici sont différents des objectifs discutés précédemment par qu'ils sont applicables uniquement dans des situations que nous supposons normalement ne pas arriver : quand une clé à long terme est compromise.

Si une telle clé est connue par un adversaire, alors il est clair que l'adversaire peut se faire passer pour l'utilisateur à tout moment. Donc, que peut-on sauver dans une telle situation ?

7.5.1. La confidentialité persistante (Forward Secrecy)

L'idée de la confidentialité persistante est que quand une clé à long terme est compromise, les clés de session qui ont été précédemment établies à l'aide de cette clé à long terme ne doivent pas être compromises aussi.

Définition I.20 : Un protocole d'établissement de clé qui fournit la confidentialité persistante si le compromis des clés à long terme d'un ensemble de participants ne compromet pas les clés de session établis dans des exécutions précédentes du protocole en impliquant ces participants.

Le concept crucial en fournissant la confidentialité persistante semble être la clé publique éphémère ; ceci est une clé publique qui est utilisée uniquement pendant la durée du protocole d'établissement de clé et est ensuite détruite avec la clé privée correspondante. Si la clé publique à long terme est utilisée uniquement pour authentifier la clé de session, puis la clé de session ne peut pas être récupérée sans la clé privée éphémère. Les clés éphémères les plus couramment utilisées sont de type nécessaire dans le protocole Diffie-Hellman [I.32]. Il convient de noter que ce n'est pas le seul type de clé publique ; clés éphémères peuvent être utilisées pour tout système cryptographique à clé publique.

Chapitre II :
Les automates cellulaires et la cryptographie

Chapitre II : Les automates cellulaires et la cryptographie

Sommaire

1. Introduction

Les automates cellulaires sont d'excellents exemples de systèmes avec de simples composants et interactions qui conduisent à des phénomènes complexes, ils peuvent être parfaitement adaptés à une étude et une analyse détaillée, parce qu'ils peuvent être simulés facilement dans un ordinateur.

La robustesse des automates cellulaires en tant que classe de modélisation est exposée dans la vaste gamme d'applications dans lesquelles ils ont été employés depuis leur introduction par J. von Neumann and S. Ulam [II.7, II.8]. Ils ont été utilisés dans une variété d'applications biologiques, y compris la modélisation de fibrillation cardiaque, les aspects du développement et les tests des théories de l'évolution. En physique, les automates cellulaires ont été utilisées pour modéliser les systèmes de réaction chimique non-linéaires, l'évolution des galaxies spirales et la croissance des cristaux dendritiques...

En théorie de calcul, les automates cellulaires peuvent être considérés comme des ordinateurs parallèles [II.9, II.10, II.11]. Ils ont été utilisés en tant que multiplicateurs parallèles [II.12, II.13], trieurs [II.14] et nombre de tamis [II.15]. Ils ont également été extrêmement utiles dans le traitement d'image et de reconnaissance de motif [II.16, II.17] ; et il est connu que certains automates cellulaires sont des ordinateurs universels, par conséquent équivalent à une machine de Turing [II.18].

2. Historique

Depuis les jours de Von Neumann et Ulam qui ont d'abord proposé le concept des automates cellulaires (CA), le livre récent de Wolfram « A New Kind of Science» [II.1], la structure simple des CA a attiré des chercheurs de diverses disciplines. Des conférences spécialisées et des numéros spéciaux de plusieurs revues sur CA ont été lancés dans les dernières décennies. Plusieurs universités ont également commencé à offrir des cours sur les automates cellulaires.

La raison de la popularité des automates cellulaires peut être attribuée à leur simplicité, et de l'énorme potentiel qu'ils détiennent dans la modélisation de systèmes complexes, en dépit de leur simplicité. Les automates cellulaires peuvent être considérés comme un modèle simple d'un système décentralisé spatialement étendu constitué d'un certain nombre de composants individuels (cellules). La communication entre les cellules qui le composent est limitée à l'interaction locale. Chaque cellule individuelle est dans un état spécifique qui

change au fil du temps en fonction des états de leurs voisinages. La structure générale peut être considérée comme un dispositif de traitement parallèle. Cependant, cette structure simple lorsque itérée plusieurs fois produit des motifs complexes affichant le potentiel de simuler différents phénomènes naturels sophistiqués.

En se basant sur le concept théorique de l'universalité, les chercheurs ont essayé de développer des architectures plus simples et plus pratiques des CA qui peuvent être utilisées pour modéliser des domaines d'application très divergentes. En 1970, le mathématicien John Conway a proposé son fameux jeu de la vie [II.2] qui a reçu un large intérêt parmi les chercheurs. Dans le début des années quatre-vingt, Stephen Wolfram a étudiée en profondeur une famille de règles d'automates cellulaires unidimensionnels simples (règles Wolfram désormais célèbres [II.3]) et a montré que même ces règles simples sont capables de simuler des comportements complexes. Conformément à ces études, nous décrivons une étude concise mise à jour de la théorie et les applications de ce modèle de calcul dans le domaine de la cryptographie précisément l'authentification.

Nous donnons d'abord une introduction pour les non-mathématiciens ; une définition mathématique des automates cellulaires suit.

3. Définition non-formelle d'un Automate Cellulaire

Les automates cellulaires (CA) sont des systèmes dynamiques dans lesquels l'espace et le temps sont discrets. Les cellules sont disposées sous forme d'une structure régulière du réseau et chacune doit avoir un nombre fini d'états. Ces états sont mis à jour de façon synchrone selon une règle locale d'interaction spécifiée.

Par exemple, un automate cellulaire unidimensionnel à deux états sera composé d'une ligne de cellules, dont chacune peut prendre la valeur 0 ou 1. En utilisant une règle spécifiée, les valeurs de toutes les cellules sont mises à jour de façon synchrone en fonction de temps. Avec un automate de K états, chaque cellule peut prendre l'une des valeurs entières entre 0 et K-1.

L'élément de base d'un automate cellulaire est la cellule. Une cellule est une sorte de mémoire qui permet de stocker un état. Cependant, l'état d'une cellule peut être plus complexe et comporter plusieurs attributs, mais le nombre d'états possibles pour la cellule reste fini.

Un espace cellulaire unidimensionnel CA-1D à double état est représenté sur la figure II.1. Un nœud donné peut être soit activé (attribué une valeur d'état 1) ou désactivé (valeur 0). Les nœuds les plus proches de tout nœud donné sont ceux immédiatement à sa gauche et à sa droite. Pour chaque nœud appelé une cellule centrale, un voisinage de rayon r est défini, constitué par N = 2r+1 cellules, y compris la cellule même. Dans ce cas, nous pouvons avoir un voisinage local de trois cellules (r = 1). L'état d'un nœud à l'instant t+1 sera déterminé par les états des cellules de son voisinage à l'instant t.

Figure II. 1 - Espace cellulaire unidimensionnel

Une autre structure est celle d'un réseau carré CA-2D représenté sur la figure II.2. L'intersection des carrés forment les nœuds de l'automate. Les nœuds les plus proches d'un nœud donné sont les quatre nœuds Nord, Sud, Est et Ouest, en se déplaçant le long des lignes reliant les nœuds.

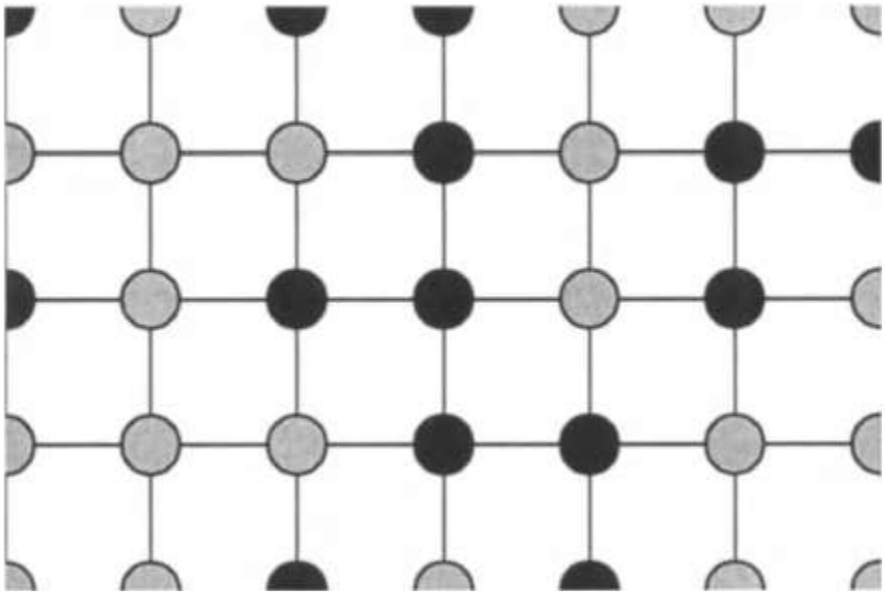


Figure II. 2 - Espace cellulaire carré à deux dimensions

Le nœud spécifié, avec ses quatre voisins les plus proches, forment un voisinage. Encore une fois, l'état d'un nœud donné à l'instant t+1 sera déterminé à partir des états des nœuds de son voisinage à l'instant t. La règle d'évolution est appliquée à tous les nœuds (et à leur voisinage associé) en même temps. Un espace cellulaire tirée d'un réseau hexagonal est représenté sur la figure II.3.

Dans ce cas, un voisinage possible est ce d'un nœud et de ses six plus proches voisins.

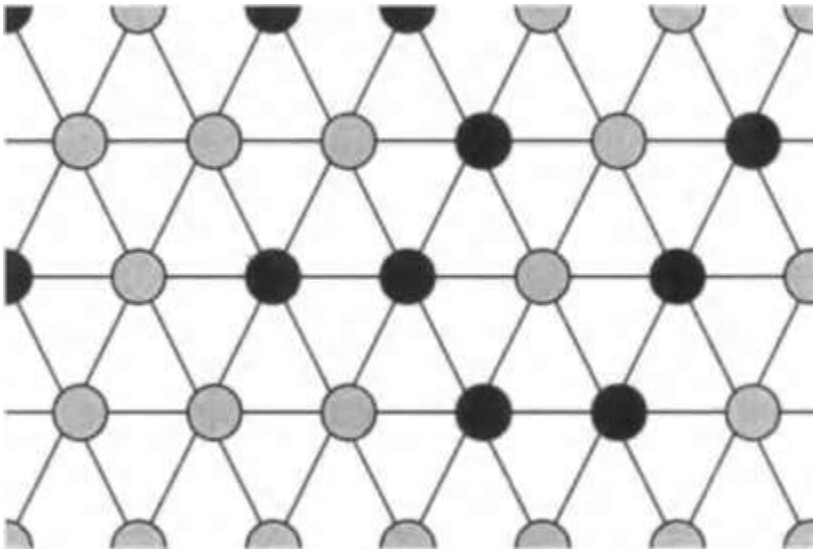


Figure II. 3 - Espace cellulaire hexagonale à deux dimensions

Les cellules sont disposées sur un réseau dont la géométrie est celle d'un réseau régulier de dimension n. Cependant, dans la plupart des cas, on choisit un réseau carré avec n =1,2. En dimension deux, on utilise aussi le réseau hexagonal. On peut construire des réseaux à n = 3, mais ceux-ci sont plus difficiles à visualiser. Les réseaux unidimensionnels sont plus faciles à étudier et permettent d'obtenir des résultats théoriques. La terminologie de site du réseau s'emploie aussi pour désigner une cellule du réseau. Le terme site présuppose une appartenance et une localisation sur un réseau.

La dynamique d'un CA nécessite la définition de la notion de voisinage d'une cellule : c'est l'ensemble des cellules susceptibles d'interagir avec elle à un temps donné. Le voisinage contient toute l'information nécessaire pour la mise à jour de l'état de la cellule à chaque pas de temps. Il détermine donc l'évolution temporelle de la cellule. Précisons qu'une cellule appartient à son propre voisinage.

Les voisinages les plus courants dans un environnement de deux dimensions (CA-2D) sont :

- **Voisinage de Von Neumann :** On considère les seuls voisins Nord/Sud/Est/Ouest. (Figure II.4)

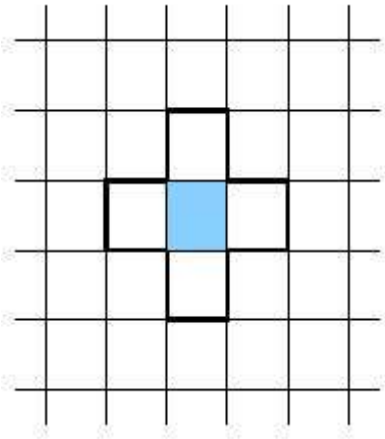


Figure II. 4 - Voisinage de Von Neumann

- **Voisinage de Moore :** On ajoute les diagonales au voisinage de Von Neumann. C'est le cas du Jeu de la vie. (Figure II.5)

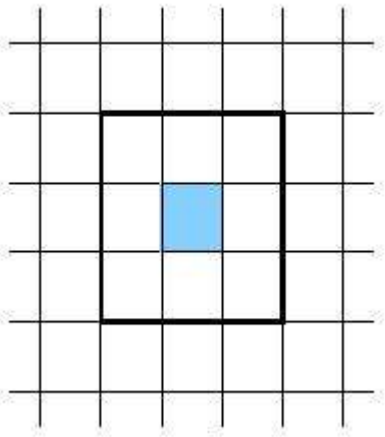


Figure II. 5 - Voisinage de Moore

- **Voisinage de Moore étendu :** On étend la distance de voisinage de Moore au-delà de un. (Figure II.6)

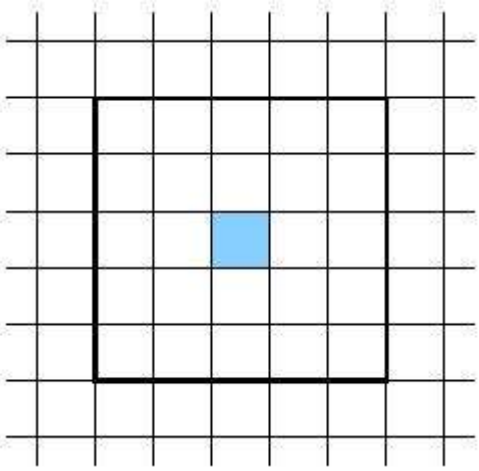


Figure II. 6 - Voisinage de Moore étendu

- **Voisinage de Margolus :** On considère des ensembles de 2x2 éventuellement alternés. (Figure II.7)

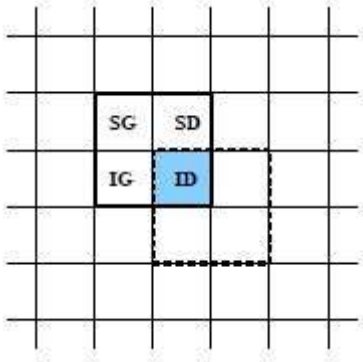


Figure II. 7 - Voisinage de Margolus

En pratique, le nombre de cellules du réseau est bien évidemment fini. Il faut alors préciser la notion de voisinage pour les cellules aux bords du domaine. L'information concernant la localisation d'une cellule est codée dans la cellule même. Une règle d'évolution différente devrait être appliquée aux cellules de bord. En réalité, plutôt que de définir une règle différente sur les bords, on étend le voisinage des cellules de bord. On définit alors plusieurs conditions aux bords :

- **Périodique :** En dimension deux, cela signifie que les bords gauche et droite, ainsi que supérieur et inferieur sont connectés.
- **Fixe :** Le voisinage est complété avec des cellules dont l'état est fixe.
- **Adiabatique :** Le voisinage est complété en dupliquant la cellule.
- **Symétrique :** Le voisinage est complété en dupliquant la cellule du voisinage dans la direction opposée.

C'est souvent la nature du système étudié qui qui dicte le choix de la condition aux bords.

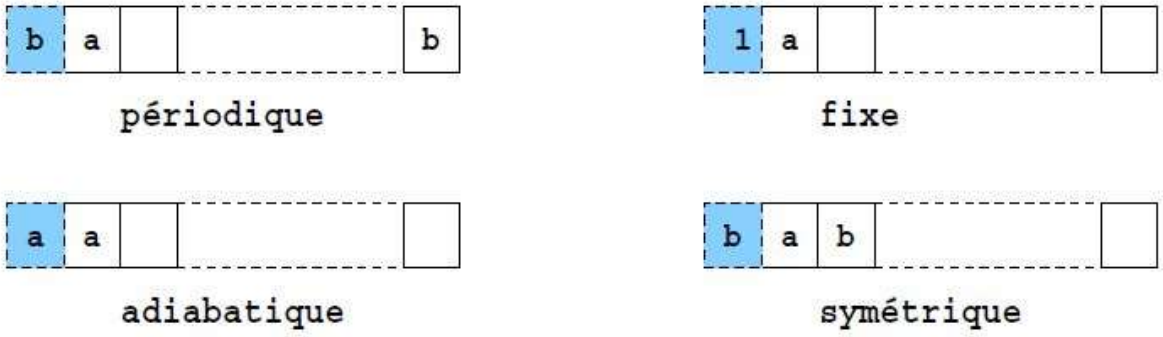


Figure II. 8 - Les conditions aux bords

La dynamique est décrite à l'aide d'une règle d'évolution. La principale source de variété dans l'univers des automates cellulaires est le grand nombre de règles potentielles. La règle d'évolution permet de déterminer l'état d'une cellule au pas de temps suivant comme une fonction de l'état des cellules de son voisinage. En fait, si **k** est le nombre d'états possibles de chaque cellule et m le nombre de cellules de son voisinage, il existe $\mathbf{(k^k)^m}$ règles possibles.

4. Définition formelle d'un Automate Cellulaire

Il n'y a pas d'accord sur la formalisation ou notation standard pour les automates cellulaires. La notation utilisée dans cette thèse est basée sur la formalisation de Delorme et Mazoyer [II.20], mais avec des modifications afin de permettre une adaptation plus tard à un ordre supérieur des automates cellulaires, et quelques autres petits changements qui ont été jugé approprié.

4.1. Qu'est-ce qu'un Automate Cellulaire ?

Nous devons définir quelques concepts pour commencer. Un réseau est une grille ordonnée, typiquement $\boldsymbol{Z^d}$ (ou $\boldsymbol{L}$ par certains auteurs), avec $\boldsymbol{(d \in Z_+)}$. Un voisinage est un sous-ensemble ordonné fini de $\boldsymbol{Z^d}$ et on désigne par $\boldsymbol{N}$. Une cellule est un objet qui a un état à la fois. L'ensemble de tous les états admissibles est un ensemble fini appelé alphabet, désigné par $\boldsymbol{S}$. Il y a une cellule à chaque nœud du réseau.

L'alphabet ***S*** est très souvent un ensemble {0, 1, …, n-1} ou une variante de celui-ci, bien que ce soit en aucun cas obligatoire.

Définition II.1 : Un automate cellulaire d-dimensionnel ***A*** (un ***d-CA***) est un quadruplet (***d***, ***S***, ***N***, ***f***), où ***d*** est la dimension, ***S*** est un alphabet, ***N*** est un voisinage et ***f*** est une fonction, appelée la règle de transition locale : $f : S^{|N|} \rightarrow S$.

4.2. La fonction de transition locale

La fonction f : $S^{|N|} \rightarrow S$ est appelée fonction de transition locale abrégée LTF (Local transition function). L'ensemble $S^{|N|}$ est l'ensemble de tous les états de voisinage possibles. Chaque élément de $S^{|N|}$ est un ensemble ordonné d'états $\{s_0, s_1, \ldots, s_{|N|-1}\}$ avec $s_i \in S$. Pour certains automates cellulaires, le LTF peut être défini par une formule, par exemple $f(s_0, s_1, s_2) = s_0 + s_1 + s_2 \pmod 2$. Dans ce cas ***S*** est le plus souvent un sous-ensemble des nombres entiers.

La fonction de transition locale détermine la manière dont l'état de chaque cellule est modifié à partir d'un instant à l'autre. Cette décision est généralement basée sur le propre état actuel de la cellule ainsi de ses voisins. De plus, les fonctions locales peuvent être déterministes ou probabilistes.

Chaque CA évolue dans les étapes de temps discrets et toutes les cellules sont mises à jour simultanément. L'état global φ est l'état de chaque cellule dans le réseau à l'instant t :

$$\varphi : Z^d \rightarrow S$$

L'état d'une cellule arbitraire à $x \in Z^d$ peut maintenant être spécifié par $\varphi(x)$. Soit $\mathbf{N = \{z_0 ; z_1 ; \ldots z_{n-1}\}}$ est le voisinage de taille n d'un CA arbitraire et soit φ^{t+1} l'état global des cellules à l'instant t+1. L'état d'une cellule arbitraire x à l'instant t+1 est donné par l'équation II.1 :

$$\varphi^{t+1}(x) = f[\varphi^t(x+z_0), \varphi^t(x+z_1), \ldots \varphi^t(x+z_{n-1})], \forall x \in Z^d \quad \ldots (II.1)$$

L'évolution temporelle d'une cellule est donnée par l'application récursive de la fonction locale de la cellule :

$$\varphi^t(x) \rightarrow \varphi^{t+1}(x) \rightarrow \varphi^{t+2}(x) \rightarrow \ldots$$

4.3. Les automates cellulaires Elémentaires (ECA)

Pour un voisinage d'une cellule x de rayon r sa taille est donnée par : **|N| = 2r+1**. Si **r=1** et **|S|=2**, nous avons un voisinage à trois cellules **N = {-1, 0, 1}** avec deux états possibles pour chaque cellule. Par conséquent, nous avons $\mathbf{S^{|N|} = 2^3 = 8}$ configurations possibles du voisinage, qui peut être représenté comme représenté sur la figure II.9, où les carrés noirs et blancs représentent les états '1' et '0', respectivement.

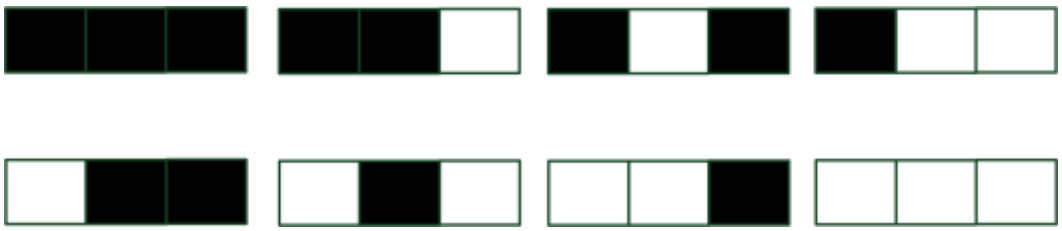


Figure II. 9 - Toutes les configurations des états possibles de voisinage dans chaque ECA

Dans le cas d'ECA, l'équation (II.1) devient :

$$\varphi^{t+1}(x) = f\,[\varphi^t(x-1), \varphi^t(x), \varphi^t(x+1)]$$

Nous pouvons avoir jusqu'à $(2^2)^3$=256 ECAs possibles. Prenons l'exemple simple suivant, où la fonction de transition ne dépend que du voisin gauche et celui de la droite de $\boldsymbol{x : \varphi^{t+1}(x) = \varphi^t(x-1) + \varphi^t(x+1) \ (mod\ 2)}$. Nous pouvons représenter cette fonction par un tableau qui mappe l'état du voisinage à l'état suivant pour la cellule centrale :

Tableau II. 1 - Fonction de transition locale pour un ECA

$\phi^t(x-1)$	$\phi^t(x)$	$\phi^t(x+1)$	$\phi^{t+1}(x)$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

Ou par la représentation graphique ci-dessous :

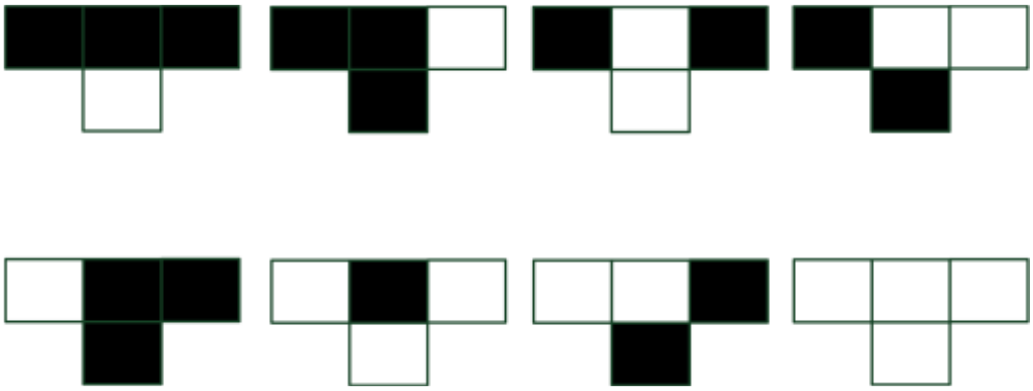


Figure II. 10 - Fonction de transition locale pour un ECA

Où la rangée supérieure représente l'état d'une cellule et de ses voisins à l'instant t, tandis que la rangée du bas représente l'état d'une cellule à l'instant t+1.

Pour coder ce tableau, Wolfram a suggéré la lecture de la dernière colonne comme un nombre binaire de 8 bits. Depuis 01011010 en binaire est 90 en décimal, Wolfram nommé cette règle de CA par 90.

Par conséquent, les 256 ECA sont venus à être connu sous le nom des règles de Wolfram et les numéros associés comme les numéros de Wolfram.

4.4. La fonction globale

Représentons ***c*** comme la configuration actuelle de l'automate avec $c \in Z^d$. La configuration suivante est donnée par $\oint(c)$, où $\oint : \sum Z^d \rightarrow \sum Z^d$. Nous appelons $\oint$ la fonction globale. L'évolution temporelle de l'AC est alors :

$$c \rightarrow \oint(c) \rightarrow \oint^2(c) \rightarrow \ldots$$

Nous nommons la séquence c, $\oint$ (c), $\oint$2 (c), ... l'orbite de c. Dans la figure II.11 nous pouvons observer tous les types possibles d'évolutions périodiques de chaque CA :

- point fixe : $\oint$ (c)=c ;
- périodique : $\oint$ (c)=c pour certains t ∈ Z_+ ;
- éventuellement fixe : ∃ t ∈ N : $\oint^{t+1}$ (c)= $\oint^t$ (c) ;
- éventuellement périodique : ∃ t ∈ N et k ∈ Z_+ : $\oint^{t+k}$ (c)= $\oint^t$ (c).

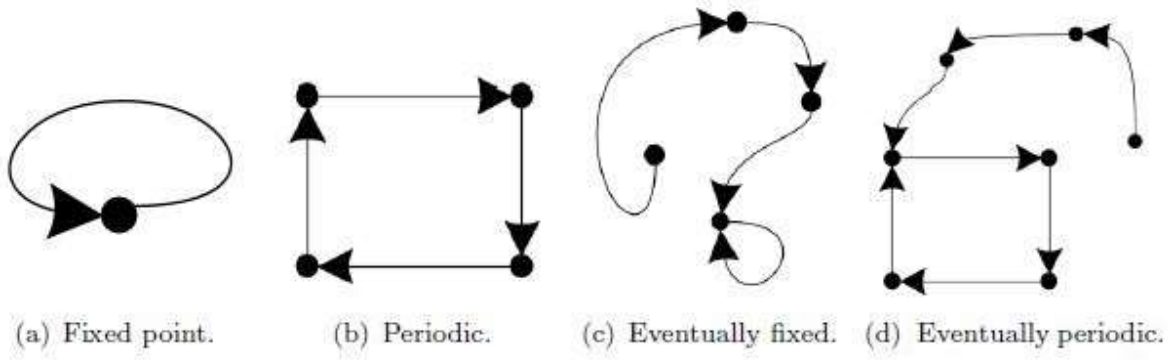


Figure II. 11 - Orbite

Il est courant d'utiliser des graphiques, appelés diagrammes spatio-temporels, pour la représentation de l'évolution de la configuration temporelle d'un CA unidimensionnel. Dans ces diagrammes, les configurations sont représentées horizontalement et leur évolution sont tirés vers le bas comme on peut le voir dans la figure II.12. Naturellement, ces schémas ne sont possibles que pour d≤2.

Figure II. 12 - Diagramme spatio-temporel

4.5. L'ensemble de voisinage

Nous avons déjà donné une définition non-formelle sur le voisinage dans la section précédente. Nous essayons maintenant de définir le voisinage d'une façon formelle.

Pour un automate cellulaire à deux dimensions, le voisinage de von Neumann et de Moore sont les deux formes les plus courantes formes de voisinage (Section 3). Pour d ≥ 1 et de rayon r ces voisinages sont définies par :

$$N_v(y) = \{ x | x \in Z^d, d_1(x, y) \leq r \}$$

Pour le voisinage de von Neumann, contenant $(2r + 1)^d$ éléments :

$$N_M(y)= \{ x | x \in Z^d, d_\infty(x, y) \leq r \}$$

Pour le voisinage de Moore, où d_1 et d_∞ sont des mesures associées à la norme Manhattan $||y||_1$ et la norme maximale $||y||_\infty$ respectivement. Pour un vecteur n-dimensionnel y, ces normes sont donnés par :

$$||\mathbf{y}||_1 = \sum_{i=1}^{n} |y^i|$$

$$||\mathbf{y}||_\infty = \max\{|\mathbf{y}|_i \mid i \in \{1, 2 \dots, n\}\}$$

Noter que dans un CA unidimensionnel, le voisinage de von Neumann et de Moore sont identiques.

Un autre voisinage habituel est le voisinage de second ordre, qui sera expliqué et utilisé plus tard.

5. Les systèmes de classification des Automates Cellulaire

Wolfram a proposé un système de classification qui a divisé les règles d'automates cellulaires en quatre classes, en fonction du résultat de l'évolution du système à partir d'un état initial [II.38]. Ces classes de Wolfram sont les suivantes :

5.1. La règle de Wolfram numéro 1 (W1)

Les automates de cette classe meurent complètement après seulement peu d'étapes de l'évolution. L'évolution du système conduit à des configurations de points uniformes fixes après un nombre fini d'étapes, après aucun changement ne se produit. Environ 86% des 256 ECA appartiennent à cette classe. Parmi les exemples de ces règles nous citons 119 et 222.

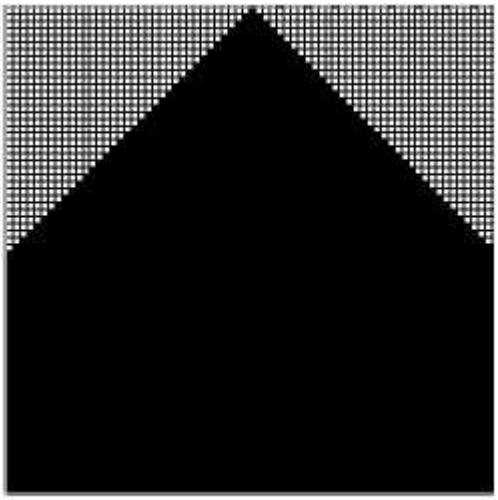


Figure II. 13 - Exemple de diagramme spatio-temporel de W1 (règle 222)

5.2. La règle de Wolfram numéro 2 (W2)

L'évolution conduit à une configuration à répétition périodique, à savoir, après quelques itérations, les configurations commencent à se répéter. Environ 9% des 256 ECA appartiennent à cette classe. Les règles 4 et 126 sont des exemples pour ce cas.

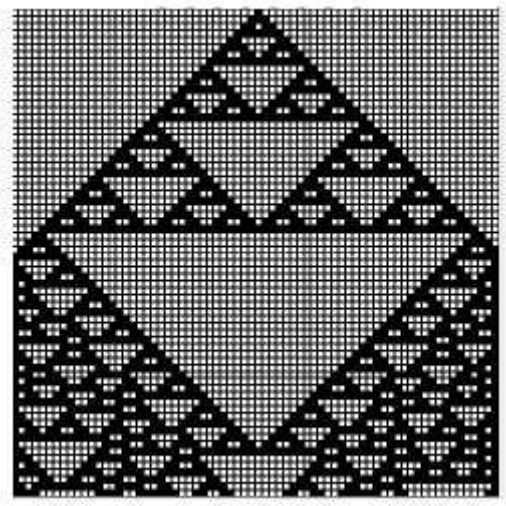


Figure II. 14 - Exemple de diagramme spatio-temporel de W2 (règle 126)

5.3. La règle de Wolfram numéro 3 (W3)

L'évolution conduit à un comportement chaotique. Pratiquement toutes les conditions initiales conduisent à des motifs qui ne se répètent jamais. De petits changements se propagent généralement à un taux uniforme et cela peut éventuellement affecter toutes les parties du système. Environ 4% des 256 ECA appartiennent à cette classe. Les règles 30 et 90 en sont des exemples.

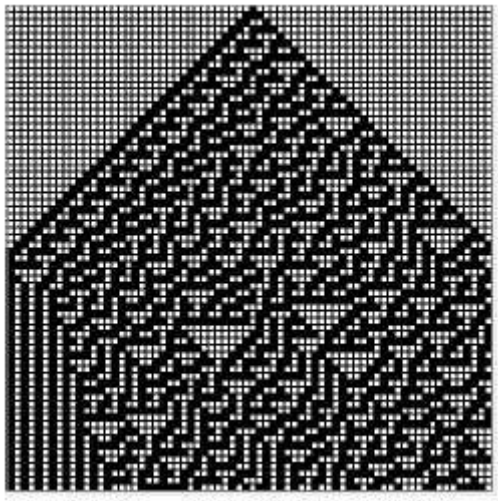


Figure II. 15 - Exemple de diagramme spatio-temporel de W3 (règle 30)

5.4. La règle de Wolfram numéro 4 (W4)

L'évolution conduit à localiser et à propager des structures avec des motifs complexes qui peuvent durer pendant des longueurs arbitraires de temps. Seulement 4 des 256 automates cellulaires élémentaires appartiennent à cette classe. Les exemples incluent les règles 54 et 110.

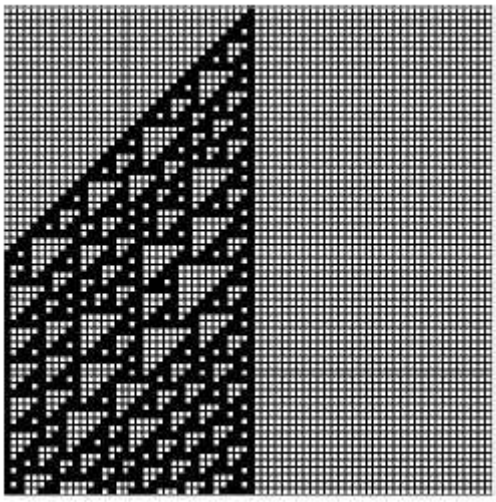


Figure II. 16 - Exemple de diagramme spatio-temporel de W4 (règle 110)

Cependant, cette classification n'est pas unique. D'autres auteurs tels que Li-Packard [II.39, II.40], Kurka [II.41, II.42] et Culik & Yu [II.43], ont également fourni des systèmes de classification alternatifs.

6. Les propriétés fondamentales des CA

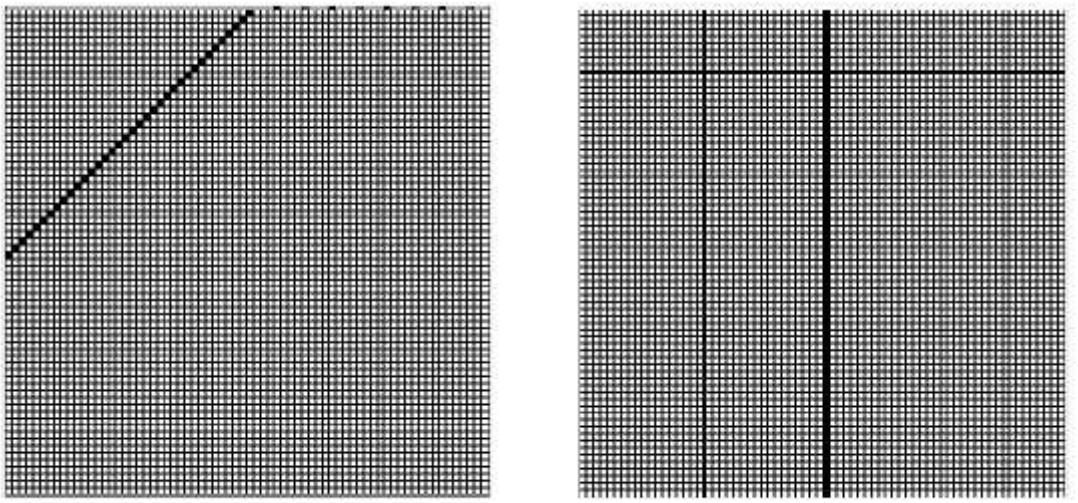
Ci-dessous quelques propriétés remarquables d'automate cellulaire dont :

- **L'uniformité (homogénéité) :** Si la totalité des cellules évoluent en utilisant la même règle de transition, et les règles de transition restent inchangeables avec le temps.
- **La non-uniformité (hétérogénéité) :** Si la totalité de ses cellules n'évoluent pas en utilisant la même règle de transition, ou lorsque les règles de transition changent avec le temps.
- **L'asynchronisme :** Si la mise à jour des cellules du CA se fait de manière indépendante, c'est-à-dire sans aucune restriction que les cellules doivent être mises à jour en même temps. Sinon l'automate cellulaire est synchrone.

- **La réversibilité :** On appelle automate inverse l'automate qui permet de revenir aux états précédents. Autrement dit, l'automate est réversible si la configuration à l'instant t+1 n'a qu'une seule configuration précédente possible a l'instant t, quel que soit t.
- **Le parallélisme :** Un système est dit parallèle si ses constituants évoluent simultanément et de manière indépendante.
- **La proximité (LOCALITY) :** Le nouvel état d'une cellule ne dépend que de son état actuel et de l'état du voisinage immédiat.
- **La reproduction :** Le but de Von Neumann, a l'origine, lorsqu'il a conçu le principe des automates cellulaires, était de concevoir un mécanisme copiant la vie dans le sens où il pourrait se reproduire lui-même.
- **L'indécidabilité :** L'une des caractéristiques importantes des automates cellulaires est le caractère d'indécidabilité qui touche le nombre de leurs propriétés. Pour déterminer si un automate cellulaire possède un inverse est indécidable : il ne sera jamais possible d'écrire un programme prenant en paramètres un automate quelconque et pouvant décider si oui ou non cet automate possède un inverse.

7. Les automates cellulaires réversibles

Automates cellulaires réversibles (RCA) ou automates cellulaires inversibles (ICA), sont des automates cellulaires capables de récupérer l'état initial de l'automate. Dans la figure II.17, nous pouvons observer deux exemples simples de règles réversibles.



(a) Rule 170. (b) Rule 204.

Figure II. 17 - Des exemples d'automates cellulaires réversibles

Automates cellulaires réversibles sont des CA pour lesquels la fonction globale est inversible. Pour être réversible, sa fonction globale doit être une fonction bijective $\oint : \sum Z^d \rightarrow \sum Z^d$. Nous pouvons dire que un automate cellulaire A est réversible s'il existe un automate cellulaire B, qui est tel que :

$$\oint \circ \psi = \textit{fonction d'identité}$$

Où $\oint$ et ψ sont les fonctions globales de A et B respectivement.

Nous pouvons alors énoncer le théorème suivant :

Théorème II.1 [II.44, II.45] : Un automate cellulaire est dit réversible si et seulement s'il est injectif.

Le problème de décider si un automate à une dimension est inversible, est un problème décidable, même pour un automate cellulaire infinie à une dimension, tel que démontré par Amoroso et Patt [II.46].

Théorème II.2 [II.46] : Il est décidable pour un automate cellulaire unidimensionnel qu'il soit réversible ou non.

Cependant, la réversibilité (surjectivité) dans des dimensions supérieures à un sont indécidables.

Théorème II.3 [II.47, II.48] : Il est indécidable pour un automate cellulaire à deux dimensions de décider s'il est réversible ou non.

Comme les automates cellulaires inversibles constituent un très petit sous-ensemble de CA, leur réversibilité peut être une question indécidable. On peut fournir des moyens pratiques pour construire des automates cellulaires réversibles, à condition qu'on réduise encore plus l'ensemble de CA inversible, en veillant à ce que, pendant la construction d'un automate la règle vers l'arrière est construite en même temps que la règle vers l'avant. Nous présentons dans la section suivante une technique simple qui peut être utilisée pour créer de simples RCA. Pour d'autres techniques, nous nous référons aux travaux de Toffoli et Margolus [II.49].

8. Les automates cellulaires de second ordre

Suite aux travaux de Fredkin, Vichniac et d'autres auteurs [II.1, II.50] ont suggéré l'utilisation de règles spéciales pour atteindre l'inversibilité.

Dans ces RCA, le nouvel état d'une cellule est déterminée non seulement par la cellule elle-même et ses voisins à un instant en arrière, mais aussi par l'état de la cellule à deux instants en arrière (voir les figures II.8 et II.9).

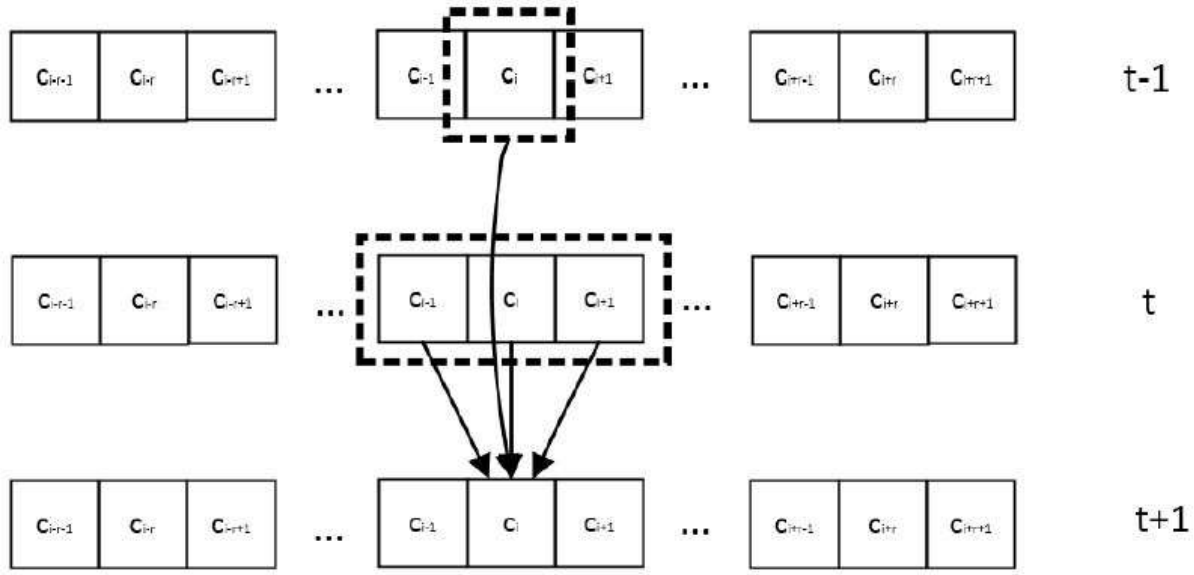


Figure II. 18 - Simple CA explicitement défini pour être réversible

La configuration c^{t+1} est déterminée par la combinaison linéaire des deux configurations précédents c^t et c^{t-1} :

$$c^{t+1} = \Gamma(c^t) - c^{t-1}$$

Où $c \in \sum Z^d$ et Γ est la loi dynamique du système, qui dans notre cas est la fonction globale $\oint$. En réarrangeant simplement la formule ci-dessus, on obtient l'inverse :

$$c^{t-1} = \Gamma(c^t) - c^{t+1}$$

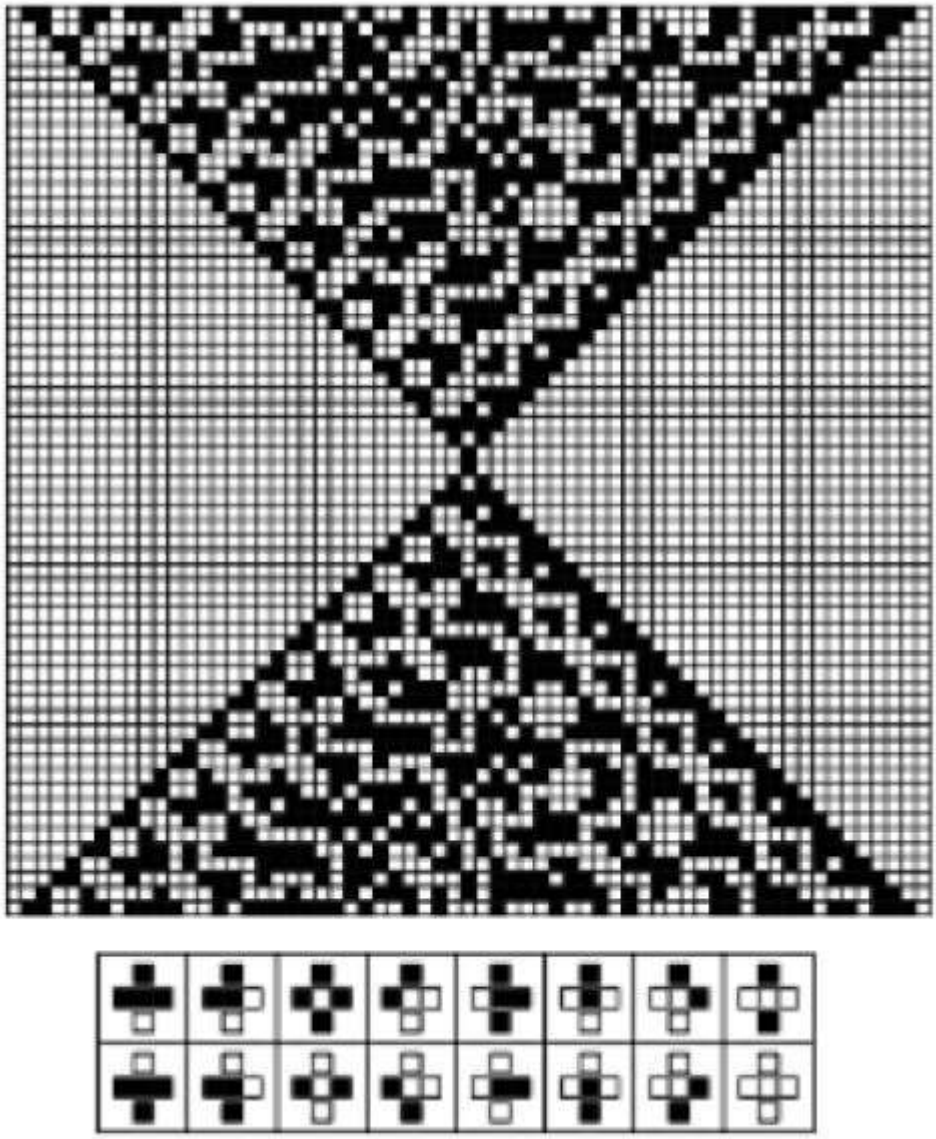


Figure II. 19 - Simple CA explicitement conçu pour être réversible

9. Les automates cellulaires à mémoire

Les automates cellulaires de second ordre ne sont qu'un cas particulier des automates cellulaire à mémoire.

Cela veut dire qu'on peut considérer les automates cellulaires pour lesquels les états des cellules voisines à l'instant t, ainsi que t-1, t-2… y contribuent pour déterminer l'état au temps t+1. Ceci est le concept de l'automate cellulaire à mémoire (MCA). [II.4]

Si la fonction de transition globale du MCA est la suivante :

$$C^{t+1} = \phi(C^t, C^{t-1}, \ldots, C^{t-m+1})$$

Alors le MCA est appelé m-ième ordre MCA. A noter que, dans ce cas, il est nécessaire de connaître m configurations initiales, C (0), ..., C (m-1), afin de calculer les configurations restantes du m-ième ordre MCA. Le MCA qui évolue avec la fonction globale de transition est de toute évidence réversible. [II.5]

10. Introduction à la cryptographie à base des CA

Les automates cellulaires (CA) sont un moyen de calcul intéressant à étudier en raison de leur simplicité et de leur fonctionnement intrinsèquement parallèle. Ces caractéristiques en font un outil de calcul utile et efficace pour des applications telles que la cryptographie, en particulier lorsqu'ils sont appliqués sur un matériel parallèle spécialisé.

Certains des premiers travaux de connexion des automates cellulaires avec la cryptographie étaient introduits par Wolfram [II.21, II.22]. Il a montré qu'un automate cellulaire particulier pourrait être utilisé comme un générateur de bits pseudo-aléatoire rapide. Ce flux de bits aléatoires peut être utilisé comme une clé symétrique ; Une opération de XOR est effectuée avec le message d'entrée et un flux aléatoire, formant le cryptogramme. Le message d'entrée peut être rétabli en faisant la même opération du XOR entre le cryptogramme et le flux aléatoire. Ce système a été analysé par Meier et Staffelbach [II.23], qui a trouvé une attaque montrant que la réduction de la taille de l'espace clé suffit à rendre le système pratiquement incertain.

Gutowitz a présenté une approche plus sophistiquée [II.24, II.25] en combinant les résultats à partir d'un CA irréversible avec ceux d'un CA réversible dans plusieurs tours de calcul pour le chiffrement / déchiffrement. Il affirme que le système est résistant à une méthode d'attaque avancée connue sous le nom de cryptanalyse différentielle [II.26]. Nous ne pouvions pas trouver des articles ultérieurs contestant cette affirmation.

10.1. Les fonctions de hachage à base des CA

Plusieurs auteurs ont suggéré des modèles pour la construction d'une fonction de hachage à base des CA. Les automates cellulaires semblent se prêter bien pour la conception des fonctions de hachage en raison de leur nature régulière et parallélisme élevé.

Damgård propose un moyen de construire une sortie d'une fonction de hachage de taille 128 bits en utilisant la règle 30 sur 512 cellules [II.51]. Daemen, Govaerts

et Vandewalle [II.52] montrent comment trouver des collisions avec les CA à base des fonctions de hachage de Damgård. Ils présentent également leur propre fonction de hachage de 257 bits appelé Cellhash qui utilise les règles 30 et 150. Mihaljevic, Zheng et Imai proposent une fonction de hachage de type DaviesMeyer en utilisant les CA additifs linéaires programmables [II.53]. Les auteurs notent que leur fonction de hachage est nettement plus rapide que les fonctions de hachage dédiées telles que MD5, SHA-1...

Le travail effectué dans le domaine des fonctions de hachage avec les CA est de meilleure qualité que dans d'autres domaines. Les auteurs proposant des algorithmes de hachage ont fourni les spécifications complètes ainsi que des analyses de sécurité approfondies, ouvrant la voie d'acceptation par la communauté cryptographique.

10.2. Les codes d'authentification de message à base des CA

Deux documents ont exploré l'utilisation des CA pour générer des codes d'authentification de message (MAC). L'idée générale est d'utiliser le texte en clair comme état initial, puis faire évoluer du CA par un certain nombre d'itérations afin de calculer le MAC tout en mélangeant des informations clés.

Dasgupta, Chattopadhyay et Sengupta proposent un modèle pour la création d'un code d'authentification de message (MAC) en utilisant les règles linéaires 90 et 150 [II.54].

Mukherjee, Ganguly et Chaudhuri en 2002 [II.55] montrent comment construire un code d'authentification de message (MAC) en utilisant CA. Ils soulignent que leur schéma est immunisé contre les attaques par force brute. Lee, et al. Ont montré que le système de Mukherjee (et al.) est vulnérable à une attaque de message choisie en raison de sa conception intrinsèquement régulière [II.56].

Le fait que seuls deux articles ont été publiés sur le sujet, montre clairement que cela n'a mené à aucun résultat pratique. Néanmoins, ce domaine semble mûr pour la recherche, comme les CA sont connus pour leur parallélisme efficace en hardware qui est attrayant pour ce domaine.

10.3. Les systèmes cryptographiques symétriques à base des CA

Plusieurs systèmes de clé symétrique ont été proposés récemment. Le système introduit par Tomassini et Perrenoud [II.27] a utilisé la même idée que Wolfram en faisant une opération XOR bit à bit entre la clé aléatoire et le message pour obtenir le cryptogramme.

Bouvry et al. [II.28] ont continué cette ligne de recherche, en utilisant une technique de calcul évolutif appelé «programmation cellulaire» pour choisir les automates cellulaires de génération de bits aléatoires appropriés.

Seredynski et Bouvry [II.29] décrivent un crypto-système qui ajoute essentiellement au régime de Wolfram l'évolution du cryptogramme en utilisant un automate cellulaire réversible unidimensionnel pour plusieurs générations. Le texte en clair est ensuite récupéré en inversant l'automate cellulaire jusqu'à ce que le cryptogramme d'origine soit obtenu, puis un XOR est effectué à nouveau avec la clé. Anghelescu et al. [II.30] utilisent une combinaison de CA comme un générateur de nombres aléatoires, avec une autre combinaison de CA qui est périodique, de sorte que l'évolution du CA soit périodique pendant une demi-période pour chiffrer et l'autre moitié pour déchiffrer.

Cependant, des algorithmes assez similaires par Nandi et al. [II.31] et Sen et al. [II.32] ont été contestés [II.33, II.34].

Récemment Faraoun, K. M. [II.6] a proposé une nouvelle façon de chiffrement authentifié et randomisé en utilisant les automates cellulaires réversibles. L'auteur affirme que le système est le premier système de chiffrement par blocs qui assure la sécurité contre les deux attaquants actifs et passifs par le biais d'un mécanisme d'authentification forte. L'auteur affirme aussi que le système est plus rapide que les standards existants du chiffrement authentifié.

10.4. Les systèmes cryptographiques à clé publique à base des CA

Peu de recherches ont été faites vers le développement de systèmes cryptographiques à clé publique à base de CA. Seuls quelques-uns sont disponibles : le crypto-système de Tao-Chen [II.35] et un autre système proposé par Guan [II.36] n'utilise qu'un CA non-uniforme à la fois. Un autre système a également été proposé par Kari [II.37].

L'objectif général d'un système de chiffrement à clé publique basé sur les automates cellulaires réversibles (RCA) est de concevoir un RCA qui est très difficile à inverser sans la connaissance d'un certain secret. De cette façon, le RCA peut être publié et son inverse peut être gardé comme une clé privée. Kari souligne l'importance que le RCA soit au moins bidimensionnelle, car il existe des algorithmes pour inverser n'importe quel RCA unidimensionnel [II.46], et aussi à cause du théorème II.3 indiqué ci-dessus.

Récemment Zhang, Xing, et al. Ont proposé une nouvelle façon de chiffrement à clé publique basé sur les automates cellulaires multicouche (LCA) [II.57]. Les auteurs prouvent que leur schéma et plus performant et plus rapide que le fameux crypto-système RSA. Les auteurs affirment aussi que leur schéma est sémantiquement sécurisé contre l'attaque à texte clair choisi.

Partie II : La première approche

Chapitre III :

Etat de l’art sur l’authentification et l’établissement de clés à base de clé secrète partagée en utilisant les automates cellulaires unidimensionnels

Partie II : La première approche

Chapitre III : Etat de l'art sur l'authentification et l'établissement de clés à base de clé secrète partagée en utilisant les automates cellulaires unidimensionnelle

Sommaire

1. Introduction

Le système d'authentification à distance par mot de passe a été proposé pour la première fois par Leslie Lamport [III.1], et le système d'OTP (One Time Password) a été proposé par Neil Haller [III.2]. Cependant leurs inconvénients sont la synchronisation de la séquence, les frais généraux élevés du hachage, et le coût de maintien et de protection de la liste des mots de passe partagés n'est pas évolutif (scalable) [III.1, III.2].

De nombreux chercheurs ont exploré différents protocoles d'authentification de l'utilisateur pour des fonctionnalités et des environnements différents. Récemment, plusieurs protocoles d'authentification à base de cartes à puce ont été proposés et analysées [III.3-III.9]. La plupart de ces schémas utilisent des opérations d'exponentiation modulaire et de multiples instances de fonction de hachage.

Pendant ce temps, les automates cellulaires ont été acceptés comme un bon modèle de calcul, et ont été utilisés dans divers domaines, mais peu utilisés dans les systèmes d'authentification.

La première tentative a été faite par Nandi et al. [III.10] qui ont proposé un système d'authentification d'un utilisateur distant en utilisant les CA dans le but de fournir une complexité de calcul optimal, mais ce schéma utilise de multiples instances de fonction de hachage à base des CA, et ne fournit pas l'authentification du serveur auprès du client, n'est donc pas robuste contre (Server counterfeit attack). En outre, le partage d'une clé de session après la phase d'authentification pour empêcher le détournement n'est pas mise en œuvre.

Dans [III.11], les mêmes auteurs ont proposé un schéma d'authentification mutuelle à distance basée sur les CA afin d'assurer une authentification mutuelle entre le serveur et les clients, et de partager une clé de session après la phase d'authentification, mais cela n'est effectué que par l'ajout de plusieurs instances de fonction de hachage qui augmente les frais généraux de calcul du schéma.

Dans [III.12], Jeon et al. Ont proposé un système d'authentification basé sur CA singulier afin d'optimiser les exigences de calculs et de transmissions. Au lieu d'utiliser des fonctions de hachage, le système utilise des opérations binaires, alors que la sécurité du protocole est basée sur l'irréversibilité et l'unicité des configurations du CA. Ce protocole à un problème d'évolutivité (scalabilité) puisque sa sécurité est très sensible au choix de la taille et de la longueur du CA

sous-jacent. Une adaptation du protocole proposé dans [III.12] a été proposée dans le papier [III.13] et utilisée comme un protocole d'authentification RFID pour la vie privée et l'anonymat.

2. Etat de l'art

Peu de protocoles ont été proposés sur l'authentification en utilisant les automates cellulaires nous citons dans cette section les travaux qui ont été publiés dans ce contexte avec une analyse en terme de sécurité et d'implémentation ; nous montrons aussi les limites de chaque approche.

2.1. L'approche CARA : Tripathy, S., Nandi, S., & Chowdhury, A. R. (2006) [III.10]

Dans cet article, Tripathy, S. et al proposent CARA ; un système d'authentification d'un utilisateur distant à base des automates cellulaires. CARA n'exige pas au serveur de maintenir des listes de mots de passe.

La sécurité du système provient des opérations réversibles des automates cellulaires qui sont simples à mettre en œuvre. Comme ce schéma utilise les automates cellulaires du second ordre expliqués dans le chapitre précédent alors, la prochaine configuration d'un RCA dépend à la fois de sa configuration précédente ainsi que sa configuration courante, et ne peut pas être générée si l'une d'entre elles est inconnue. De plus, si nous ne connaissons que la configuration suivante, la propriété de réversibilité assure que nous ne pouvons pas trouver ni la configuration courante ni les configurations précédentes. L'utilisation de la règle 30 de Wolframe est suggérée en raison de l'aspect aléatoire présenté par ses configurations consécutives.

Une autre contribution de cet article est le concept du nonce à base de la mémoire cache pour éviter les problèmes de synchronisation de l'horodatage. La mémoire cache est un bon générateur de nombres aléatoires, car c'est l'endroit de mémoire le plus rapidement changeable au sein de l'ordinateur. Cela rend la corrélation entre les nonces successifs imprévisibles pour un adversaire.

Le serveur gère une petite file d'attente FIFO de nonces générés périodiquement depuis la mémoire cache. Chaque nonce est utilisé dans un scénario de défi-réponse, où le serveur transmet un nonce et espère de le récupérer sous une forme codée. Dans la phase de défi, le serveur associe l'identité de l'utilisateur au nonce le plus frais et lui envoie à l'utilisateur concerné comme indiqué sur la figure III.1.

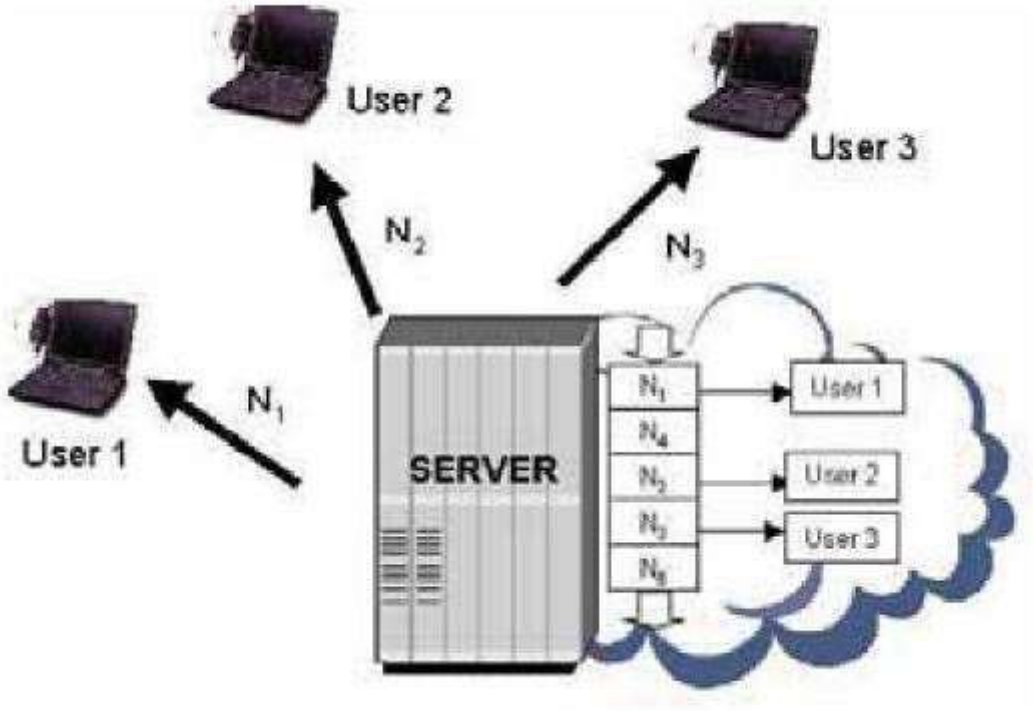


Figure III. 1 - La gestion de la file d'attente (FIFO) des nonces

Sur réception d'une réponse, le serveur vérifie l'existence du nonce transmis dans sa file d'attente FIFO. Une entrée dans la file d'attente-nonce reste en vie pour une période excédant légèrement le temps nécessaire à l'étape d'authentification. Si le nonce reçu est dupliqué dans le nonce-file d'attente le serveur exécute l'étape d'authentification d'utilisateur correspondante ; sinon il suppose que c'est une attaque de rejeu potentielle et refuse la demande du service.

2.1.1. Les détails de fonctionnement du CARA

CARA fonctionne en trois phases, à savoir. La phase d'enregistrement, la phase d'identification et la phase d'authentification. Au cours de ces opérations une fonction de hachage forte **H** est utilisée. Les détails sur une telle fonction de hachage à base des CA peuvent être trouvés dans [III.15].

2.1.1.1. La phase d'enregistrement

L'utilisateur $\mathbf{U_i}$ fournit son identité $\mathbf{ID_i}$ et son mot de passe $\mathbf{PW_i}$ au serveur, en utilisant un canal sécurisé. Le serveur code l'information comme représenté sur l'équation (III.1), sur une carte à puce, qui est donnée à l'utilisateur. Le serveur n'a besoin de mémoriser aucune de ces procédures.

$$S_i = RCA\ (PW_i, RCA\ (PWs, ID_i)) \ldots (III.1)$$

2.1.1.2. La phase d'identification

L'utilisateur doit s'identifier avant de demander le service. La phase d'identification se déroule comme suit :

- L'utilisateur envoie son nom d'utilisateur et une demande de service au serveur ;
- Pour une demande de service valide, le serveur défie avec un nonce **Ni** ;
- L'utilisateur répond par la chaîne $\mathbf{Y_i||N_i}$ calculée en utilisant les équations (III.2) et (III.3) ;

$$X_i = RCA\ (PW_i, S_i) \ldots (III.2)$$

$$Y_i = H\ (RCA\ (X_i, N_i)) \ldots (III.3)$$

2.1.1.3. La phase d'authentification

L'étape d'authentification comprend le serveur en calculant la valeur de $\mathbf{Z_i}$ à partir de l'équation (III.4). Ceci peut se faire facilement puisque les trois paramètres $\mathbf{PW_s}$, $\mathbf{ID_i}$ et $\mathbf{N_i}$ sont connus pour le serveur.

$$Z_i = H\ (RCA\ (RCA\ (PW_s, ID_i), N_i)) \ldots (III.4)$$

L'utilisateur U_i est considéré comme authentifié si $\mathbf{Z_i = Y_i}$. Cette égalité peut être prouvée par des substitutions appropriées à (III.3). L'ensemble du protocole d'authentification d'utilisateur distant est résumée dans la figure (III.2).

2.1.1.4. La phase de modification du mot de passe

Si un utilisateur soupçonne son mot de passe d'avoir été compromis, il peut demander qu'il soit modifié. Ce processus est très similaire à un nouvel enregistrement, sauf que l'information existante sur l'ancienne carte est conservée.

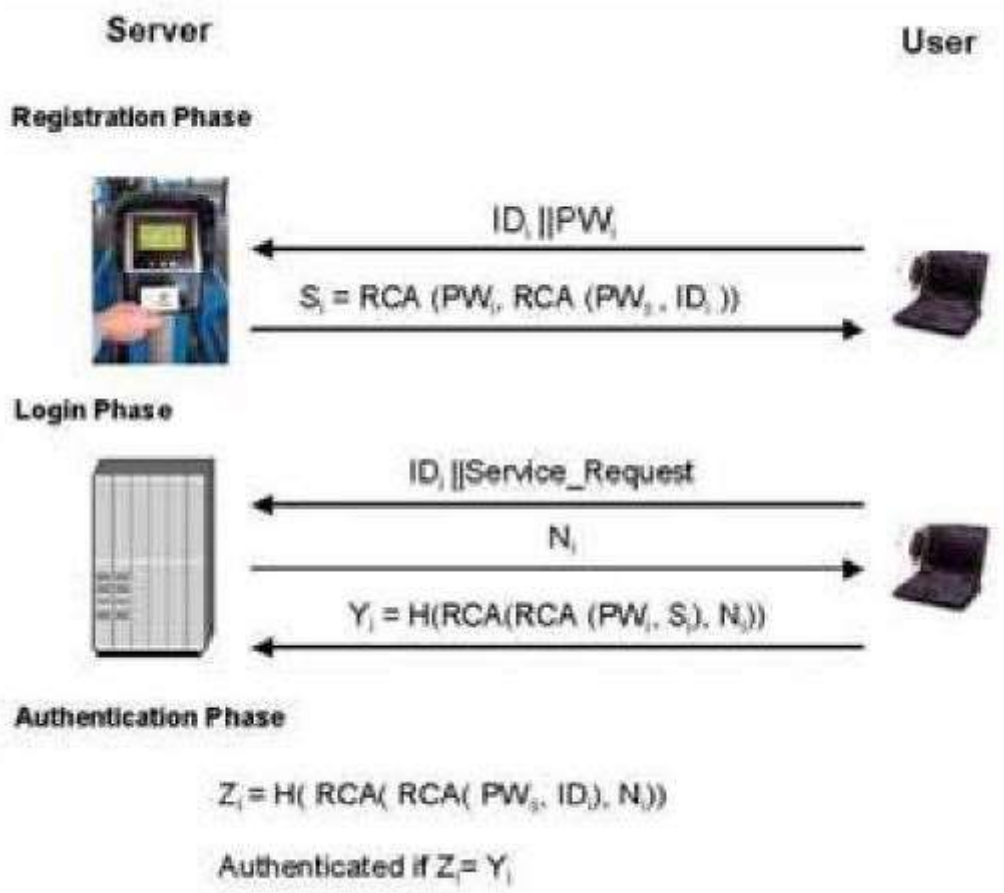


Figure III. 2 - Fonctionnement du CARA

2.1.2. L'analyse de CARA

La performance du CARA est analysée à partir de son aspect de sécurité, ainsi que son implémentation matérielle.

2.1.2.1. L'analyse de sécurité

Nous présentons ici la résilience de CARA contre diverses attaques :

• Un adversaire peut capturer l'empreinte Y_i haché, envoyé par l'utilisateur. Par la suite, il va essayer d'extraire le mot de passe du serveur à partir de la hache capturée. Cependant une telle tentative devient impossible en raison de la forte résistance à la collision de la fonction de hachage.

• Les attaques de rejeu sont empêchées grâce à l'utilisation du cache-nonce et une file d'attente FIFO, comme expliqué précédemment. Fondamentalement ce schéma annule le nonce après une certaine période et donc ne pas accepter le message rejoué.

• Un imposteur malveillant $\mathbf{U_M}$ de l'utilisateur $\mathbf{U_i}$, échoue dans sa tentative parce qu'il n'a pas le mot de passe $\mathbf{PW_i}$ de l'utilisateur $\mathbf{U_i}$. Par conséquent $\mathbf{U_M}$ ne peut pas terminer la troisième étape du protocole.

• La fabrication de carte est impossible en raison de la propriété de la non-linéarité de la classe RCA fondée sur la règle 30. Comme mentionné précédemment, la prochaine configuration d'un RCA ne peut être générée si l'une de sa précédente ou ses configurations actuelles sont inconnues.

Le serveur est le seul à connaître son mot de passe $\mathbf{PW_s}$. Par conséquent, il est le seul qui peut extraire des informations significatives à partir de $\mathbf{S_i}$.

2.1.2.2. L'analyse d'implémentation

Les opérations des CA sont des procédures simples, binaires qui peuvent être exécutées indépendamment pour chaque bit. Ce parallélisme inhérent prévoit une implémentation très rapide et efficace. En second lieu CARA ne nécessite aucun chiffrement, ni de la part des clients, ni de la part du serveur. Cela ajoute à la vitesse avec laquelle la procédure d'authentification peut être réalisée.

Cela empêche également les attaques DOS sur le serveur par la surcharge des demandes de services non essentiels. La littérature montre que les mécanismes qui n'emploient pas des techniques de chiffrement ont généralement recours au hachage. Cependant le hachage est également un processus long, et donc CARA minimise l'utilisation de hachage en utilisant le hachage à base des CA, qui peut être implémenté pour le calcul parallèle.

2.1.3. Les limites du CARA

En termes de sécurité nous montrons que CARA est vulnérable à diverses attaques.

2.1.3.1. Vulnérable à l'attaque par cassage de mot de passe

Le protocole CARA proposé dans [III.10] est vulnérable à l'attaque du cassage de mot passe ; Un espion qui est sur l'écoute du réseau peut capturer le message Y_i de l'équation (III.3) puis il peut trouver le mot de passe en essayant un dictionnaire de mots de passe jusqu'à l'obtention de la même chaine $\mathbf{Y_i}$. D'où le schéma est vulnérable à l'attaque (Off-line password guessing attack). En ce qui concerne une attaque en ligne (On-line password guessing attack) de ce genre les auteurs n'ont rien indiqué comment leurs protocole peut contrecarrer contre une telle attaque.

2.1.3.2. Vulnérable à l'attaque par usurpation d'identité du serveur :

Supposons qu'un adversaire $\mathbf{S_z}$ qui veut se faire passer pour le serveur légal **S** pour tromper l'un de ces utilisateurs $\mathbf{U_i}$. Puisque le schéma ne fournit aucune authentification du serveur auprès de l'utilisateur ; l'utilisateur n'a aucune garantie qu'il soit connecté sur le bon serveur ; et donc l'adversaire $\mathbf{S_z}$ réussira son attaque. En conséquent, ce schéma est vulnérable à l'attaque (Server counterfeit attack). En outre, ce schéma ne fournit pas une authentification mutuelle entre le serveur et l'utilisateur.

2.1.3.3. L'absence de la phase d'établissement de clé session

Au fait, la phase d'authentification n'est qu'un début d'une communication sécurisée entre les participants légitimes ; alors si la même clé secrète partagée est utilisée pour chiffrer les données réelles durant plusieurs sessions cela peut engendrer des problèmes tels que le rejeu d'un message d'une session précédente. Une autre raison pour laquelle une nouvelle clé doit être générée pour chaque session est qu'elles sont susceptibles d'être utilisées avec une variété de formats de données régulières, ce qui les rend cibles à la cryptanalyse. Donc Pour atteindre un niveau de sécurité élevé, Il faut que les données de chaque session soient chiffrées par une clé aléatoire, imprévisible et différente pour chaque session, qui rend les messages échangés d'une session donnée valable uniquement pour cette session, et ne pouvaient pas être réutilisé dans d'autres sessions. Le CARA ne fournit aucun mécanisme de génération de clés de session ; par conséquent il est cible à la cryptanalyse. En outre, tout type d'attaque devient possible en se basant sur la notion que les anciens messages peuvent être rejoués dans une session ultérieure.

En terme de performances il est évident que CARA utilisent plusieurs instances de fonctions de hachage ce qui dégrade les performances d'implémentation surtout quand il s'agit d'une implémentation sur un dispositif de faible calcul tels que les cartes à puce.

2.2. L'approche CARMA : Tripathy, S., Chowdhury, A. R., & Nandi, S. (2006). [III.11]

Dans cet article, Tripathy, S. et al proposent CARMA ; un système d'authentification mutuelle à base des automates cellulaires. CARMA ; un système facilement implémentable dont la sécurité est basé sur les opérations à base des RCA. Par exemple, si nous ne connaissons que la configuration suivante, la propriété de réversibilité assure que nous ne pouvons pas trouver ni la configuration courante ni les configurations précédentes. L'utilisation de la règle 30 de Wolframe suggérée en raison du caractère aléatoire présenté par ses configurations consécutives.

En utilisant le même principe du protocole précédent CARA comme CARMA utilisent une file d'attente FIFO des nonces-cache ainsi que le principe de défi-réponse afin d'assurer l'authentification mutuelle.

2.2.1. Le principe de fonctionnement de CARMA

CARMA comporte une phase de pré-distribution de la clé et une phase d'authentification mutuelle. Au cours de ces opérations une fonction de hachage forte H [III.15] est utilisée. Les opérations du fonctionnement du protocole CARMA sont illustrées sur la figure (III.3).

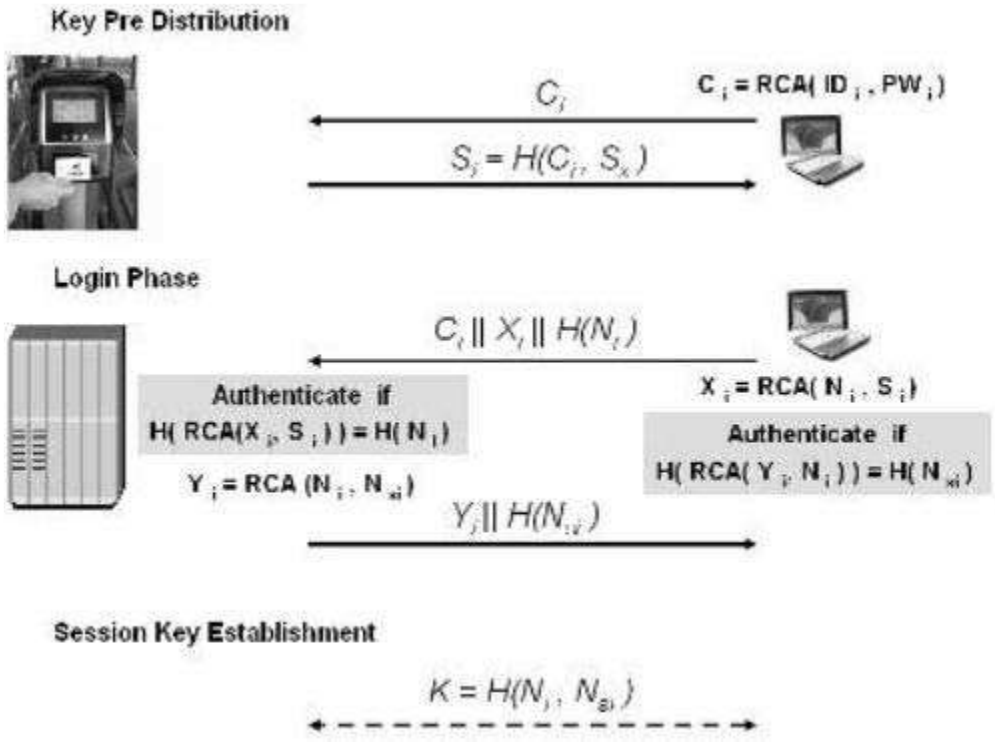


Figure III. 3 - Fonctionnement du CARMA

2.2.1.1. La phase de pré-distribution de la clé

L'utilisateur $\mathbf{U_i}$ fournit son identité $\mathbf{ID_i}$ et son mot de passe $\mathbf{PW_i}$ au serveur, en utilisant un canal sécurisé. Le serveur encode l'information, ainsi que le code d'authentification du serveur $\mathbf{S_x}$, tel que représenté dans l'équation (III.5). Ce secret est ensuite transmis à l'utilisateur et est utilisé dans des transactions ultérieures. Le serveur n'a besoin de mémoriser aucune de ces procédures.

$$C_i = RCA\ (ID_i, PW_i) \ldots (III.4)$$

$$S_i = H\ (C_i, S_x) \ldots (III.5)$$

2.2.1.2. La phase d'authentification mutuelle

L'utilisateur doit s'identifier avant de demander pour le service. **A** ce stade, il devient nécessaire pour le serveur et l'utilisateur de s'authentifier l'un auprès de l'autre. La phase d'authentification se déroule comme suit.

• L'utilisateur envoie un codage composé de son identifiant ainsi que son mot de passe $\mathbf{C_i}$, le secret du serveur $\mathbf{X_i}$ et son propre nonce $\mathbf{N_i}$ au serveur d'authentification. Cette chaîne codée Z_i est représentée dans l'équation (III.7).

$$X_i = RCA\ (N_i, S_i) \ldots (III.6)$$

$$Z_i = C_i \,||\, X_i \,||\, H\ (N_i) \ldots (III.7)$$

• Le serveur authentifie l'utilisateur si et seulement s'il est capable de récupérer le nonce N_i généré par l'utilisateur $\mathbf{U_i}$ depuis la chaine codée $\mathbf{Z_i}$. Pour ce faire, l'information calculée au niveau du serveur doit correspondre à l'information transmise, c.-à-d. $\mathbf{H\ (N_i)}$. Ce calcul peut être exprimé par l'équation (III.8).

$$H\ (N_i) = H\ (RCA\ (X_i, H\ (C_i, S_x))) \ldots (III.8)$$

• De son côté, le serveur défie l'utilisateur par un nonce-cache $\mathbf{N_{si}}$. La chaîne de défi $\mathbf{Y_i||H\ (N_{si})}$ est envoyée à l'utilisateur, avec $\mathbf{Y_i}$ étant calculé comme indiqué dans l'équation (III.9).

$$Y_i = RCA\ (N_i, N_{si}) \ldots (III.9)$$

• L'utilisateur prouve l'authenticité du serveur s'il peut recouvrir le nonce-cache $\mathbf{N_{si}}$ en partant du principe que seul le serveur est capable de générer une telle chaine. Cela est effectué en appliquant l'équation (III.10).

$$H\ (N_{si}) = H\ (RCA\ (Y_i, N_i)) \ldots (III.10)$$

• Chacune des parties se met d'accord sur la clé de session $\mathbf{K_{si}}$ valable pour une seule session, calculée par l'équitation (III.11).

$$K_{si} = H (N_i, N_{si}) \dots (III.11)$$

2.2.2. L'analyse de sécurité de CARMA

Suivant le même principe de la section précédente nous allons présenter ici la résistance de CARMA contre diverses attaques ;

• Un adversaire peut capturer l'empreinte $\mathbf{Y_i}$ haché, envoyé par l'utilisateur. Par la suite, il va essayer d'extraire le mot de passe du serveur à partir de la hache capturé. Cependant une telle tentative devient impossible en raison de la forte résistance à la collision de la fonction de hachage. Cet argument est valable pour les attaques de dictionnaire ainsi.

• Un imposteur malveillant $\mathbf{U_M}$ de l'utilisateur $\mathbf{U_i}$, échoue dans sa tentative parce qu'il n'a pas le mot de passe $\mathbf{PW_i}$ de l'utilisateur $\mathbf{U_i}$. Par conséquent $\mathbf{U_M}$ ne peut pas terminer la troisième étape du protocole.

• La fabrication de carte est impossible en raison de la propriété de la non-linéarité de la classe RCA fondée sur la règle 30. Comme mentionné précédemment, la prochaine configuration d'un RCA ne peut être générée si l'une de sa précédente ou ses configurations actuelles sont inconnues.

• Cependant, la caractéristique la plus attrayante de CARMA est sa mise en œuvre rapide et efficace, en particulier avec le mécanisme d'authentification dans les deux sens (authentification mutuelle).

2.2.3. Les limites du CARMA

2.2.3.1. Vulnérable à l'attaque par vol de table de vérification :

Le serveur doit maintenir une liste des mots de passe des utilisateurs afin qu'il soit capable d'accomplir l'équation (III.8) du protocole ; par conséquent le système peut être vulnérable à l'attaque de modification. En outre, ce protocole n'aide pas la réduction du coût de stockage.

2.2.3.2. Vulnérable contre le vol de carte à puce et l'absence d'authentification forte :

CARMA n'utilise qu'un seul facteur d'authentification « ce que j'ai » ; l'utilisation du mot de passe n'intervient nulle part lors de la phase d'authentification au niveau de l'utilisateur. Une authentification basée sur un seul facteur n'est pas une authentification forte (strong authentication). En outre si un utilisateur perd

sa carte à puce et qu'elle tombe dans les mains d'un adversaire, ce dernier peut se faire passer pour l'utilisateur légitime.

2.2.3.3. Vulnérable à l'attaque par cassage de mot de passe :

Puisque C_i est fixe et utilisable à chaque session en lui envoyant avec la chaine $\mathbf{Z_i}$ dans l'équation (III.7) dans il peut être récupérer facilement à partir de cette chaine puisqu'il est concaténé avec d'autres éléments. Une attaque off-line peut être effectuée sur $\mathbf{C_i}$ afin de deviner le mot de passe correcte de l'utilisateur. En ce qui concerne une attaque en ligne (On-line password guessing attack) de ce genre les auteurs n'ont rien indiqué comment leurs protocole peut contrecarrer contre une telle attaque.

2.3. L'approche de Jeon, J. C., & Yoo, K. Y. (2006) [III.12]

Les schémas d'authentification d'OTP (One-Time Password) ont grandi en fonction de la synchronisation de temps ou les fonctions de hachage à sens unique, cela nécessite une haute complexité de calcul. Afin de remédier à ces problèmes, le présent protocole fournit un système d'authentification à faible complexité qui se compose uniquement d'opérations binaires logiques telles que XOR, AND, OR et NOT. Ce système minimise fortement la complexité de calcul et de transmission et résout les problèmes de temps ou de synchronisation de séquence en appliquant les automates cellulaires singuliers SCA sur la base de la non-réversibilité et l'unicité de la configuration de l'état. Ainsi, notre système d'authentification sécurisé peut être efficacement utilisé pour d'autres applications nécessitant une authentification qui est sécurisé contre les attaques passives et actives basées sur le rejeu des mots de passe réutilisables capturés.

2.3.1. L'authentification basée sur SCA

La sécurité du système est fondée sur la non-réversibilité et l'unicité de la configuration de l'état. Une telle fonction doit être facilement calculable vers l'avant, mais irréversible, et les états évolués doivent être distinctifs. Afin d'atteindre les conditions, un tel système doit calculer et enregistrer la règle de singularité et la longueur en fonction de l'état initial donné. Une telle longueur représente le nombre des états uniques dans un CA.

La figure III.4 montre les configurations de transition et la longueur lorsque l'état initial est un vecteur de 4 bits (1011). Le CA non singulier est que les règles appliquées 90 et 150 aient la propriété cyclique que l'état initial apparaît après un certain nombre d'évolutions. Le CA singulier est que les règles appliquées 171 et 129 aient la propriété qu'un état indéfini apparaisse après une certaine

période qui est appelée longueur **L**. Il montre que chaque CA a les états uniques dans les limites de sa longueur (**L**).

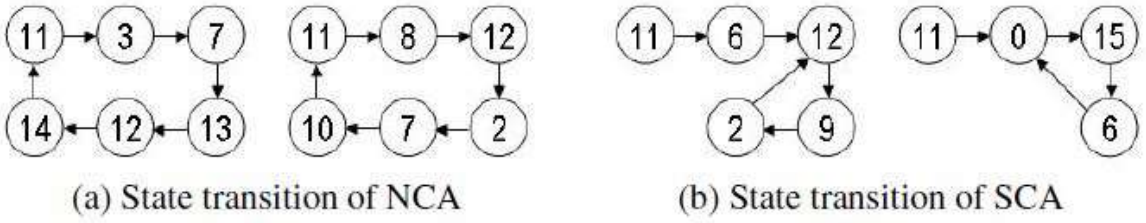


Figure III. 4 - Les configurations de transition et la longueur pour un VI (1011)

Dans ce système, il n'y a pas d'états qui se chevauchent dans une longueur donnée afin qu'il garantisse l'unicité de la configuration de l'état. Le nombre d'états uniques est exactement égal à sa longueur.

2.3.1.1. La phase d'enregistrement

• **S-1 :** Un utilisateur (**A**) choisit une phrase de passe **ρ**, puis la transmet au système (**B**).

• **S-2 :** La phrase de passe est concaténée avec un ''seed'' (ensemble de bits aléatoires), **ς**, par le participant **B** où **($\delta = \rho || \varsigma$)**. **B** en décide une règle singulière **ω**, et trouve la longueur **τ** du SCA. Ensuite, il calcule et enregistre **$\delta_o = T_\omega^\tau(\delta)$**, et initialise son compteur pour **A** à **C_A=1**.

• **S-3 :** Bob transfère **ω**, **t**, et **δ** au participant **A**.

2.3.1.2. La phase d'authentification

• **L-1 : A** calcule **$\delta_i = T_\omega^{\tau-i}(\delta)$**, et transmet **A**, **i** et **δ_i** à **B**.

• **L-2 :** B vérifie **$i = C_A$**, et **$T_\omega(\delta_i) = \delta_{i-1}$**. Si les deux vérifications réussissent, **B** accepte le mot de passe, fixe **$C_A \leftarrow C_A+1$**, et enregistre **δ_i** pour la prochaine vérification de session.

Le système proposé se compose de deux phases : La phase d'enregistrement et la phase d'authentification. Dans la phase d'enregistrement, les informations sont échangées via un canal sécurisé pendant que le message d'authentification est envoyé au système par un canal non sécurisé. L'authentification efficace OTP proposée fonctionne comme ci-dessus.

Dans la phase de configuration, **ρ** peut prendre toute longueur entre 64 bits et 96 bits, et **ς** devrait être bits de 32 bits à 64 bits, de sorte que le secret **δ** est initialisée comme 128 bits de longueur.

La notation || indique l'opération de concaténation. En (S-2), **ω** peut être juste une règle unique ou la combinaison de plusieurs règles, par exemple, <107, 230> où les règles sont appliquées au CA en alternance.

Le mot de passe pour la ième session d'identification, $\mathbf{1 \leq i \leq \tau}$, est défini comme $\boldsymbol{\delta_i = T_\omega^{\tau-i}(\delta_0)}$. Dans la phase d'identification, pour la ième session, **B** possède au préalable l'identité et le compteur de l'utilisateur, avec le vérificateur, $\boldsymbol{\delta_{i-1}}$, de sorte que **B** vérifie simplement l'authentification de l'utilisateur en appliquant la fonction de transition $\mathbf{T_\omega}$ une seule fois. Après avoir utilisé toutes les configurations pendant les temps **T**, la règle singulière **ω** doit être changée en une autre règle **ω'**.

2.3.2. Analyse et discussion

Comme tout protocole ayant ces avantages conformes à ces objectifs aura ainsi ces limites. Dans cette section nous allons discuter sur les avantages et les limites de ce système en termes de sécurité et de performance.

Afin de résoudre le problème de haut calcul de hachage, ce système a profité des caractéristiques des CA où le calcul ne se compose que des opérations binaires logiques telles que XOR, AND, OR et NOT, tandis que la fonction de hachage est composé non seulement des opérations binaires logiques, mais aussi le rembourrage (padding), annexion (appending) de longueur, et des opérations supplémentaires. En outre, les calculs de CA sont effectués en parallèle.

Ce système a un chemin minimal pour vérifier qu'un utilisateur est certifié dans une phase d'identification, et qu'il n'a pas besoin de la synchronisation du temps. Les auteurs affirment que même si **A** et **B** sortent de leur synchronisation à cause de l'instabilité du réseau informatique, **A** envoie $\boldsymbol{\delta_i}$ et **B** utilise $\boldsymbol{T_\omega(\delta_j)}$ pour l'authentifier, avec $\mathbf{i \neq j}$; alors cela peut être détecté en appliquant de façon répétée **ω** à la valeur d'authentification de **B** jusqu'à ce qu'un accord soit obtenu.

Les concepteurs de ce système recommandent que toute taille de la configuration de l'état, $\mathbf{|\delta|}$ soit de 128 bits qui se composent de **ρ** et **ς**. Afin de réduire les risques techniques tels que la recherche exhaustive où les attaques par dictionnaire, la phrase de passe secrète de l'utilisateur doit être comprise entre 64 bits et 96 bits.

Cela semble assez long pour être sûr et assez court pour être saisie manuellement. Les concepteurs recommandent également un CA de voisinage

égal à 3 (**r=1**), un **CA** de voisinage de plus de trois ne garantit pas une meilleure sécurité, mais augmente la complexité des calculs.

Pendant ce temps, la longueur **τ**, devrait être raisonnablement assez longue, mais pas trop. Si **τ** d'un CA est trop courte ou trop longue, cela provoque le renouvellement fréquent du mot de passe devient devinable par l'attaquant. Une autre recommandation est que la longueur du CA soit de plusieurs centaines à plusieurs milliers de bits.

2.3.2.1. Robuste contre l'attaque par cassage de mot de passe :

Il est mathématiquement impossible pour l'attaquant de choisir un mot de passe qui est identique au mot de passe courant δ_i depuis les sessions précédentes δ_{i-j}, où $\mathbf{1 \leq j \leq i-1}$.

2.3.2.2. Robuste contre l'attaque par rejeu :

Bien que l'attaquant rejoue δ_i à **B** dans la phase d'authentification, la demande sera rejetée, puisque δ_i n'est utilisable qu'une seule fois.

2.3.2.3. Robuste contre l'attaque par usurpation d'identité d'utilisateur :

Il est également impossible que l'attaquant puisse acquérir un secret partagé δ_0 puisqu'il est incapable de l'extraire à partir des informations qu'il obtient.

2.3.2.4. Robuste contre l'attaque par vol de table de vérification :

Bien que l'attaquant ait pu voler le vérificateur $\mathbf{T_\omega^{\tau-i}(\delta_0)}$, en ième session, $\mathbf{T_\omega^{\tau-(i+1)}(\delta_0)}$ ne peut pas être calculée par une méthode quelconque, car il est impossible de trouver l'inverse dans le SCA.

2.3.3. Les limites de l'approche

Il est clair que les concepteurs ont exigé plusieurs contraintes (voir la section 2.3.2) afin de garantir la sécurité de leur système, ce qui entrave et perturbe la scalabilité de l'approche.

Ce schéma souffre d'un problème de compteur, où leur schéma dépend de la stabilité du réseau cela n'est pas reconnu chez les concepteurs des protocoles d'authentification.

Selon les mêmes arguments indiqués dans la section (2.1.3) du protocole CARA ce schéma ne résiste pas aux attaques : server counterfeit attack et forgery attack et ne fournit ni l'authentification mutuelle ni l'accord de clé.

Chapitre IV :

Approche 1 : Construction de nouveau schéma d'authentification et d'établissement de clés à base de clé secrète partagée en utilisant les automates cellulaires unidimensionnels

Chapitre IV : Approche 1 : Construction de nouveau schéma d'authentification et d'établissement de clés à base de clé secrète partagée en utilisant les automates cellulaires unidimensionnels

Sommaire

1. Introduction

L'authentification et l'établissement de clés sont des éléments fondamentaux pour la sécurisation des communications électroniques. Les algorithmes cryptographiques pour le chiffrement et l'intégrité ne peuvent pas remplir leur fonction, sauf si des clés sécurisées seront établies et les utilisateurs savent que telles parties partagent telles clés. Il est essentiel que les protocoles d'authentification et d'établissement de clés soient adaptés à leur usage.

1.1. La problématique

Si deux communicants veulent échangés des données, il devient essentiel pour eux de s'authentifier les uns auprès des autres puis d'établir une clé commune valable pendant une session de communication. C'est une hypothèse que l'adversaire peut capturer tous les messages envoyés ou reçus. Ceci est une hypothèse réaliste dans les systèmes de communication typiques tels que l'Internet et les réseaux d'entreprise. En effet, si cette possibilité peut être écartée alors il n'y a probablement pas une nécessité d'appliquer la sécurité. Donc la confidentialité, une telle clé de session est obligatoire afin de contrecarrer contre les attaques passives.

En plus des attaques passives ; une autre hypothèse réaliste que l'adversaire peut intercepter, modifier et rejouer des messages. Donc il devient nécessaire pour les concepteurs des protocoles d'authentification de proposer des techniques afin de contrecarrer contre ces attaques actives.

Encore une fois, une hypothèse tout à fait réaliste - il est largement admis que les participants internes sont souvent plus une menace que les étrangers. Cela nous amène à une autre hypothèse de sécurité.

En plus de ces problèmes de sécurité, un autre aspect de complexité et de calcul est pris en charge afin de garantir une adaptation à des dispositifs à faible puissance de calcul tel que les cartes à puce. En outre, le nombre d'inscriptions sur les systèmes en ligne augmente de plus en plus ; augmenter le nombre de serveur et la capacité du matériel n'est pas une solution bénéfique pour les entreprises ; ce qui rend l'optimisation des protocoles d'authentification en termes de calcul, de performance et d'implémentation beaucoup plus intéressante.

1.2. La contribution

La contribution la plus attrayante de notre approche est sa mise en œuvre rapide et efficace en utilisant seulement des opérations basiques des automates cellulaires sans avoir besoin ni d'un algorithme de chiffrement ni de fonction de hachage lors de la phase d'authentification en fournissant un niveau de sécurité élevé résistant à tout type d'attaques connues.

2. Notre nouvelle conception [IV.1]

2.1. Les hypothèses de conception du protocole

Un domaine de la collision commun est un segment du réseau qui n'est pas commuté ou bridgé (i.e. connecté à travers un concentrateur : point d'accès ou hub). Toute circulation que n'est pas commutée sur un segment du réseau peut être vue par toutes les machines sur ce segment. Donc il sera facile aux attaquants de capturer potentiellement tous les paquets sur le LAN. La majorité des outils du reniflement des données est adaptée idéalement pour renifler des données dans un environnement du concentrateur. Ces outils sont appelés des renifleurs passifs comme ils attendent passivement les données envoyées afin de les capturer.

Hypothèse IV.1 : C'est une hypothèse que l'adversaire peut écouter tous les messages envoyés ou reçus. Ceci est une hypothèse réaliste dans les systèmes de communication typiques.

Une contre-mesure contre le renifleur passif est de remplacer le concentrateur du réseau par un commutateur. Contrairement à un (hub ou AP) a basé le réseau, l'Ethernet commuté ne diffuse pas toute l'information à tous les systèmes sur le LAN. Le switcher règle le courant de données entre ses ports en dirigeant activement l'adresse MAC sur chaque port qui l'aide à laisser passer seulement les données à sa cible projetée.

Cependant, il doit être noté que le développement des réseaux changés a été conduit par le besoin de plus de bande passante, et pas pour le besoin de réseaux plus sûrs.

Dans un réseau commuté (Switch), le reniflement semble impossible à réaliser puisque seul le destinataire reçoit les trames qui lui sont envoyées. Les renifleurs pour un LAN commuté injectent activement la circulation dans le réseau pour permettre renifler le trafic des données. D'où vient le terme 'renifleur actif'.

Hypothèse IV.2 : Une autre hypothèse réaliste que l'adversaire peut intercepter, modifier et rejouer des messages.

Hypothèse IV.3 : Encore une autre hypothèse tout à fait réaliste - il est largement admis que les participants internes sont souvent plus une menace que les étrangers. Cela nous amène à une autre hypothèse de sécurité.

Un scénario typique est celui où deux participants s'engagent à un protocole de clé partagée et l'un retourne un défi qui été destiné à lui-même.

Hypothèse IV.4 : Cette attaque ne peut être possible que si l'exécution parallèle du même protocole est autorisée ce qui est souvent une hypothèse réaliste.

2.2. L'architecture du protocole

Comme nous avons décrit dans le premier chapitre ; Il y a trois caractéristiques que nous considérons comme des critères architecturaux pour classer les différents protocoles d'authentification et d'établissement de clés cryptographiques : les clés cryptographiques existantes, comment la clé de session est établie et le nombre d'utilisateurs du protocole conçu pour servir.

2.2.1. Les clés cryptographiques existantes

Notez qu'il est impossible d'établir une clé de session sans canaux sécurisés existants déjà disponibles. Par conséquent, si deux participants souhaitent établir une nouvelle clé de session, il y a essentiellement trois possibilités.

1. Les participants partagent au préalable une clé secrète.

2. Un serveur hors-ligne est utilisé. Cela signifie que les participants possèdent la certification des clés publiques.

3. Un serveur en ligne est utilisé. Cela signifie que chaque participant partage une clé avec un serveur de confiance.

Nous avons opté pour le premier choix où le client et le serveur partagent au préalable une clé partagée. Nous expliquerons dans la section (2.8.1) comment la clé est partagée et transmise au client en toute sécurité.

Critère architectural IV.1 : Le client et le serveur partagent au préalable une clé secrète partagée afin de garantir l'authentification de l'entité et l'établissement de clé sécurisés.

2.2.2. Les méthodes de génération de clés de session

Il y a trois façons pour générer des clés de session dans un protocole d'établissement de clé : la première : il s'agit d'un mécanisme de transport de clé dans lequel l'un des participants génère une clé de session valable qui sera envoyée aux autres participants. La deuxième : il s'agit d'un mécanisme d'accord de clé dans lequel la clé de session est une fonction de contribution par tous les participants du protocole. Tandis que la troisième est une méthode hybride des deux premières façons.

Dans notre conception nous avons utilisé la méthode d'accord de clés ; là où le client et le serveur contribuent tous les deux pour la génération d'une clé de session valable.

Critère architectural IV.2 : La clé de session est une fonction de contribution entre le serveur distant et l'utilisateur du protocole.

2.2.3. Le nombre d'utilisateurs

A ne pas confondre avec le nombre maximal d'utilisateurs supporté par le protocole. Ici on parle de combien d'utilisateurs contribuent à la fois à la phase d'établissement de clés cryptographiques.

Dans notre protocole seul le serveur et l'utilisateur concerné contribuent à la phase de génération de la clé de session. Par conséquent, seul le serveur et l'utilisateur légitime auront connaissance de la clé de session après l'exécution du protocole.

Critère architectural IV.3 : Deux utilisateurs (le client et le serveur) contribuent à la fois pour établir une clé de session valable pour une communication sécurisée.

2.3. La fraicheur de la clé de session

Il existe deux façons pour que la fraicheur d'une valeur puisse être garantie à un utilisateur particulier. La première est la contribution de ce dernier au choix de la valeur (la clé de session dans notre cas), tandis que la seconde repose sur la réception d'une valeur connue comme fraîche.

Nous avons conçu notre protocole de telle sorte que le serveur et l'utilisateur doivent contribuer ensemble à la création d'une clé de session fraiche pour chaque exécution du protocole. Nous expliquons en détails la phase de génération de la clé de session ainsi que la fonction de contribution dans la section (2.8.2).

Une propriété souhaitable de la fonction de contribution est qu'il ne devrait être possible ni pour le serveur ni pour le client de forcer une ancienne valeur de la clé de session même si l'entrée de l'autre est connue. Cela signifie que chaque participant dispose d'une assurance indépendante de la fraîcheur de la clé de session.

Dans notre schéma, la clé de session est dérivée de la clé secrète partagée à long terme entre le serveur et le client légitime puisque la confidentialité persistante n'est pas un objectif souhaitable dans notre protocole. Cela va nous permettre d'économiser le nombre de messages transmis entre le client et le serveur et d'éviter les calculs couteux de l'exponentiation modulaire et les clés éphémères de la cryptographie asymétrique. A noter que la clé secrète partagée n'a aucun rapport avec la fraicheur de la clé de session. Par contre c'est une assurance que seuls les participants légitimes peuvent construire une clé fraiche valable pour une session de communication.

2.4. Les objectifs souhaités

Nous allons citer une liste détaillée d'objectifs souhaités par notre protocole, la liste est illustrée dans le tableau suivant :

Tableau IV. 1 - Les objectifs souhaités

- Le protocole doit assurer l'authentification de l'entité ; c.-à-d. l'une des parties (client ou serveur) doit être assurée de l'identité de la seconde impliquée dans le protocole, et que la seconde doit effectivement avoir participé.
- Le protocole doit assurer l'authentification mutuelle entre le serveur et le client ; c'est-à-dire l'objectif numéro un doit être assuré dans les deux sens.
- Le protocole doit établir une clé de session fraîche, connue seulement des participants à la session.
- Le protocole doit fournir l'intégrité des clés ; c.-à-d. la propriété que la clé n'est engendrée que par les contributions des participants légitimes (protocole d'accord de clé).
- Le protocole doit fournir la confirmation de clé du serveur au client c.-à-d. que le client doit avoir l'assurance que la clé de session K_s est une bonne clé pour communiquer avec le serveur, et que le serveur est en possession de cette clé.
- Le protocole a pour objectif de fournir la confiance mutuelle en clé K_s c.-à-d. Prévision du serveur que K_s est une bonne clé pour une utilisation avec le client, et souhaite du client de communiquer avec le serveur en utilisant la clé K_s en la croyant bonne pour cet usage.

2.5. Motivation et choix des outils

2.5.1. Pourquoi choisir les automates cellulaires comme outil mathématique ?

La raison de la popularité des automates cellulaires peut être attribuée à leur simplicité, et de l'énorme potentiel qu'ils détiennent dans la modélisation de systèmes complexes, en dépit de leur simplicité. En outre, étant donné que les opérations CA utilisent les opérateurs binaires simples qui sont exécutées indépendamment pour chaque bit, le mécanisme d'évolution basé sur les CA a un parallélisme inhérent, et fournit donc une implémentation extrêmement

rapide et efficace, en particulier sur la conception du matériel. Le système proposé ne nécessite pas de cryptage des données explicites, ni du côté du client, ni sur le serveur. Ces faits conduisent à une grande vitesse d'authentification des performances de fonctionnement en temps réel.

Les automates cellulaires peuvent être parfaitement adaptés à une étude et à une analyse détaillée, parce qu'ils peuvent être simulés facilement dans un ordinateur ou une carte à puce (un dispositif de faible capacité de calcul). Même avec un automate cellulaire unidimensionnel simple (règles Wolfram) on peut simuler des comportements complexes et imprévisibles.

2.5.2. Pourquoi une Java Card ?

La sécurité, la portabilité et la facilité d'utilisation sont les autres avantages clés des cartes à puce. La carte à puce est résistante à l'attaque car elle n'a pas besoin de dépendre des ressources externes potentiellement vulnérables. Les éléments de sécurité des cartes à puce sont encore renforcés par des fonctions cryptographiques.

En outre l'accès à une carte à puce nécessite habituellement du titulaire de la carte d'entrer le code PIN ou un mot de passe, ce qui empêche la carte de n'être utilisée que par une personne autorisée.

Les cartes à puce sont souvent utilisées pour le stockage sécurisé des données et pour authentifier et garantir les transactions. Elles ont une puissance et un impact dans le commerce et l'industrie.

Avec une carte à microprocesseur, les données ne sont jamais à la disposition directe des applications externes. Le microprocesseur gère le traitement des données et l'accès mémoire selon un ensemble donné de conditions (mot de passe, cryptage, etc.)

Ce type de carte offre une carte possédant une mémoire assez suffisante pour le stockage et comprend les deux interfaces avec et sans contact. Ceci permet l'élargir le champ d'utilisation de la carte, vu que l'interface avec contact peut être utilisée pour le contrôle d'accès logique et la deuxième interface (sans contact) pour le contrôle d'accès physique (par exemple : l'accès à un bâtiment sécurisé, laboratoire de recherche…).

Les avantages de Java Card comme plateforme de gestion d'applications se situent d'une part au niveau des développeurs et d'autre part au niveau des fabricants de cartes. En effet, au niveau des développeurs d'applications la

plateforme Java Card apporte les mêmes avantages que le Java par rapport aux autres langages de programmation existants.

Il s'agit ainsi de cartes multi-applicatives hébergeant divers programmes pouvant éventuellement communiquer entre eux. Java Card est une plateforme pour cartes à puce capable de faire fonctionner plusieurs applications d'origines diverses. Ainsi, une fois la carte remise à son utilisateur, il est potentiellement possible d'ajouter ou de retirer des applications de la carte ; ce qui facilite la mise à jour des programmes sans avoir à changer les cartes.

2.6. Que veut-on sécuriser ? Contre quels dangers ?

La confidentialité de la clé de session et les clés à long terme est obligatoire afin de contrecarrer contre les attaques passives. En plus des attaques passives : il est nécessaire pour les concepteurs des protocoles d'authentification de proposer des techniques afin de contrecarrer contre les attaques actives demandant une intervention de l'attaquant.

Le tableau suivant présente l'ensemble des attaques qu'on souhaite contrecarrer durant la conception et la construction de notre protocole ; afin de mieux éclaircir les objectifs requis de notre protocole.

Tableau IV. 2 - Liste des attaques possibles

(FR)	(EN)
Cassage de mot de passe	Password guessing attack
Vol de carte à puce et authentification forte	smart card stolen attack and strong authentication
Attaque par rejeu et la non-synchronisation de l'horloge	Replay attack and clock un-synchronization
Attaque par usurpation d'identité de l'utilisateur	User impersonating attack
Attaque par usurpation d'identité du serveur	Server counterfeit attack
Authentification mutuelle et l'attaque de l'homme du milieu	Mutual authentication and man in the middle attack
Accord de clé de session et attaque par clé connue	Session key agreement, forgery attack and known key security
Pas une table de stockage et vol de table de vérification	No verifier-table stored and stolen verifier attack
Attaque par réflexion	Reflection attack
Attaque par déni de service	Denial Of Service (DOS)

Attaque par mauvaise interprétation de message	Typing attack
Attaque par interaction des protocoles	Protocole interaction

Ces attaques vont être détaillées dans une section ultérieure (L'expérimentation des résultats).

2.7. S'identifier puis s'authentifier

Pour l'identifiant qui est une valeur non-secrète stockée clairement dans la carte à puce ; lorsque le client insère sa carte dans le lecteur, il sera automatiquement invité à taper son identifiant comme une première entrée et son mot de passe comme deuxième. La première entrée est comparée avec l'identifiant stocké sur la carte à puce ; donc l'étape d'identification s'effectue d'abord au niveau de la carte à puce (coté client). Puis la deuxième entrée entre en jeu afin d'accomplir l'étape d'authentification ; mais cette fois ci la vérification ne s'effectue pas au niveau de la carte à puce puisque le mot de passe n'est stocké nulle part dans la carte mais elle s'effectue au niveau du serveur, nous expliquons en détail cette procédure dans la section suivante (la construction du protocole).

2.8. La construction du protocole

En utilisant le mécanisme des automates cellulaires réversibles de second ordre, nous proposons un nouveau système d'authentification mutuelle des utilisateurs distants. La sécurité de notre approche est basée sur le secret (une clé) partagée entre les deux communiants (le client et le serveur), et sur la difficulté d'inverser un CA de second ordre sans la connaissance de la règle de transition correspondante.

Le concept de nonce est incorporé dans le système afin d'éliminer le problème de synchronisation de temps entre le serveur et ses clients. Le nonce est utilisé dans un scénario de défi-réponse dans lequel le serveur conserve une valeur à usage unique propre à chaque utilisateur. Le serveur envoie alors le nonce et espère le récupérer sous une forme codée correctement, sinon, le serveur refuse la demande du client.

La longueur de la clé ainsi que celle du nonce utilisée est égale à 128 bits, et peut être facilement étendue à des valeurs plus élevées ou plus faibles en raison de l'évolutivité (scalabilité) de l'approche. Nous considérons la classe d'un

automate cellulaire réversible binaire unidimensionnelle de second ordre avec un rayon de taille 3, qui gèrent 2^{128} règles de transition possibles.

Le mécanisme de l'évolution du RCA de second ordre est défini par une fonction "**Evolve-RCA**" qui prend deux configurations consécutives comme entrée et produit deux nouvelles selon deux paramètres différents, à savoir la règle de transition **R** et le nombre d'itérations **N** :

Fonction : Evolve-RCA

1. $\{0, 1\}^{128} \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128} \times \{0, 1\}^{128}$

2. $(C^t, C^{t+1}) \rightarrow (C^{t+n}, C^{t+n+1}) = \text{Evolve-RCA}\ (C^t, C^{t+1}, R, N)$

Fonction IV.1 – Le mécanisme d'évolution du RCA

Cette fonction est auto-réversible. Lorsqu'elle est appliquée sur les deux configurations (C^{t+n}, C^{t+n+1}) en utilisant la même règle de transition et le même nombre d'itérations, sa sortie génère exactement les deux configurations initiales (C^t, C^{t+1}).

L'inversibilité de cette fonction est infaisable si la règle de transition est inconnue, ou si l'une des configurations de départ est manquante. La sécurité du système repose sur une telle hypothèse.

Dans le schéma proposé, la clé secrète est utilisée comme la règle de transition afin de faire évoluer la paire de configurations (pré-initiale C^0, initiale C^1) défini respectivement par la valeur du nonce et la clé secrète. Un nombre total de 19 itérations est réalisé à partir de ces deux configurations pour obtenir la paire de configurations (pré-finale C^{19}, finale C^{20}). En utilisant la même règle et le même nombre d'itérations, les deux configurations C^0 et C^1 peuvent être récupérées à partir de C^{20} comme configuration pré-initiale et C^{19} comme configuration initiale. Les détails et les justifications de ces choix sont détaillés dans la section (2.8.2).

Le système d'authentification proposé comprend deux phases : une phase d'enregistrement, et une phase d'authentification mutuelle et d'accord de clé. La première phase est similaire à celle réalisée dans des systèmes d'authentification classiques existants, puisque la contribution principale de ce travail réside en la phase d'authentification en utilisant le mécanisme de RCA afin d'atteindre des performances optimales et une sécurité plus élevée. Les notations suivantes sont adoptées dans ce qui suit dans cette thèse :

Tableau IV. 3 - La table des notations

H() :	Une function cryptographique à sens unique
⊕ :	L'opération OU exclusive (XOR)
|| :	L'opération de concaténation de chaine
N :	La valeur du nonce
ID_i :	L'identificateur de l'utilisateur U_i
PW_i :	Le mot de passe de l'utilisateur U_i
S_x :	La clé secrète maitresse maintenue par le serveur
K_i :	La clé secrète dérivée pour l'utilisateur U_i
K_s :	La clé de session convenue

2.8.1. La phase d'enregistrement

Comme d'habitude, la phase d'enregistrement est exactement la même que celle utilisée dans les schémas classiques. On suppose un utilisateur **U_i** soumet son identité **ID_i** et son mot de passe **PW_i** à un serveur pour la phase d'enregistrement via un canal sécurisé, puis si le serveur accepte la demande, il calcule **$R_i = H(ID_i \oplus S_x) \oplus PW_i$**. Finalement une carte à puce contenant **R_i**, **ID_i** et l'implémentation de la fonction Evolve-RCA est délivrée à l'utilisateur **U_i**. La figure IV.5 illustre le mécanisme de la phase d'enregistrement.

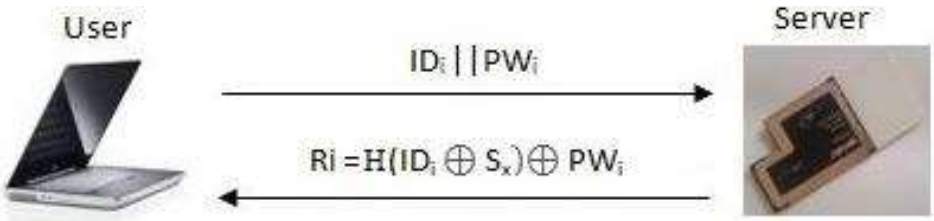


Figure IV. 1 - Les étapes de la phase d'enregistrement

2.8.2. La phase d'authentification mutuelle et d'accord de clé

Lorsque les utilisateurs enregistrés ont besoin pour se connecter au serveur, les informations de la carte à puce sont utilisées pour effectuer la phase d'authentification. Cette phase permet une authentification mutuelle entre les deux parties (le client et le serveur), de sorte que chacun vérifier l'identité de l'autre. Après la phase d'authentification, une clé de session partagée est mise en place afin de permettre une communication sécurisée entre les deux parties au cours d'une session de communication prédéterminée. Un indiscret qui a accès aux messages échangés ne devrait pas être en mesure de tirer des informations sur le secret **S_x**, ou sur la clé de session établie **K_s**.

2.8.2.1. L'authentification mutuelle

L'authentification mutuelle se déroule selon les étapes suivantes :

• M**A-1 :** Pour un utilisateur $\mathbf{U_i}$ qui souhaite se connecter au serveur, il doit insérer d'abord sa carte à puce dans un lecteur de carte, saisir ensuite son identité **IDi** et son mot de passe **PWi**. La carte à puce effectue ensuite les étapes suivantes pour commencer une session d'accès :

- Identifier l'utilisateur à partir de son $\mathbf{ID_i}$ entré ;
- Calculer $\mathbf{K_i = R_i \oplus PW_i}$.

Notez que l'utilisateur est identifié mais non pas authentifié car il n'y a aucun moyen d'effectuer l'authentification au niveau de la carte à puce (pour éviter les attaques hors-ligne permettant de deviner le mot de passe), de sorte que l'utilisateur est généralement authentifié au niveau du serveur.

L'utilisateur envoie son identité avec une demande de service au serveur, puis la clé secrète $\mathbf{K_i}$ de l'utilisateur $\mathbf{U_i}$ dérivé de son identité et la clé maîtresse secrète $\mathbf{S_x}$ (qui est maintenu et connu seulement par le serveur). La valeur de $\mathbf{K_i}$ est maintenue au niveau de la carte à puce pour être utilisée plus tard au cours du processus d'authentification ;

• M**A-2 :** Lorsque le serveur reçoit la demande du service et l'identité de l'utilisateur, il génère et envoie un nonce **N** (qui est un nombre entier sur 128 bits) à l'utilisateur comme étant un défi ;

• M**A-3 :** Lors de la réception du nonce, la carte à puce de l'utilisateur utilise la mise en œuvre de la fonction Evolve-RCA (fonction IV.1), et l'applique sur la paire $\mathbf{(C^0, C^1) = (N, K_i)}$, lorsque le nonce **N** est la configuration pré-initiale et la clé de l'utilisateur dérivée $\mathbf{K_i}$ est la configuration initiale. Avec l'utilisation de $\mathbf{K_i}$ comme une règle de transition, l'utilisateur applique 19 itérations pour obtenir deux configurations ; pré-finale C^{19} et finale C^{20}.

Un nombre aléatoire **n** est alors généré dans l'ensemble {2 ... 18} et envoyé comme un défi pour le serveur conjointement avec les deux configurations C^{19} et C^{20}. Le serveur, puis recevoir le triplet (C^{19}, C^{20}, n).

• M**A-4 :** Lors de la réception du triplet ci-dessus, le serveur calcule tout d'abord la valeur de la clé $\mathbf{K_i}$ dérivée de l'utilisateur en utilisant la formule $\mathbf{K_i = H\ (ID_i \oplus S_x)}$. Celui-ci est utilisé comme une règle de transition pour exécuter le RCA vers

l'arrière à partir des configurations C^{19} reçues et C^{20} (en utilisant C^{20} comme une configuration pré-initiale et C^{19} comme configuration initiale). Après avoir appliqué 19 itération, le serveur doit récupérer exactement les deux configurations C^0 et C^1 utilisés par l'utilisateur, à savoir le nonce qu'il envoie et la clé de l'utilisateur dérivée $\mathbf{K_i}$. Après avoir comparé les deux configurations récupérées avec **N** et $\mathbf{K_i}$, le serveur authentifie et accepte la demande de l'utilisateur en cas d'égalité, sinon, la requête de l'utilisateur est refusée.

• M**A-5 :** Afin d'effectuer une authentification mutuelle, le serveur doit également prouver son identité à l'utilisateur. Le défi reçu (défini par le nombre **n**) est utilisé par le serveur et la configuration intermédiaire $\mathbf{C^n}$ est renvoyée à l'utilisateur. Lors de la réception, cette configuration est utilisée conjointement avec son successeur $\mathbf{C^{n+1}}$ qui a déjà été mémorisé par l'utilisateur. Le RCA est itéré en arrière par l'utilisateur de n itérations, et les configurations résultantes doit correspondre exactement à (C^0, C^1) égales respectivement à **N** et $\mathbf{K_i}$. Si l'égalité est vérifiée, alors le serveur est authentifié. Dans le cas contraire, le serveur est considéré comme intrus, et la connexion est interrompue. La figure IV.6 illustre le schéma entièrement détaillé de la phase d'authentification mutuelle.

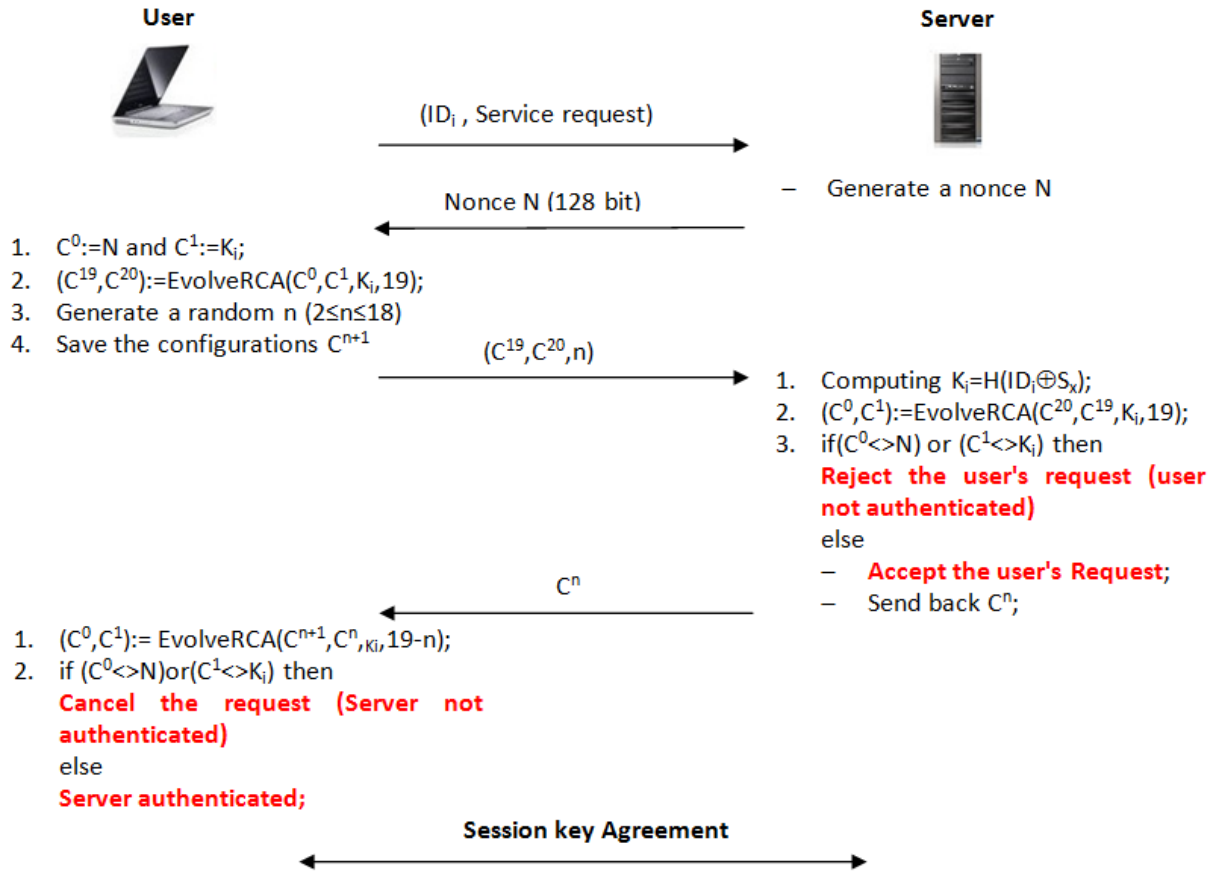


Figure IV. 2 - Diagramme d'authentification mutuelle proposé

2.8.2.2. L'accord de clé

Après que l'authentification mutuelle soit effectuée avec succès, les deux parties doivent se mettre d'accord sur une clé de session partagée qui est utilisée pour la communication sécurisée lors de la session en cours. La clé de session $\mathbf{K_s}$ est dérivée des messages échangés lors de la phase d'authentification, à savoir le nonce **N**, la clé $\mathbf{K_i}$ de l'utilisateur dérivé et la clé maitresse partagée $\mathbf{S_x}$. Un espion qui recueille tous les messages échangés est incapable d'en déduire toutes les informations relatives à la clé de session. En utilisant le mécanisme du RCA second ordre, la clé de session est dérivée selon le schéma suivant :

• K**A-1** : Le ième octet est extrait de chaque configuration $\mathbf{C_i}$ **(i = 0 ... 20)** qui est générée au cours de l'évolution du RCA dans les deux côtés utilisateur et serveur. Supposons ces octets sont notés $\mathbf{b_i}$ de (i = 0 ... 20) et sont définis par $\mathbf{b_i = C_i}$ **[i]** lorsque on considère chaque configuration comme un tableau d'octets.

• K**A-2** : Alors que la clé de session a une longueur de 128 bits (16 octets), seuls 16 octets sont nécessaires à partir du total de 21 configurations disponibles. Nous écartons les octets émis des configurations C^0 et C^1 pour la construction de la clé de session depuis le premier est publiquement connue (correspond au

nonce **N**), et le second révèle des informations sur la clé de l'utilisateur dérivée $\mathbf{K_i}$. En outre, les octets émis à partir des deux configurations C^{19} et C^{20} sont rejetés car ils sont transmis en clair et sont donc connus au public. L'octet issu de la configuration C^n (ce qui correspond au défi présenté par l'utilisateur au serveur) est également écarté pour les mêmes raisons. Par conséquent, l'ensemble de 16 octets qui peuvent être utilisés pour obtenir la clé de session est défini par $\{b_2, b_3, ..., b_{n-1}, b_{n+1}, ... b_{18}\}$ qui sont obtenus respectivement à partir des configurations $\{C^2, C^3, ..., C^{n-1}, C^{n+1}, ... C^{18}\}$.

• K**A-3** : Les octets $\{b_2, b_3, ..., b_{n-1}, b_{n+1}, ... b_{18}\}$ sont ensuite concaténés et le résultat est haché en utilisant la fonction **H** pour produire la clé de session comme ce qui suit :

$$\boldsymbol{Ks = H\,(b2\,||\,b3||\,...||\,bn-1||\,bn+1||\,...\,b18)\,\ldots\,(IV.12)}$$

A noter que, puisque les mêmes configurations et paramètres sont utilisés par les deux parties, la même clé de session est finalement dérivée pour les deux parties. En outre, un indiscret qui recueille tous les messages échangés, est incapable de tirer des informations utiles sur la clé de session car la réversibilité du mécanisme de RCA est infaisable sans la connaissance de la règle de transition (qui est définie par la clé K_i de l'utilisateur secret). La figure IV.7 illustre les détails du système de dérivation de clé de session qui est exécutée par les deux parties.

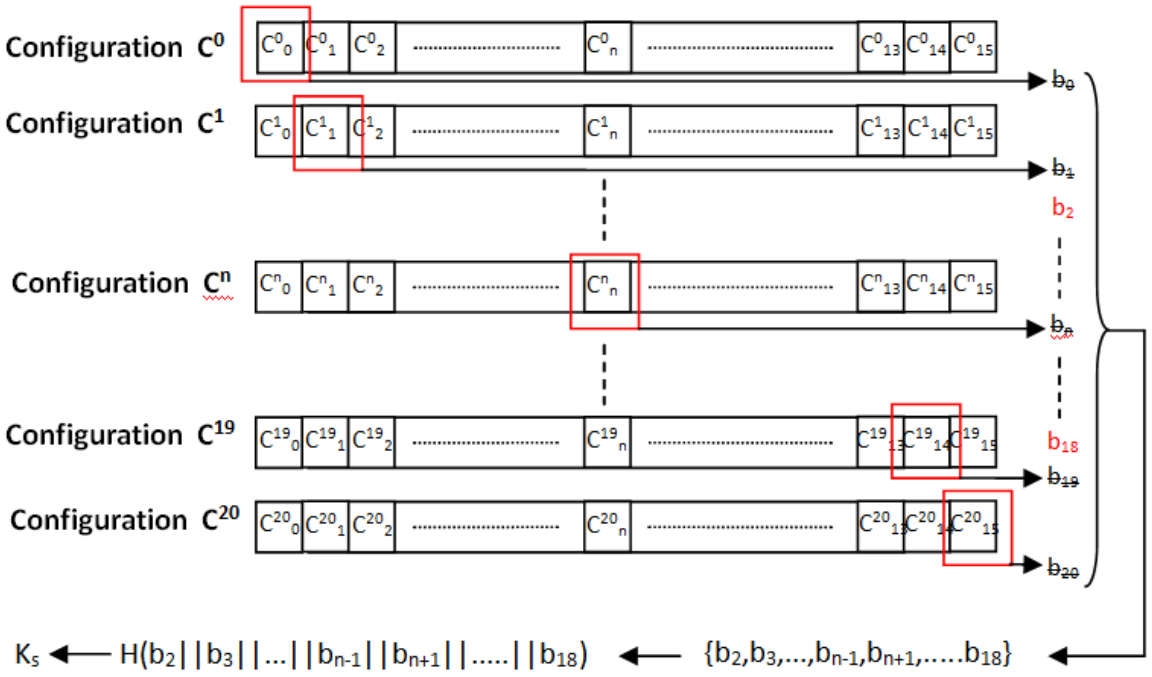


Figure IV. 3 - Les illustrations du mécanisme de dérivation de la clé de session

2.9. L'expérimentation des résultats

Dans ce qui suit, nous discutons des propriétés de sécurité du schéma proposé par rapport aux attaques potentielles communes. Il est supposé que l'attaquant a plein potentiel pour écouter le canal de communication entre l'utilisateur et le serveur et peut modifier tout message échangé. Un tableau comparatif est également présenté dans la section qui suit entre le schéma proposé et certains schémas d'authentification existants à base des CA.

2.9.1. Cassage de mot de passe

Dans le schéma proposé, le mot de passe de l'utilisateur $\mathbf{PW_i}$ est utilisé pour libérer la clé secrète $\mathbf{K_i = R_i \oplus PW_i}$ où $\mathbf{R_i = h\ (ID_i \oplus S_x) \oplus PW_i}$. Alors que la clé secrète $\mathbf{S_x}$ du serveur est protégée par une fonction de hachage sécurisée à sens unique, il est mathématiquement impossible de dériver $\mathbf{S_x}$ depuis la valeur de $\mathbf{h\ (ID_i \oplus S_x)}$. En outre, le mot de passe est transmis entre l'utilisateur et le serveur à travers l'authentification, mais est seulement impliqué en utilisant le Nonce.

Par conséquent, l'attaquant est incapable d'extraire des informations à propos de mot de passe à partir des identifiants et des messages écoutés transmis entre l'utilisateur et le serveur, donc il n'y a aucun moyen de monter une attaque hors ligne permettant de deviner le mot de passe (off-line password guessing attack). Enfin, pour résister contre une attaque en ligne (on-line password guessing attack), le système limite la quantité des tentatives jusqu'à un nombre limité de fois.

2.9.2. Vol de carte à puce et authentification forte

En ce qui concerne la sécurité, l'authentification basée sur un seul facteur d'authentification dépend uniquement de la connaissance de l'utilisateur d'un secret (par exemple un mot de passe) n'est pas assez sécurisée. L'authentification à deux facteurs est une meilleure option en utilisant le mot de passe comme (quelque chose que vous savez) et la carte à puce comme l'autre facteur (quelque chose que vous avez). Lorsque l'attaquant obtient la carte à puce, il peut savoir $\mathbf{h\ ()}$, $\mathbf{R_i = h\ (ID_i \oplus S_x) \oplus PW_i}$ et $\mathbf{ID_i}$ qui sont stockés dans la carte à puce. Cependant, il ne peut pas constituer un message de connexion légal, parce qu'il ne peut pas libérer la clé secrète sans connaître le mot de passe correct.

2.9.3. Attaque par rejeu et la non-synchronisation de l'horloge

Le système d'authentification à base de timestamp peut souffrir de l'attaque de rejeu comme le délai de transmission est imprévisible dans un environnement de réseau. Pour cette raison, nous utilisons un modèle à base des nonces au lieu d'une conception basée sur l'horodatage (timestamp) pour le schéma proposé, de sorte que le système peut prévenir un tel problème grave de la non-synchronisation de l'horloge.

Au cours de la phase d'authentification, un adversaire peut prétendre être un utilisateur valide et tente de se connecter au serveur en lui envoyant des messages qui ont été précédemment transmis par un utilisateur légal. Pour éviter une telle attaque, le serveur génère un nonce aléatoire, imprévisible et différent pour chaque session, qui rend les messages échangés d'une session donnée valable uniquement pour cette session, et ne pouvaient pas être réutilisé dans d'autres sessions.

Par conséquent, un adversaire ne peut pas passer le processus de vérification auprès du serveur avec des valeurs légitimes précédemment écoutées.

De toute évidence, l'attaque de rejeu échoue. Par conséquent, le schéma proposé peut résister à une attaque de rejeu en raison de l'utilisation de la conception basée sur le nonce.

2.9.4. Attaque par usurpation d'identité de l'utilisateur

Les attaques par usurpation d'identité veulent tout simplement dire que les utilisateurs malveillants essaient de forger un message de connexion valide pour réussir l'authentification auprès du serveur sans avoir la clé secrète de l'utilisateur légitime. Bien qu'un imposteur malveillant $\mathbf{U_z}$ d'un utilisateur $\mathbf{U_i}$ peut intercepter des messages de demande de connexion d'une session ultérieure (La configuration pré-finale et finale), mais ces valeurs sont inutiles pour la prochaine connexion en raison de la propriété (one time valid propriety). En outre, $\mathbf{U_z}$ échoue dans sa tentative de tirer un nouveau message valable de demande de connexion, car il n'a pas la clé secrète $\mathbf{K_i}$ de l'utilisateur $\mathbf{U_i}$, donc $\mathbf{U_z}$ ne peut pas terminer la deuxième étape du protocole. La résistance contre l'attaque par usurpation d'identité de l'utilisateur est garantie par le protocole proposé.

2.9.5. Attaque par usurpation d'identité du serveur

Supposons qu'un adversaire $\mathbf{S_z}$ qui veut se faire passer comme un serveur légal S pour tromper un utilisateur $\mathbf{U_i}$. Il doit calculer une configuration de réponse valide correspondant à un numéro de défi valide (configuration de défi). Cependant, $\mathbf{S_z}$ échoue dans sa tentative de tirer un nouveau message de réponse valide, parce qu'il ne possède pas la clé secrète $\mathbf{K_i}$ de l'utilisateur $\mathbf{U_i}$. Par conséquent $\mathbf{S_z}$ ne peut pas terminer la troisième étape du protocole. De plus, pour calculer une clé de session, la clé secrète $\mathbf{K_i}$ est obligatoire. Par conséquent, l'attaquant n'a aucun moyen de se faire passer comme serveur légitime.

2.9.6. Authentification mutuelle et l'attaque de l'homme du milieu

Il est une exigence de base d'un système d'authentification que l'utilisateur et le serveur s'authentifient l'un auprès de l'autre. Dans le schéma proposé, le serveur authentifie l'utilisateur en vérifiant le message de demande de connexion en utilisant le mécanisme de réversibilité des automates cellulaires de second ordre. Si le serveur arrive à extraire le nonce à partir du message de demande de connexion, le serveur confirme la légitimité de l'utilisateur. Le serveur répond à l'utilisateur avec un message spécifique (configuration qui correspond au défit), qui est utilisée avec sa précédente afin d'authentifier le serveur. Si les deux configurations sont compatibles, alors l'utilisateur confirme que le serveur est légitime.

Par conséquent, l'authentification mutuelle est garantie par les étapes 3 et 4 du protocole. Selon les sections 6.4 et 6.5, nous savons que l'attaque par usurpation d'identité de l'utilisateur et l'attaque de faux serveur ne peuvent être invoquées par un utilisateur malveillant $\mathbf{U_z}$. Par conséquent, il est impossible pour $\mathbf{U_z}$ de mener une attaque man-in-the-middle. Il est bien évident que le protocole est résistant à l'attaque de l'homme du milieu et l'authentification mutuelle est bien atteinte.

2.9.7. Accord sur la clé de session et l'attaque par clé connue

Après une authentification mutuelle réussie, une clé de session est construite et partagée enfin entre le client et le serveur pour être utilisée pendant le reste de la session de communication. Notre protocole garantit que la clé de session est seulement connue par les entités légitimes (le client et le serveur) ; par conséquent, aucun tiers ne peut envoyer un message forgé au client ou au serveur. Étant donné que le nombre aléatoire (nonce) est différent pour chaque

session, alors même si une clé de session donnée est compromise, ni la clé secrète ni les clés de sessions (passés et futurs) ne seront compromises. Par conséquent, le schéma proposé assure (known key security).

2.9.8. Vol de table de vérification

Comme aucune table de vérification n'est stockée sur le serveur distant, le système résiste à l'attaque de modification et assure la réduction du coût de stockage. Le serveur distant utilise la clé secrète $\mathbf{K_i}$ pour authentifier l'utilisateur légitime $\mathbf{U_i}$ par la propriété de réversibilité sécurisée des automates cellulaires. Par conséquent, le système proposé est sécurisé contre (stolen verifier attack).

2.9.9. Attaque par réflexion

Une attaque par réflexion ne peut être effectuée si et seulement si des exécutions parallèles du même protocole sont autorisées. Par exemple, si un participant est un hôte Internet, il peut accepter des sessions de plusieurs autres participants.

3. Analyse comparative

Afin d'illustrer les avantages du système proposé, une étude comparative a été réalisée par rapport à plusieurs schémas d'authentification existants basée sur les CA. Il est clair que l'approche proposée assure toutes les propriétés énumérées tout en nécessitant moins de coût de calcul.

3.1. Analyse de sécurité

La comparaison est effectuée en termes de propriétés de sécurité et de performances d'implémentation. Le tableau IV.4 présente plusieurs aspects de sécurité assurée par le schéma proposé par rapport à la plupart des schémas d'authentification à base des CA-base connus proposées dans [III.10], [III.11] et [III.12].

Tableau IV. 4 - comparaison des aspects de sécurité à l'égard des systèmes d'authentification connus à base des CA

SECURITY CRITERION	CARA [III.10]	CARMA [III.11]	JOEN ET AL. [III.12]	PROPOSED [IV.1]
RESISTANCE TO REPLAY ATTACK	Yes	Yes	Yes	Yes
USER IMPERSONATING ATTACK	Yes	Yes	Yes	Yes
SERVER COUNTERFEIT ATTACK	No	Yes	No	Yes
MUTUAL AUTHENTICATION	No	Yes	No	Yes
RESISTANCE TO STOLEN VERIFIER ATTACK	Yes	No	Yes	Yes
RESISTANCE TO MAN IN THE MIDDLE ATTACK	No	Yes	No	Yes
RESISTANCE TO FORGERY ATTACK	No	Yes	No	Yes
SESSION KEY AGREEMENT	No	Yes	No	Yes
OFFLINE-PASSWORD GUESSING	No	No	Yes	Yes
ONLINE-PASSWORD GUESSING	No	No	Yes	Yes
PROVIDING STRONG AUTHENTICATION	Yes	No	No	Yes

3.2. Analyse d'implémentation

En outre, étant donné que les opérations CA utilisent les opérateurs binaires simples qui sont exécutées indépendamment pour chaque bit, le mécanisme d'évolution basé sur les CA a un parallélisme inhérent, et fournit donc une implémentation extrêmement rapide et efficace, en particulier sur la conception du matériel. Le système proposé ne nécessite pas de cryptage des données explicites, ni du côté du client, ni sur le serveur. Cela empêche également les attaques par déni de service basées sur la surcharge du serveur avec des demandes de service parasites.

Ces faits conduisent à une grande vitesse d'authentification des performances de fonctionnement en temps réel. En utilisant une implémentation logicielle à la fois de l'approche proposée et celles qui existent déjà, le tableau IV.5 des performances énumère les performances d'exécution mesurées pour 100 authentifications consécutives réalisées pour simuler une session d'authentification (utilisateur / serveur). Il est clair que celui proposé effectue plus rapidement que ceux qui existent déjà, tout en offrant un niveau de sécurité plus élevé.

Tableau IV. 5 - Comparaison des performances d'exécution mesurée pour 100 authentifications consécutives

AUTHENTICATION SCHEMES	RUNTIME PERFORMANCES (FOR 100 AUTH.) IN MS
CARA [III.10]	78
CARMA [III.11]	67
JEON ET AL. [III.12]	56
PROPOSED [IV.1]	22

Un autre tableau comparatif IV.6 de performance montre que l'approche proposée prend avantage en termes de coût de communication et de calcul.

Tableau IV. 6 - Comparaison des performances

	CARA [III.10]	CARMA [III.11]	JEON ET AL. [III.12]	PROPOSED [IV.1]
INSTANCE OF HACH FUNCTIONS	2	5	0	0
VERIFICATION TABLE	No	Yes	No	No
SYNCHRONIZATION REQUIREMENT	No	No	No	No
COMPUTATION & COMMUNICATION COST	Low	Low	Low	Extremely Low

Partie III : La deuxième approche

Chapitre V :

Etat de l’art sur l’échange de clés authentifié à base de mot de passe (PAKE) en utilisant les automates cellulaires bidimensionnels

Chapitre V : Etat de l'art sur l'échange de clés authentifié à base de mot de passe (PAKE) en utilisant les automates cellulaires bidimensionnels

Sommaire

1. Introduction

L'échange de clés authentifié à base de mot de passe (PAKE) permet de partager de courts mots de passe entre deux utilisateurs et se mettre d'accord sur une clé de session cryptographiquement forte.

Les premiers protocoles à base de mot de passe ne sont apparus qu'en 1989, ces premiers protocoles. Lomas et al. [I.35] ont utilisé l'hypothèse supplémentaire que le client a connaissance de la clé publique du serveur, en plus le mot de passe partagé avec le serveur. Plus tard Bellovin et Merritt [I.36] ont introduit une classe de protocoles EKE (Encrypted Key Exchange) qui ne nécessitent pas cette hypothèse.

Il y a eu de nombreuses variantes de protocole d'EKE de Bellovin et Merritt, ainsi que de nombreux autres. La majorité des protocoles proposés utilisent l'accord de clé basée sur Diffie-Hellman avec le mot de passe partagé utilisé à des fins d'authentification.

Le protocole EKE original ne précise pas l'algorithme de chiffrement à utiliser ou comment transformer le mot de passe à une clé correspondante.

Récemment quelques protocoles d'authentification à base de mot de passe ont été implémentés par les automates cellulaires puisque ces derniers ont été acceptés comme un bon modèle de calcul, et ont été utilisés dans divers domaines, mais peu utilisés dans les systèmes d'authentification.

La première tentative a été faite par DK Bhattacharryya & S Nandi [V.1] qui ont proposé un protocole d'authentification permettant à deux parties qui partagent un petit mot de passe de générer une clé cryptographique commune en utilisant de simples opérations d'automates cellulaires afin de simplifier la structure logique. Mais ce schéma utilise de nombreuses instances de fonction de hachage, ne fournit pas la confidentialité persistante et n'est pas robuste contre l'attaque sur les clés de sessions. De plus, le protocole ne peut être exécuté que par la présence d'une clé secrète partagée à long termes.

Dans [V.2] S Tripathy & S Nandi (2006) ont proposé un protocole d'authentification à base de mot de passe en utilisant les automates cellulaires réversibles, ce protocole fournit l'authentification mutuelle et l'accord de clé sur un canal non sécurisé, il est aussi robuste contre les attaques passives et actives tout en offrant la confidentialité persistante. Mais cela n'est effectué que par

l'ajout de plusieurs instances de fonction de hachage augmentant les frais généraux de calcul.

Finalement, une authentification basée sur le CA-sécurisé et 2-parties du protocole d'échange de clés a été proposé dans [V.3] en utilisant les caractéristiques de groupe et non-groupe des automates cellulaires. La sécurité du protocole se base principalement sur le chiffrement asymétrique, où chaque participant possède une paire de clé certifiée. Le protocole dépend d’une infrastructure PKI, c.-à-d. une gestion complète de clés (demande de certification, vérification de validité des certificats…).

2. Etat de l'art

Peu de protocoles ont été proposés sur l'authentification à base de mot de passe (PAKE) en utilisant les automates cellulaires nous citons dans cette section les travaux qui ont été publiés dans ce contexte avec une analyse en terme de sécurité et d'implémentation ; nous montrons aussi les limites de chaque approche.

2.1. L'approche CPAKE : DK Bhattacharryya & S Nandi (2000) [V.1]

Cette approche présente une méthode d'échange de clé authentifiée en se basant sur un mot de passe sécurisé, conçue en fonction des caractéristiques de la linéarité et la non-linéarité des automates cellulaires. Il prend l'avantage de plusieurs schémas en fournissant l'authentification et l'établissement de clés sur un canal non sécurisé en utilisant essentiellement un petit mot de passe. En raison de la structure logique régulière, modulaire et en cascade des CA, la méthode peut être trouvée significative et attractive par rapport aux autres méthodes.

2.1.1. Les détails de fonctionnement du CPAKE

Le projet CPAKE fonctionne en deux grandes phases : La première phase d'établissement de la clé de session secrète $\mathbf{K_i}$, par l'échange des valeurs calculées intermédiaires $\mathbf{Q_i}$; Et la seconde phase, les deux participants confirment mutuellement la connaissance de la clé $\mathbf{K_i}$ avant de l'utiliser comme une clé de session.

Chacune de ces phases fonctionne selon une séquence d'étapes. La phase d'établissement des clés est mise en œuvre en deux étapes principales et chaque étape suit plusieurs sous-étapes de traitement.

Dans la première étape, **A** et **B** utilisent un petit mot de passe partagé **S** de taille (**r-bits**) comme entrée d'une fonction de permutation d'expansion **f (S)**, qui la dilate en **S'** de taille (**2r-bit**). **S'** est ensuite entré dans un schéma de hachage unidirectionnel **H** qui fonctionne en itération, pour générer des empreintes de **r-bits**. Dans chaque cycle, une fonction de XOR prend en entrée l'empreinte générée $\mathbf{h_i}$ et un motif aléatoire $\mathbf{p_i}$, généré par un générateur de motif pseudo-aléatoire **R**. La sortie est appliquée à la fonction de hachage **H** pour le cycle suivant et elle se répète pour, par exemple, **n = 4** fois. Enfin, la quatrième empreinte $\mathbf{h_4}$ est concaténée et mixée en utilisant la fonction de mixage **ɳ** pour générer la clé intermédiaire **Q**. Alors **A** communique sa clé intermédiaire, $\mathbf{Q_A}$ à **B**,

à l'autre extrémité également, **B** suivra le protocole similaire pour générer $\mathbf{Q_B}$ et la communiquer à **A**.

La deuxième étape calcule la clé de session secrète sur la base des valeurs de $\mathbf{Q_A}$ et $\mathbf{Q_B}$ et une petite clé secrète (**r-bits**) partagée **m**. Elle s'effectue en deux étapes : dans la première étape, **A** et **B** utilisent les clés intermédiaires $\mathbf{Q_A}$ et $\mathbf{Q_B}$ qu'elles mélangent en utilisant la fonction de mixage **ɳ** et la sortie est soumise à la fonction de hachage **H** pour obtenir les valeurs de hachage correspondantes $\mathbf{Q_{AB}}$ de taille (**r-bits**). Ensuite, le petit secret **m** est utilisé avec $\mathbf{Q_{AB}}$ en utilisant la fonction **ɳ** pour générer la clé de session finale **K** de (**r + m**)-bits.

Les instructions logiques pour les opérations de la première phase pour établir la clé de session sont présentées ci-dessous. Les différentes notations ainsi que leurs significations utilisées dans ce schéma sont tabulées dans le tableau V.1.

Tableau V. 1 - Les notations utilisées et leurs significations

S	Un petit mot de passe (r-bits) partagé entre A et B.
f(S)	Une fonction de permutation pour développer S en un 2r-bits S'.
H	Une fonction de hachage unidirectionnel.
m	Une petite clé secrète (r-bits) partagé entre A et B.
A → B : Υ_A	A envoie la valeur Υ_A à B.
p_i	Un motif aléatoire de (r-bits) généré en utilisant un générateur.
ɳ (w, z)	La fonction de mixage.
K	La clé de session générée de taille de (2r-bits).

Dans la deuxième phase, A et B confirment mutuellement la connaissance de **K** avant de procéder à son utilisation comme clé de session. À des fins de confirmation, les auteurs utilisent la méthode traditionnelle de confirmation de clé, comme on la trouve dans [V.5]. Pour confirmer, chacun de **A** et **B** communique à l'autre un nombre choisi aléatoirement $\mathbf{C_A}$ et $\mathbf{C_B}$ respectivement. Les étapes de traitement pour chaque phase sont notées ci-dessous.

2.1.1.1. Phase I : calcul de la clé de session secrète

S1- A calcule : $h_i \leftarrow H(f(S))$

$h_{i'} \leftarrow H(h_i \oplus p_i)$

…

$Q_A \leftarrow ɳ(h_i, h_{i'}, h_{i''}, h_{i'''})$

$A \rightarrow B : Q_A$

S2- B calcule : $h_i \leftarrow H(f(S))$

$h_{i'} \leftarrow H(h_i \oplus p_i)$

…

$Q_B \leftarrow \eta(h_i, h_{i'}, h_{i''}, h_{i'''})$

$B \rightarrow A : Q_B$

S3- A calcule : $K \leftarrow \eta(H(\eta(Q_A, Q_B)), m)$

S4- B calcule : $K \leftarrow \eta(H(\eta(Q_A, Q_B)), m)$

2.1.1.2. Phase II : confirmation mutuelle de la connaissance de la clé de session

S5- A choisit un nombre C_A aléatoire et calcule : $\Upsilon_A \leftarrow T^{r1}(C_A)$.

$A \rightarrow B : \Upsilon_A$

S6- B choisit un nombre C_B aléatoire et calcule : $\Upsilon_B \leftarrow T^{r2}(C_B)$.

$B \rightarrow A : \Upsilon_B$

S7- A vérifie si Υ_B est un attracteur de T ou pas.

S8- B vérifie si Υ_A est un attracteur de T ou pas.

Pour une confirmation plus concrète, CPAKE peut utiliser en plus un schéma de chiffrement symétrique de clé secrète sur Υ_A et Υ_B avant la communication.

Les différents modules et fonctions utilisés sont décrits brièvement.

La fonction d'expansion : *f(S)*

Cette opération est similaire à l'opération E-Box découverte dans DES [V.7]. Les r-bits de S sont étendus à S' 2r-bits par une permutation d'expansion. Cette opération a principalement deux objectifs : elle rend le résultat de la même taille (c'est-à-dire 2r-bits) pour la prochaine étape de fonctionnement, et elle fournit un résultat plus long qui peut être compressé pendant l'opération de hachage. En raison de cette expansion, la dépendance des bits de sortie sur les bits d'entrée se propage plus rapidement.

La fonction de hachage unidirectionnel basé sur les CA : *H ()*

Le CPAKE proposé utilise un schéma de hachage fondé sur les CA, conçu en fonction des propriétés de non-linéarité et non-groupe des CA. Le schéma hache les messages 2r-bits en une valeur de hachage r-bits en mode de retour. Il utilise les caractéristiques de non linéarité des CA pour la génération de l'empreinte. Une description de conception de détail du schéma de hachage peut être trouvée dans [V.6].

Dans la section suivante, une analyse contre les différentes attaques connues et inconnues des intrus est présentée.

2.1.2. L'analyse du CPAKE

Le schéma proposé est conçu avec un échange de clés intégré soutenu par une disposition d'authentification mutuelle. Il prouve suffisamment à chacune des deux parties que l'autre connaît le mot de passe. Le système vise à générer une clé de session pour sécuriser une session authentifiée ultérieure entre les parties. La sécurité du dispositif contre les différentes attaques possibles est maintenant discutée :

2.1.2.1. L'attaque par dictionnaire

Tous les mots de passe sont vulnérables à l'attaque de dictionnaire - si une opportunité est donnée. Ainsi, le travail principal du concepteur est de supprimer le maximum des possibilités. L'attaque par dictionnaire peut être effectuée en ligne ou hors ligne. Dans le cas du schéma proposé, l'attaque de dictionnaire en ligne peut être facilement détectée. Toutefois, les attaques par dictionnaire hors ligne sont de type complexe et elles doivent être manipulées avec soin.

On peut faire cette attaque en se présentant comme une partie légitime pour recueillir des informations, ou par celui qui surveille les messages entre deux parties au cours d'un échange valide légitime. Une très petite fuite d'information pendant un échange peut être exploitée.

Dans le cas de la permutation d'expansion, en permettant à chaque bit du mot de passe secret, à savoir **S** de subir cette permutation d'expansion, l'effet d'avalanche peut être augmenté, car la dépendance des bits de sortie sur les bits d'entrée a sensiblement augmenté.

2.1.3. Les limites et les points faibles du CPAKE

2.1.3.1. Attaque par vol de clé de session

Ce protocole n'est pas résistant à l'attaque de vol de clé de session ; si l'attaquant obtient la clé $\mathbf{K_i}$ par hasard, **m** peut être déterminé en utilisant les informations

entendues ($\mathbf{Q_A}$, $\mathbf{Q_B}$) pendant la phase 1, cela peut casser entièrement le protocole. D'après tout, ce schéma ne dépend pas de la fonction de hachage puisque **p** n'est pas nécessaire pour le calcul de la clé de session.

2.1.3.2. Attaque par clé connue et le manque de la confidentialité persistante

La divulgation de **m** est suffisante pour dévoiler les conversations antécédentes et postérieures. Car les secrets intermédiaires ($\mathbf{Q_A}$, $\mathbf{Q_B}$) pour chaque session sont envoyés en clair sur le réseau. Par conséquent, le protocole ne fournit aucune garantie de la confidentialité persistante.

2.1.3.3. Les frais de calcul

Les limites de ce schéma ne se limitent pas seulement à l'aspect de sécurité mais aussi en termes de complexité de calcul. Ce schéma utilise de multiples instances de fonction de hachage à base des CA, cela augmente le temps d'exécution du protocole et dégrade les performances d'implémentation surtout lorsqu'il s'agit d'un dispositif de faible calcul. Une étude comparative effectuée au prochain chapitre illustre les lacunes d'implémentation de cette approche.

2.2. L'approche RPAKE : S Tripathy & S Nandi (2006) [V.2]

Dans cette section, un protocole PAKE basé sur les RCA appelé RPAKE est proposé. Les notations utilisées sont les mêmes que celles du schéma précédent.

2.2.1. Les détails de fonctionnement du RPAKE

Ce protocole fonctionne en deux phases ; La première phase génère la clé de session $\mathbf{K_i}$ suivie d'une deuxième phase, ce qui confirme la connaissance de $\mathbf{K_i}$. La description détaillée de la première phase de ce protocole proposé est expliquée ci-dessous.

S1- Alice génère un automate cellulaire réversible (**A**) et son inverse ($\mathbf{A^{-1}}$). Ensuite, il génère un motif aléatoire $\mathbf{R_A}$ et calcule (l'identité pour la ième session) $\mathbf{OID_i = H\ (A, R_A, A, K_{i-1})}$. Il envoie $\mathbf{<ID_A, R_A, A, OID_i>}$ à Bob, où $\mathbf{ID_A}$ est l'identité d'Alice et $\mathbf{K_{i-1}}$ est la clé de la session précédente.

S2- Bob vérifie l'authenticité d'Alice en vérifiant $\mathbf{OID_i}$ et rejette la connexion si elle n'est pas satisfaite. Sinon, il calcule (2r-bit) ***P = f (p)***. Puis il génère un motif aléatoire $\mathbf{R_B}$ et calcule $\mathbf{h_b = H\ (P, R_A)}$; $\mathbf{Q = A\ (R_B) \oplus h_b}$. Enfin, Bob envoie ***<Q>*** à Alice.

S3- Alice calcule $\mathbf{h_b'}$, en utilisant le mot de passe **p** et détermine $\mathbf{R_B}$ de **Q** comme suit :

$\mathbf{P = f(p)}$;

$\mathbf{h_b = H(P, R_A)}$;

$\mathbf{Q' = Q \oplus h_b'}$;

$\mathbf{R_B = A^{-1}(Q')}$;

S4- Alice et Bob calculent tous les deux la clé de session en utilisant une fonction de mixage $\boldsymbol{\eta}$ **(.)** ; $\mathbf{K = \eta\ (h_b, H(R_B, P))}$. Ici, une fonction de hachage peut être utilisée à la place de $\boldsymbol{\eta}$ pour obtenir une meilleure sécurité.

Avant d'utiliser la clé de session $\mathbf{K_i}$, Alice et Bob doivent se confirmer mutuellement la connaissance de $\mathbf{K_i}$. Pour la confirmation, il est possible d'utiliser une méthode traditionnelle de confirmation des clés. L'ensemble du schéma est représenté dans la figure V.1.

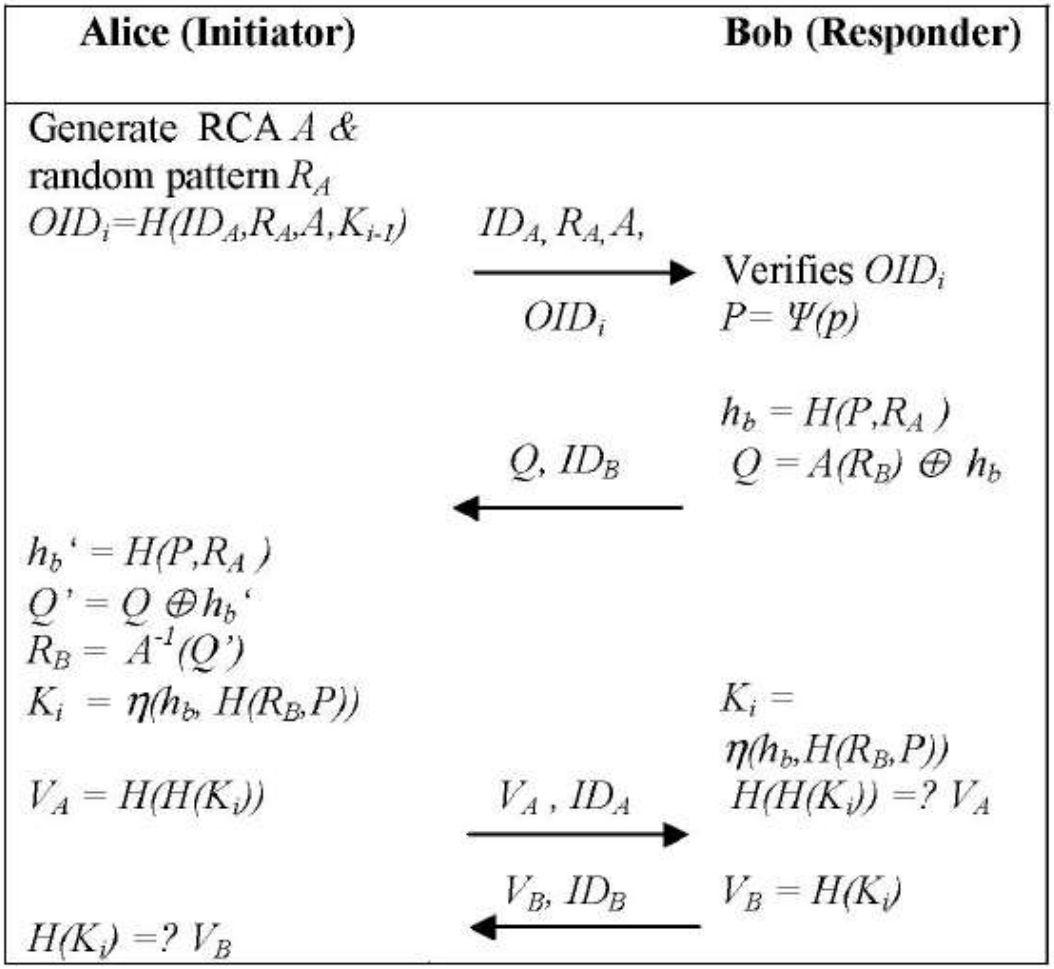


Figure V. 1 - RPAKE: Le protocole proposé

2.2.2. L'analyse du RPAKE

L'efficacité d'un système d'échange de clés est jugée par le coût de réalisation matériel ou logicielle de ce schéma et son invulnérabilité.

2.2.2.1. L'analyse d'implémentation et complexité

L'analyse de la complexité de la mise en œuvre de programmes d'échange de clés inclut à la fois les frais généraux de calcul et de communication. Ainsi le schéma (RPAKE) est comparé avec quelques protocoles existants bien connus comme SPEKE [V.8], DHEKE [V.8], PAK [V.9], SNAPI-X [V.10]. Le résultat de cette comparaison est présenté dans le tableau V.2.

Tableau V. 2 - Comparaison de la complexité computationnelle

	Schémas				
	RPAKE	SPEKE	DHEKE	PAK	SNAPI-X
#M	2	2	2	3	4
#R	2	2	2	2	3
#H	6	2	2	8	7
#E	0	4	4	6	4
#SE	0	0	2	0	0
#AE	2	0	0	0	2

Les notations suivantes sont utilisées dans le tableau :

- #M : Nombre de flux de messages
- #R : Nombre de nombres aléatoires
- #H : Instances des fonctions de hachage
- #E : Instances d'exponentiation modulaire.
- #SE : Instances de cryptage symétriques.
- #AE : Instances de cryptage asymétrique.

Le tableau en dessus montre la supériorité du schéma (RPAKE) qui n'utilise aucune opération d'exponentiation modulaire et le protocole s'exécute avec un nombre minimum de flux de messages échangés. En dehors de cela, RPAKE utilise les opérations basées sur RCA (cryptage / décryptage asymétrique). La localité inhérente et le parallélisme de CA simplifient la réalisation matérielle et logicielle ainsi et rend ainsi la mise en œuvre plus rapide.

2.2.2.2. L'analyse de sécurité

RPAKE est conçu avec un échange de clés intégré soutenu par l'authentification mutuelle. Il partage un petit mot de passe (r= 64) bits. Dans ce schéma, un

adversaire peut lire les messages produits par les parties, fournir des messages propres aux parties légitimes, modifier les messages avant d'atteindre la destination, retarder les messages ou les rejouer et faire de nouvelles instances de toutes les parties. Le succès de l'adversaire signifie qu'il doit avoir trouvé une clé de session ou un mot de passe. La sécurité de RPAKE est analysée en fonction des différentes attaques, sur la base des propriétés suivantes.

1. Il est supposé que le paramètre de sécurité de la clé de session **K**, est assez long pour être immunisé contre une attaque de force brute. La clé de session est utilisée pour une session de durée limitée, c'est-à-dire après une période de temps fixe, une nouvelle session doit être établie.

2. Il est informatiquement impossible de trouver l'inverse **(A^{-1})** d'un automate cellulaire réversible bidimensionnelle **(A)** [V.11], [V.12].

3. Il est informatiquement impossible de trouver X et Y à partir de **H (X, Y)** quel que soit **X** ou **Y** connu [caractéristique des fonctions de hachage].

4. Il est informatiquement impossible de trouver **X** et **Y** de telle sorte que **H (x) = H (Y)** [caractéristique des fonctions Hash].

Il n'est pas possible pour un attaquant de deviner la clé de session **K** en raison de la propriété 1. Le protocole est complet lorsque les deux parties vérifient avec succès, c'est-à-dire si **H (H (K_A))** = **H (H (K_B))** et **H (K_A)** = **H (K_B)**. Il est impossible pour l'attaquant de trouver **K**, en raison de la propriété 4.

Si l'attaquant est un attaquant passif ; Il capture l'information entière en entendant **R_A, A, Q**. La seule façon de trouver **K_A** ou **K_B** est de déterminer **A^{-1}** (l'inverse de l'RCA **A**) et mot de passe de **H (R_A, P)**. Il est impossible de calculer à cause des propriétés citées 2, 3 et 4.

Tous les mots de passe sont vulnérables aux attaques de dictionnaire, si la moindre opportunité est donnée. L'attaquant peut faire cette attaque en se faisant passer pour Alice ou Bob pour recueillir des informations lors de l'échange de clés. Cependant, il n'est pas possible pour lui, de se faire passer pour Alice puisqu'il ne peut pas générer **OID_i**. À cause de deux inconnues (**p** et **R_B**).

Une clé de session volée (**K_i**) est utilisée pour monter une attaque par dictionnaire sur le mot de passe **p**. Pour être protégé contre une telle attaque, le protocole doit être conçu de telle sorte que même si une clé de session est compromise, l'adversaire ne peut ni calculer le mot de passe ni vérifier la validité

du mot de passe deviné. Pour analyser ceci, supposons que l'adversaire connaisse la clé de session **Ki = (H (P, R_A), H (R_B, P))**, la clé partielle **Q = A (R_B) ⊕ H (P, R_A)** et **R_A**. Mais il est impossible de déterminer **p** à partir de ces valeurs connues en raison des propriétés 2 et 3. De plus, si l'attaquant suppose un mot de passe, il ne peut pas vérifier sa validité.

Un adversaire qui usurpe l'identité d'Alice, exploite **K_i** pour établir la session subséquente avec Bob. Il génère un RCA **A'** envoie **<ID_A, R_A, A', OID_i>** à Bob et reçoit **Q** de Bob. Mais, l'attaquant ne peut pas déterminer en raison du manque de la connaissance de **p**.

Un adversaire peut se faire passer pour Alice pour établir une connexion en répondant aux précédentes valeurs **<ID_A, R_A, A, OID_i>**. Cependant, Bob n'acceptera pas ceci comme **OID_i** est utilisé pour une seule fois. Même s'il imite Bob, en envoyant un **<Q>** précédent à Alice. La tentative échoue puisque **Q** est calculé à partir d'une valeur **R_A** différente.

Un attaquant ne peut pas dériver les clés de session précédentes même si un mot de passe **p** est compromis. En trouvant le mot de passe **p**. S'il essaie de trouver les clés de session précédentes à partir de l'information {**R_A** et **Q = A (R_B) ⊕ H (P, R_A)**} recueillies lors de sessions de communication passées, il peut déterminer **A (R_B)** de **Q**. Mais Il ne peut pas déterminer **R_B** de **A (R_B)** en raison de la propriété 2. Par conséquent RPAKE fournit la confidentialité persistante.

2.2.3. Les limites du RPAKE

Le schéma est robuste contre toutes les attaques actives et passives connues, de plus même les objectifs souhaités par ce protocole atteint les objectifs renforcés, du cout le schéma est robuste en terme d'analyse de sécurité.

Mais nous pouvons détecter des lacunes en termes d'analyse d'implémentation, ce schéma utilise 12 instances de fonction de hachage, 6 instances pour la phase d'authentification et d'établissement de clé et 6 autres pour la phase de confirmation de de la clé générée. Cela augmente le temps d'exécution du protocole au niveau du client lorsqu'il s'agit d'un dispositif à faible capacité de calcul et au niveau de serveur lorsqu'il s'agit d'un nombre d'utilisateur important. A ce point ça devient nécessaire d'optimiser le cout d'implémentation en diminuant d'un côté le nombre de message échangés durant l'exécution du protocole et les instances de fonction de hachage d'un autre côté.

2.3. L'approche CA-KEP : MH Bhuyan, DK Bhattacharyya & JK Kalita [V.3]

Dans cette section, nous présentons un protocole d'échange rapide de clés à deux parties, facilement implémentable et sécurisé, appelé CA-KEP. Ce protocole est conçu en fonction des caractéristiques des CA programmables groupées et non groupées. CA-KEP est supérieur aux approches [V.8, V.13-V.16] en considérant la rapidité d'exécution et les invulnérabilités à la plupart des attaques connues.

2.3.1. Les détails du fonctionnement du CA-KEP

Cette section décrit le protocole CA-KEP avec les illustrations nécessaires. Comme d'autres protocoles, CA-KEP comprend également deux phases : la phase de génération des clés et la phase de confirmation des clés.

Alice et Bob sont des parties légitimes bien comportées pour la communication. Dans les scénarios user-to-host, Alice est l'utilisateur. Les symboles utilisés pour définir le protocole sont présentés dans le Tableau V.3.

Tableau V. 3 - Symboles utilisées (CA-KEP)

M	Module pour prétraiter et extraire des fonctionnalités d'empreintes digitales pour générer un secret S partagé (r-bit).
f_u	Empreinte digitale de l'utilisateur mutuellement partagée entre les deux parties.
S	Un petit secret partagé de taille (r-bits)
P	Un grand nombre premier (2048 bits)
q	Un grand facteur premier de p-1.
F	Un module cryptographique à base de CA à deux phases pour générer un secret partagé de grande taille (2r bits) basé sur S.
R_A, R_B	Nombres aléatoires choisis par Alice et Bob.
Q_A, Q_B	Valeurs exponentielles envoyées par Alice et Bob
$E_A(m)$, $E_B(m)$	Le cryptage de m basé sur RSA en utilisant les clés publiques de A et B partagées entre eux.
h	Fonction de hachage unidirectionnelle à base de MACA.
$A \rightarrow B : m$	Alice envoie m à Bob.
k	Clé de session générée.

CA-KEP nécessite six étapes pour réaliser les phases de génération de clé et de confirmation de clé. Les principales étapes du protocole proposé sont présentées ci-dessous.

2.3.1.1. La phase de génération de clé

S1- Alice calcule : $Q_A = F(S)^{R_A} \bmod p$;

$A \rightarrow B : E_B (Q_A, h(Q_A), R_A)$, en utilisant la clé publique de Bob ;

S2- Bob calcule : $Q_B = F(S)^{R_B} \bmod p$;

$B \rightarrow A : E_A (Q_B, h(Q_B), R_B)$, en utilisant la clé publique d'Alice ;

S3- Alice calcule : $k = h (Q_B^{R_A} \bmod p)$;

S4- Bob calcule : $k = h (Q_A^{R_B} \bmod p)$;

2.3.1.2. La phase de confirmation de clé

S5- Alice vérifie : $A \rightarrow B : h (R_B - 1)$;

S6- Bob vérifie : $B \rightarrow A : h (R_A - 1)$;

Les trois modules de base utilisés par CA-KEP :

(i) pour générer **S** sur la base des caractéristiques d'empreintes digitales ;

(ii) pour générer des **S** expansés, aléatoires et permutés basés sur **F** ;

(iii) un hachage basé sur MACA ;

Sont discutés ci-après.

Le module : M

Ce module est utilisé pour générer S. Une illustration de ce module est représentée sur la Figure V.2. Elle fournit une option pour le changement périodique de la sélection du point d'initiation pour l'extraction des informations de minuties vraies en position de l'image d'empreinte digitale de l'utilisateur. La structure programmable de MACA facilite encore le remplacement périodique de S par les deux parties.

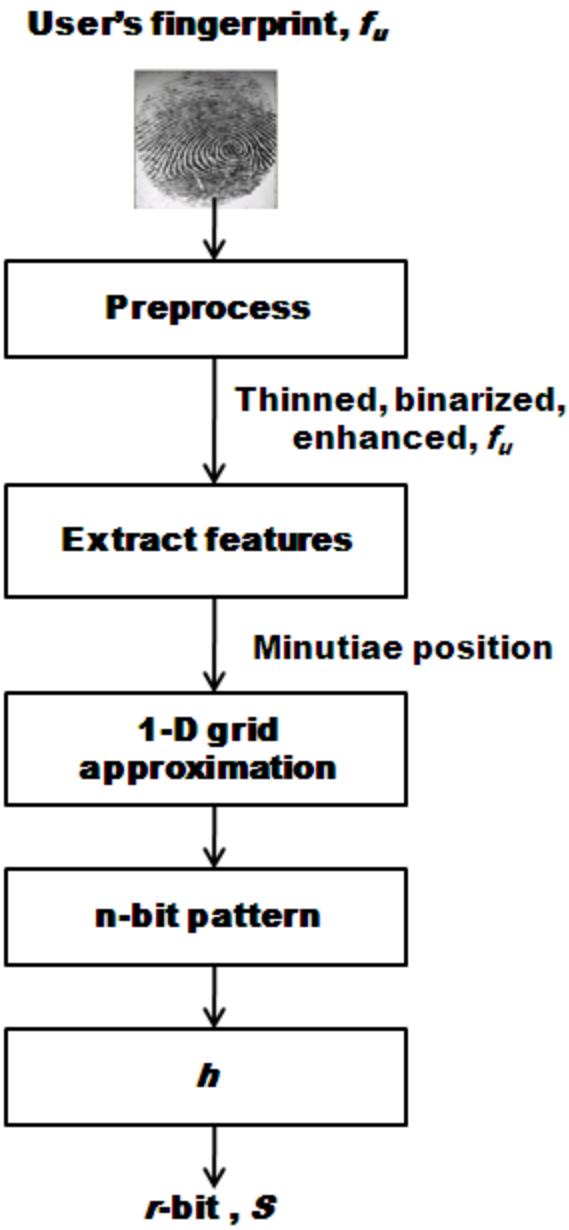


Figure V. 2 - Module M pour générer S (CA-KEP)

La fonction cryptographique d'expansion : F

Il comprend deux sous-modules : f1 et f2. Une illustration de F est montrée à la figure V.3.

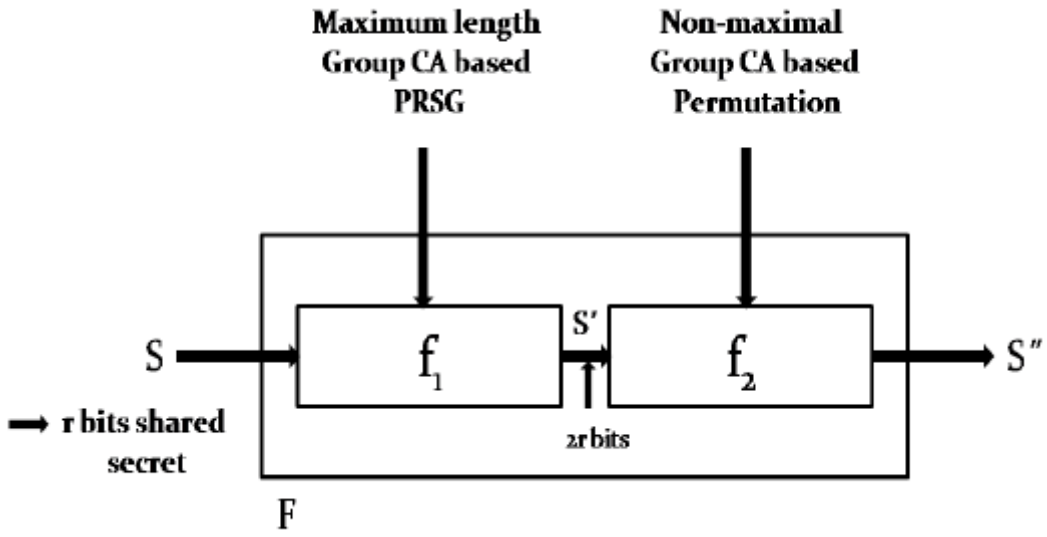


Figure V. 3 - La fonction d'expansion F (CA-KEP)

2.3.2. Analyse et discutions

Le CA-KEP est développé en fonction de la DH-base. Le protocole CA-KEP est montré invulnérable à toutes les attaques connues telles que (i) attaque de dictionnaire hors ligne, (ii) attaque de dictionnaire en ligne, (iii) attaque man-in-the-middle, (iv) attaque par rejeu, (vi) confidentialité persistante (vii), et attaque logarithmique discrète selon l'analyse de sécurité dans [V.17]. Ci-dessous, nous donnons une brève discussion sur chacune de ces attaques.

2.3.2.1. Attaque par dictionnaire hors ligne

Le CA-KEP proposé est à l'abri de cette attaque, car même si un adversaire extrait les grandes valeurs intermédiaires, c'est-à-dire $\mathbf{Q_A}$ et $\mathbf{Q_B}$ en gérant les clés privées des destinataires, il est extrêmement difficile pour lui de dériver **S**, $\mathbf{R_A}$ et $\mathbf{R_B}$.

Puisque $\mathbf{R_A}$ et $\mathbf{R_B}$ sont de grands nombres générés au hasard, les deviner correctement pour l'estimation exacte de S est extrêmement difficile. De plus, **S** n'est pas directement utilisé comme base pour l'opération exponentielle, mais il dépend d'une fonction cryptographique **F**, qui est à nouveau une combinaison de deux fonctions cryptographiques sécurisées, c'est-à-dire **f1** et **f2** (établies pour être sécurisées dans la sous-section). Par conséquent, ce protocole proposé peut contrecarrer l'attaque par dictionnaire hors ligne.

2.3.2.2. Attaque par dictionnaire en ligne

Pour monter une attaque de devinette en ligne réussie, initialement, l'adversaire doit changer la clé privée du destinataire pour extraire les valeurs de $\mathbf{Q_A}$ et $\mathbf{Q_B}$. Étant donné que le destinataire en est également fourni une disposition

supplémentaire pour vérifier l'intégrité de $\mathbf{Q_A}$ basée sur $\mathbf{h(Q_A)}$ où **h** est une fonction de hachage sécurisée, une approximation mauvaise ou incorrecte de $\mathbf{Q_A}$ ou $\mathbf{Q_B}$ est facilement détectée, même si un adversaire réussit à deviner $\mathbf{Q_A}$ et $\mathbf{Q_B}$ correctement. L'attaquant doit deviner $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** correctement. Mais, la taille de $\mathbf{R_A}$ ou $\mathbf{R_B}$ est 2r-bits et **F (S)** est estimée en appliquant la fonction cryptographique sécurisée **F** sur **S**. Même si $\mathbf{Q_A}$ et $\mathbf{Q_B}$ sont extraits, deviner $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** correctement est extrêmement difficile pour une telle attaque.

2.3.2.3. Attaque de l'homme du milieu

Dans cette attaque, un adversaire dit que **C** essaie d'imiter l'identité de **A** ou **B** pour communiquer avec **A** ou **B** en capturant soit la communication $\mathbf{E_B\ (Q_A,\ h\ (Q_A),\ R_B)}$ soit $\mathbf{E_A\ (Q_B,\ h\ (Q_B),\ R_A)}$. Même si **C** tente d'envoyer un message substitué à **A** ou **B** en cryptant avec sa clé publique, la partie légitime sera détectée au cours de l'étape 5 et de l'étape 6. De plus, pendant l'étape 3 et l'étape 4, elle conduit à deux valeurs **k** différentes. En outre, le fait de deviner le **F (S)** correct est extrêmement difficile en temps réel en raison de son espace clé extrêmement important et du changement de $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** pour chaque session. Par conséquent, le CA-KEP proposé peut facilement résister à l'attaque man-in-the-middle.

2.3.2.4. Attaque par rejeu

La CA-KEP proposée génère **k** sur la base d'un nouvel ensemble de $\mathbf{R_A}$, $\mathbf{R_B}$ (un grand nombre de 2r bits choisi au hasard) et **F (S)** pour chaque session. Ainsi, même si un adversaire capte les communications intermédiaires, c'est-à-dire $\mathbf{E_A\ (Q_B,\ h\ (Q_B),\ R_B)}$ ou $\mathbf{E_B\ (Q_A,\ h\ (Q_A),\ R_A)}$ et réussie à extraire $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** Il ne sert à rien pour la prochaine session. Il est extrêmement difficile pour l'adversaire de générer / produire le **k** correct pendant la session en cours. Là encore, à partir des informations capturées précédemment, il n'est pas non plus facile pour l'adversaire de prétendre être **A** ou **B** dans les communications successives, car à chaque session les parties légitimes utilisent un ensemble différent de $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** pour la génération de **k**.

2.3.2.5. La confidentialité persistante

Dans cette attaque, un adversaire tente d'extraire les $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)** actuels et finalement les **S** basés sur les précédemment utilisés ou passé **k**. Cependant, l'extraction de $\mathbf{R_A}$, $\mathbf{R_B}$ et finalement **S** basée sur **k** est une tâche extrêmement difficile en raison de l'utilisation d'un grand exposant et d'une fonction de hachage unidirectionnelle sécurisée. De plus, **k** est généré pour chaque session sur la base d'un nouvel ensemble de $\mathbf{R_A}$, $\mathbf{R_B}$ et **F (S)**. Par conséquent, le schéma

CA-KEP proposée fournit la confidentialité persistante même si la clé de session actuelle **k** est compromise.

2.3.3. Les limites de l'approche

En termes de sécurité le protocole semble sécurisé de toute attaque connue. Les limites de cette approche résident dans le fait que l'exécution du protocole dépend de plusieurs modules intermédiaires indépendants du protocole.

Le premier module est le générateur du secret à partir de l'empreinte digital, cela demande des processus compliqués et des dispositifs couteux pour les entreprises.

Un autre module exigé par ce protocole, est le module de gestion des clés publiques. Chaque participant doit passer par une autorité de certification hors ligne dont tous les autres participants font confiance afin de certifier sa paire de clé publique.

En plus de ces limites, nous constatons une limite applicative dans le cas d'une application (user-to-user authentication), le partage préalable de l'empreinte digital reste une problématique.

Chapitre VI :

Approche 2 :
Construction de nouveau schéma d'échange de clés authentifié à base de mot de passe (PAKE) en utilisant les automates cellulaires bidimensionnels

Chapitre VI : Construction de nouveau schéma d'échange de clés authentifié à base de mot de passe (PAKE) en utilisant les automates cellulaires bidimensionnels

Sommaire

1. Introduction

Les protocoles d'authentification à base de mot de passe sont les plus répandus dans nos jours ; il est plus facile et moins coûteux de se rappeler d'un court secret (généralement un mot de passe) que de maintenir une clé dans un support de stockage sécurisé (généralement une carte à puce).

1.1. La problématique

Le problème se résume dans la capacité de surmonter une attaque par force brute pour trouver le secret (le mot de passe à faible entropie).

Donc pour concevoir un système d'authentification solide à base de mot de passe, nous devons prendre en considération les propriétés particulières et les exigences suivantes :

• Les utilisateurs ne possèdent qu'un secret de faible entropie. Plus précisément, il est possible pour l'adversaire de rechercher dans tous les secrets possibles dans un délai raisonnable.

• Les attaques par dictionnaire hors ligne ne devraient pas être possibles. Cela signifie qu'un indiscret passif qui enregistre la transcription d'une ou plusieurs sessions ne peut pas éliminer un nombre important de mots de passe possibles.

• Les attaques par dictionnaire en ligne ne devraient pas être possibles. Cela signifie que l'adversaire actif ne peut pas abuser le protocole de façon à éliminer un nombre important de mots de passe possibles. Un adversaire actif peut toujours éliminer au moins un mot de passe par tentative de se faire passer pour l'utilisateur légitime en utilisant ce mot de passe.

1.2. La contribution

Notre principale contribution est la minimisation du nombre de flux de messages échangés durant la phase d'authentification et de confirmation de clé, ainsi que la minimisation des opérations coûteuses de calcul tel que (les fonctions de hachage…) en assurant un niveau de sécurité élevé.

2. Notre nouvelle conception

2.1. Les hypothèses de conception du protocole

Hypothèse IV.1 : C'est une hypothèse où l'adversaire peut écouter tous les messages envoyés ou reçus. Ceci est une hypothèse réaliste dans les systèmes de communication typiques.

Hypothèse IV.2 : Une autre hypothèse réaliste où l'adversaire peut intercepter, modifier et rejouer des messages.

Hypothèse IV.3 : Encore une fois, une hypothèse tout à fait réaliste - il est largement admis que les participants internes sont souvent plus une menace que les étrangers. Cela nous amène à une autre hypothèse de sécurité.

Un scénario typique est celui où deux participants s'engagent à un protocole de clé partagée et l'un retourne un défi qui été destiné à lui-même.

Hypothèse IV.4 : Cette attaque ne peut être possible que si l'exécution parallèle du même protocole sont autorisés, mais ce qui est souvent une hypothèse réaliste.

2.2. L'architecture du protocole

Comme nous avons décrit dans le premier chapitre ; Il y a trois caractéristiques que nous considérons comme des critères architecturaux pour classer les différents protocoles d'authentification et d'établissement de clés cryptographiques : les clés cryptographiques existantes, comment la clé de session est établie et le nombre d'utilisateurs d'un protocole est conçu pour servir.

2.2.1. Les clés cryptographiques existantes

Notez qu'il est impossible d'établir une clé de session sans canaux sécurisés existants déjà disponibles. Par conséquent, si deux participants souhaitent établir une nouvelle clé de session, il y a essentiellement trois possibilités.

1. Les participants partagent au préalable un secret.

2. Un serveur hors-ligne est utilisé. Cela signifie que les participants possèdent la certification des clés publiques.

3. Un serveur en ligne est utilisé. Cela signifie que chaque participant partage une clé avec un serveur de confiance.

Nous avons opté pour le premier choix où les deux participants partagent au préalable un petit secret (mot de passe à faible entropie) facilement mémorisable, puis une clé secrète est dérivée à partir du mot de passe.

Critère architectural IV.1 : Alice et Bob partagent au préalable un secret partagé afin de garantir l'authentification de l'entité et l'établissement de clé sécurisés.

2.2.2. Les méthodes de génération de clés de session

Il y a trois façons pour générer des clés de session dans un protocole d'établissement de clé : la première : il s'agit d'un mécanisme de transport de clé dans lequel l'un des participants génère une clé de session valable et celle-ci sera envoyée aux autres participants. La deuxième : il s'agit d'un mécanisme d'accord de clé dans lequel la clé de session est fonction de contribution par tous les participants du protocole. Tandis que la troisième est une méthode hybride des deux premières façons.

Dans notre conception nous avons utilisé la méthode de transport de clés ; là où l'un des participant génère une clé de session valable et la transporter en toute sécurité à l'autre partie.

Critère architectural IV.2 : La clé de session est générée par Bob et transporté à Alice en toute sécurité.

2.2.3. Le nombre d'utilisateurs

A ne pas confondre avec le nombre maximal d'utilisateurs supporté par le protocole. Ici on parle de combien d'utilisateurs contribuent à la fois à la phase d'établissement de clé.

Dans notre protocole seul le participant concerné contribue à la phase de génération de la clé de session ; et puis la transporter à l'entité homologue. Par

conséquent, seules les parties légitimes auront connaissance de la clé de session après l'exécution du protocole.

Critère architectural IV.3 : Seul l'une des parties contribue à la fois pour la génération de la clé de session et la transporter à l'entité homologue.

2.3. La fraicheur de la clé de session

Il existe deux façons pour que la fraicheur d'une valeur puisse être garantie à un utilisateur particulier. La première est la contribution de ce dernier au choix de la valeur (la clé de session dans notre cas), tandis que la seconde repose sur la réception d'une valeur connue comme fraîche.

Nous avons conçu notre protocole de telle sorte que l'un des participants génère une clé de session fraiche pour chaque exécution du protocole et que l'autre participant doit s'assurer de sa fraicheur. Nous expliquons en détails la phase de transport de la clé de session (2.7.1).

Dans notre schéma la clé de session n'est pas dérivée du secret partagé puisque la confidentialité persistante est un objectif souhaitable dans notre protocole. Nous utilisons les clés éphémères de la cryptographie asymétrique pour assurer la fraicheur de la clé et par la même occasion fournir la confidentialité persistante.

2.4. Les objectifs souhaités

Nous allons citer une liste détaillée d'objectifs souhaités par notre protocole, la liste est illustrée dans le tableau suivant :

Tableau VI. 1 - Les objectifs souhaités

- Le protocole doit assurer l'authentification de l'entité ; c.-à-d. l'une des parties doit être assurée de l'identité de la seconde impliquée dans le protocole, et que la seconde doit effectivement avoir participé.
- Le protocole doit assurer l'authentification mutuelle entre Bob et Alice ; c'est-à-dire l'objectif numéro un doit être assuré dans les deux sens.

- Le protocole doit établir une clé de session fraîche, connu seulement aux participants à la session.
- Le protocole doit fournir l'intégrité des clés ;
- Le protocole doit fournir la confirmation de clé du Alice au Bob c.-à-d. que Alice doit avoir l'assurance que la clé de session $\mathbf{K_i}$ est une bonne clé pour communiquer avec Bob, et que Bob est en possession de cette clé.
- Le protocole a pour objectif de fournir la confiance mutuelle en clé $\mathbf{K_i}$ c.-à-d. Prévision du Bob que K_s est une bonne clé pour une utilisation avec Alice, et souhaite du Alice souhaite communiquer avec Bob en utilisant la clé $\mathbf{K_i}$ dont lequel Alice la croit bonne pour cet usage.
- Finalement, le protocole a pour objectif de fournir la confidentialité persistante c.-à-d. Si le secret à long terme est compromis, les clés de session qui ont été précédemment établies à l'aide de ce secret ne doivent pas être compromises aussi.

2.5. Motivation et choix des outils

2.5.1. Pourquoi choisir les automates cellulaires comme outil mathématique ?

La raison de la popularité des automates cellulaires peut être attribuée à leur simplicité, et de l'énorme potentiel qu'ils détiennent dans la modélisation de systèmes complexes, en dépit de leur simplicité. En outre, étant donné que les opérations CA utilisent les opérateurs binaires simples qui sont exécutées indépendamment pour chaque bit, le mécanisme d'évolution basé sur les CA a un parallélisme inhérent, et fournit donc une implémentation extrêmement rapide et efficace, en particulier sur la conception du matériel. Le système proposé ne nécessite pas de cryptage des données explicites, ni du côté du client, ni sur le serveur. Ces faits conduisent à une grande vitesse d'authentification des performances de fonctionnement en temps réel.

Les automates cellulaires peuvent être parfaitement adaptés à une étude et une analyse détaillée, parce qu'ils peuvent être simulés facilement dans un ordinateur ou sur un dispositif de faible capacité de calcul.

2.6. Que veut-on sécuriser ? Contre quels dangers ?

La confidentialité de la clé de session et du secret partagé est obligatoire afin de contrecarrer contre les attaques passives. En plus des attaques passives : il est nécessaire pour les concepteurs des protocoles d'authentification de proposer des techniques afin de contrecarrer contre les attaques actives.

Le tableau suivant présente l'ensemble des attaques qu'on souhaite contrecarrer durant la conception et la construction de notre protocole ; afin de mieux éclaircir les objectifs requis de notre protocole. Puisque pour qu'un protocole puisse atteindre des objectifs particuliers, cela dépend de sa force à contrecarrer toutes les attaques considérées comme possibles.

Tableau VI. 2 - Liste des attaques possibles

(FR)	(EN)
Cassage de mot de passe	Password guessing attack
Attaque par rejeu et la non-synchronisation de l'horloge	Replay attack and clock un-synchronization
Authentification mutuelle et l'attaque de l'homme du milieu	Mutual authentication and man in the middle attack
Accord de clé de session et attaque par clé connue	Session key agreement, forgery attack and known key security
Pas une table de stockage et vol de table de vérification	No verifier-table stored and stolen verifier attack
Attaque par réflexion	Reflection attack
Attaque par déni de service	Denial Of Service (DOS)

Ces attaques vont être détaillées dans une section ultérieure (L'expérimentation des résultats).

2.7. La construction du protocole

Contrairement à la première approche qui est totalement indépendante de tout mécanisme cryptographique ; la deuxième approche est une conception théorique qui dépend de n'importe quel mécanisme cryptographique (symétrique, asymétrique et fonction à sens unique). Ce qu'on appelle modularité du crypto-système.

Nous laissons une liberté totale pour l'intégrateur de notre solution de choisir n'importe quel mécanisme cryptographique à condition qu'il soit sûr et prouvé, ce choix est effectué selon les besoins.

Nous recommandons des mécanismes à base d'automates cellulaires prouvés sûrs afin d'obtenir une implémentation rapide, légère et efficace. Pour la cryptographie symétrique nous optons pour le crypto-système de notre approche prouvée [IV.1] ; pour la cryptographie asymétrique nous recommandons le crypto-système de Kari [VI.1] et pour les fonctions à sens unique nous choisissons [VI.4] parmi [VI.2-VI.4].

En utilisant le mécanisme des automates cellulaires réversibles de second ordre, nous proposons un nouveau système d'échange de clés authentifiés. La sécurité de notre approche repose seulement sur un petit mot de passe à faible entropie ($\mathbf{PW_i}$) partagée entre les deux communicants (Alice et Bob), et sur la difficulté d'inverser un RCA à deux dimensions.

En s'appuyant sur cette théorie, l'RCA peut être utilisé comme un crypto-système. Soit T = (d, s, N, f) un RCA et T^{-1}= (d, s, N', f') soit son inverse. Le message (**M**) est écrit en matrice d-dimensionnelle t.q : d (≥2). Maintenant, le cryptogramme peut être obtenu en appliquant **T** sur **M**. En sens inverse, en utilisant $\mathbf{T^{-1}}$ sur le cryptogramme, on peut déduire le message d'origine. Une dimension supérieure d (≥2) RCA doit être utilisée à cette fin comme tous les RCA unidimensionnels peuvent être inversés efficacement [VI.5]. En dehors de cela, le RCA utilisé dans le système proposé est nécessaire pour avoir les propriétés suivantes :

1. connaissant un RCA **T**, il est infaisable de trouver $\mathbf{T^{-1}}$;
2. Cependant, il est plus facile de trouver **T** en sachant $\mathbf{T^{-1}}$.

Un ensemble de RCA dont les éléments sont facilement inversibles peut être utilisée à cette fin. La sélection d'un ensemble de RCA facilement inversible (T_1, T_2, ... T_n), et leur composition T = (T_1 o T_2 ... o T_n) peut être publiée pour l'utilisation de cryptage, T_1, T_2, ... En secret. T^{-1} = (T_1^{-1}, T_2^{-1}, ... T_n^{-1}) peut être utilisé pour le décryptage sans sacrifier le secret.

Le concept de défi d'une façon similaire du principe de nonce. Nous utilisons la clé de session précédente $\mathbf{K_{i-1}}$ comme un défi au lieu de générer et d'envoyer dans chaque exécution de nouveaux nonces, cela permet de réduire le nombre de messages échangés durant l'exécution du protocole. L'utilisation de ce défi permet par la même occasion d'éliminer le problème de synchronisation de temps entre les participants.

La clé $\mathbf{K_{i-1}}$ est utilisée dans un scénario de défi-réponse dans lequel les deux participants conservent la valeur de la clé de session précédente à usage unique

propre à chaque session bien sûr. Alice envoie alors $\mathbf{K_{i-1}}$ codé par la clé **S** dérivée depuis le mot de passe partagé ($\mathbf{PW_i}$) afin de répondre au défi que Bob attend. Alice vérifie la validité du défi, sinon, Bob refuse la demande d'Alice.

Le système d'authentification proposé comprend deux phases : une phase d'authentification et échange de clé, et une phase de confirmation de connaissance de clé. Les notations suivantes sont adoptées dans le reste de ce chapitre :

Tableau VI. 3 - La table des notations

H() :	Une fonction cryptographique à sens unique
Ψ () :	La fonction d'expansion
K_{i-1} :	La clé de session précédente utilisée comme défi
ID_i :	L'identificateur de l'utilisateur U_i (Alice)
PW_i :	Le mot de passe de l'utilisateur U_i (Alice)
S :	La clé secrète dérivée du PW_i
K_i :	La clé de session courante
T :	La clé publique générée par Alice
T^{-1}	La clé privée générée correspondante à T, générée par Alice

2.7.1. La phase d'authentification et d'échange de clé

Notre protocole dispense de la phase d'enregistrement puisque le protocole n'a besoin que d'un petit mot de passe pour être accompli avec succès. Cette phase permet une authentification mutuelle entre les deux parties (Alice et Bob), de sorte que chacun vérifie l'identité de l'autre. Après la phase d'authentification, une clé de session partagée est mise en place afin de permettre une communication sécurisée entre les deux parties au cours d'une session de communication prédéterminée. Un indiscret qui a accès aux messages échangés ne devrait pas être en mesure de tirer des informations sur le secret **S**, ou sur la clé de session établie $\mathbf{K_i}$.

2.7.1.1. La phase d'échange de clé authentifié

La phase d'authentification mutuelle et d'échange de clé se déroule selon 3 étapes :

• **MAKE-1 :** Pour Alice qui souhaite s'authentifier auprès de Bob, il doit d'abord saisir son identifiant $\mathbf{ID_i}$ et son mot de passe $\mathbf{PW_i}$ pour commencer une session d'accès :

- Une clé secrète **S** est dérivée à partir du mot de passe $\mathbf{PW_i}$;

- Alice génère, un automate cellulaire réversible **T** et son inverse $\mathbf{T^{-1}}$;
- Ensuite, Alice calcule le cryptogramme $\mathbf{X = E_s (K_{i-1}, T)}$.
- Finalement, le couple <**ID**, **X>** est envoyé à Bob.

Notez qu'Alice n'a pas besoin d'attendre un défi de la part de Bob parce qu'il est au courant du le défi ($\mathbf{K_{i-1}}$) qu'il doit coder et envoyer à Bob. Bob de son côté attend qu'il reçoive le défi sous forme de cryptogramme afin qu'il puisse authentifier Alice.

• **MAKE-2 :** Lors de la réception du couple ci-dessus, tout d'abord Bob doit récupérer la clé $\mathbf{K_{i-1}}$ à partir du cryptogramme $\mathbf{X = E_s (K_{i-1}, T)}$ en utilisant la clé secrète **S** afin qu'il puisse authentifier Alice. Après avoir comparé la clé récupérée à la clé de la session précédente stockée, Bob authentifie et accepte la demande d'Alice en cas d'égalité, sinon, la requête de l'utilisateur est refusée.

Afin d'effectuer une authentification mutuelle, le serveur doit également prouver son identité à l'utilisateur. Pour ce faire, la clé publique éphémère reçue est utilisée. Bob génère une clé de communication pour la session actuelle puis encode cette clé avec la clé de session précédente sous le cryptogramme $\mathbf{Y = T (K_{i-1}, K_i)}$ en utilisant la clé publique éphémère **T**. Alice essaie de récupérer le défi codé sous une nouvelle forme fraiche et unique. Si l'égalité est vérifiée, alors Bob est authentifié auprès d'Alice. Dans le cas contraire, Bob est considéré comme intrus, et la connexion est interrompue. La figure VI.1 illustre le schéma entièrement détaillé.

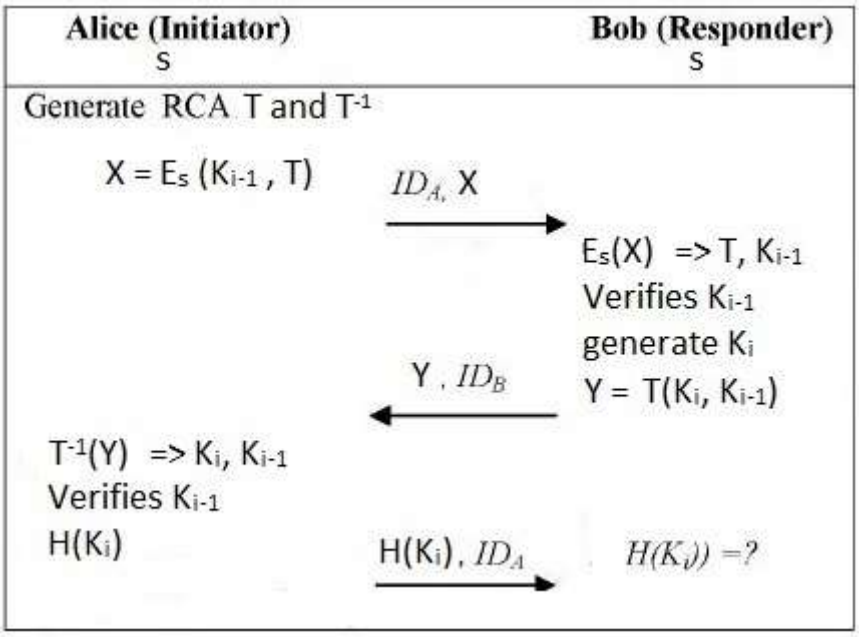


Figure VI. 1 - Diagramme d'échange de clés authentifié proposé

2.7.1.2. Confirmation de connaissance de clé

Après avoir effectué l'authentification mutuelle avec succès, les deux parties doivent se faire confirmer la connaissance de la clé qui est utilisée pour la communication sécurisée lors de la session en cours. La clé de session $\mathbf{K_i}$ est générée par Bob et envoyée à Alice durant les messages échangés lors de la phase d'authentification (MAKE-2). Un espion qui recueille tous les messages échangés est incapable d'en déduire toutes les informations relatives à la clé de session puisqu'Alice est la seule qui possède la clé privée du déchiffrement $\mathbf{T^{-1}}$.

• **KC-1** : La confirmation de la clé de Bob auprès d'Alice est une partie intégrante du message (MAKE-2) puisque on utilise le mécanisme de transport de clé.

• **KC-2** : La confirmation de la clé d'Alice auprès de Bob est effectuée par l'envoi de la clé de session courante haché $\mathbf{H(K_i)}$; il est clair que ce message est frais et unique et ne peut pas être forgé par un utilisateur malveillant. A la réception de ce hach, Bob effectue la même opération et compare les deux haches, s'ils sont égaux Bob sera sûr que Alice possède la bonne clé de communication. Enfin une communication sécurisée peut être effectuée.

2.8. L'expérimentation des résultats

Dans ce qui suit, nous discutons des propriétés de sécurité du schéma proposé par rapport aux attaques potentielles communes. Il est supposé que l'attaquant a le potentiel requis pour écouter le canal de communication entre Alice et Bob et peut modifier tout message échangé. Un tableau comparatif est également présenté dans la section qui suit entre le schéma proposé et certains schémas d'authentification existants à base des CA.

2.8.1. Cassage de mot de passe

Dans le schéma proposé, le mot de passe de l'utilisateur $\mathbf{PW_i}$ est transformé en une clé secrète **S** par le biais d'une fonction d'expansion **Ψ** où $\mathbf{S = \Psi(PW_i)}$. En outre, la clé secrète dérivée du mot de passe est utilisée pour chiffrer le RCA **T** qui est lui-même généré aléatoirement et la clé de session précédente $\mathbf{K_{i-1}}$. Sachant que **T** et $\mathbf{K_{i-1}}$ n'ont été transmis nulle part en clair sur le réseau. Le seul paquet concernant le mot de passe transmis de Alice à Bob contient <$\mathbf{ID, X = E_S(K_{i-1}, T)}$>.

Par conséquent, l'attaquant est incapable d'extraire des informations à propos de mot de passe à partir des identifiants et des messages écoutés transmis entre Alice et Bob, donc il n'y a aucun moyen de monter une attaque hors ligne permettant de deviner le mot de passe (off-line password guessing attack). Enfin,

pour résister contre une attaque en ligne (on-line password guessing attack), le système limite la quantité des tentatives jusqu'à un nombre limité de fois.

2.8.2. Attaque par rejeu et la non-synchronisation de l'horloge

Le système d'authentification à base de timestamp peut souffrir de l'attaque de rejeu puisque le délai de transmission est imprévisible dans un environnement de réseau. Pour cette raison, nous utilisons un modèle à base des nonces au lieu d'une conception basée sur l'horodatage (timestamp) pour le schéma proposé, de sorte que le système peut prévenir un tel problème grave de la non-synchronisation de l'horloge. Afin de mieux optimiser notre système les nonces ne sont pas générés à chaque fois pour lancer un défi, mais l'idée est d'utiliser la clé de session précédente $\mathbf{K_{i-1}}$ comme un nonce qui est à la fois connu par Alice et Bob.

Au cours de la phase d'authentification, un adversaire peut prétendre être un utilisateur valide et s'identifier auprès du serveur (Bob) en envoyant des messages qui ont été précédemment transmis par un utilisateur légitime (Alice). Au cours de la phase d'authentification, un adversaire peut prétendre être un utilisateur valide et tente de connecter au serveur en lui envoyant des messages qui ont été précédemment transmis par un utilisateur légitime. Pour éviter une telle attaque, l'utilisateur génère un RCA **T** imprévisible et différent pour chaque session, a qui rend les messages échangés d'une session donnée valable uniquement pour cette session, et ne pourraient pas être réutilisés dans d'autres sessions. Donc un adversaire qui rejoue un message **X** précédent, n'est pas en mesure d'effectué la dernière phase du protocole.

Par conséquent, un adversaire ne peut pas surmonter le processus de vérification auprès du serveur avec des valeurs légitimes précédemment écoutées.

De toute évidence, l'attaque de rejeu échoue. Par conséquent, le schéma proposé peut résister à une attaque de rejeu.

2.8.3. Authentification mutuelle et l'attaque de l'homme du milieu

Il est une exigence de base d'un système d'authentification que l'utilisateur et le serveur s'authentifient les uns auprès des autres. Dans le schéma proposé, le serveur authentifie l'utilisateur en vérifiant le message d'authentification en utilisant n'importe quel mécanisme de chiffrement symétrique ; dans notre

protocole nous recommandons le mécanisme de réversibilité des automates cellulaires de second ordre [IV.1]. Si le serveur arrive d'extraire $\mathbf{K_{i-1}}$ à partir du message de demande de connexion, le serveur confirme la légitimité de l'utilisateur. Le serveur répond alors avec un message spécifique ($\mathbf{Y = (T (K_i, K_{i-1})}$) à l'utilisateur, qui contient la nouvelle clé de session avec l'ancienne afin d'authentifier le serveur en vérifiant ce couple qui ne peut être construit que par le serveur légitime.

Par conséquent, l'authentification mutuelle est garantie par l'étape MAKE-1 et MAKE-2 du protocole. Nous savons que l'attaque d'usurpation d'identité de l'utilisateur et l'attaque de faux serveur ne peuvent être invoquées par un utilisateur malveillant $\mathbf{U_z}$. Par conséquent, il est impossible pour $\mathbf{U_z}$ de mener une attaque man-in-the-middle. Il est bien évident que le protocole est résistant à l'attaque de l'homme du milieu et l'authentification mutuelle est bien atteinte.

2.8.4. Accord sur la clé de session et l'attaque par clé connue

Après une authentification mutuelle réussie, une clé de session est transportée en toute sécurité et partagée enfin entre le client et le serveur pour être utilisée pendant le reste de la session de communication. Notre protocole garantit que la clé de session est connue seulement par les entités légitimes (le client et le serveur), Puisque RCA **T** est généré par Alice, donc seul Alice peut extraire la clé de session courante à partir du cryptogramme **Y** puisqu'elle est la seule qui possède son inverse $\mathbf{T^{-1}}$; par conséquent, aucun tiers ne peut envoyer un message forgé au client ou au serveur. Étant donné que le défi ($\mathbf{K_{i-1}}$) et le RCA généré aléatoirement sont différents pour chaque session, alors même si une clé de session donnée est compromise, ni la clé secrète ni les clés de sessions (passées et futures) ne seront compromises. Par conséquent, le schéma proposé assure (known key security).

2.8.5. Vol de table de vérification

Du moment qu'aucune table de vérification n'est stockée ni par Alice ni par Bob, l'authentification dépend seulement d'un petit mot de passe à faible entropie. Par conséquent, le système résiste à l'attaque par modification de table de stockage et assure la réduction du coût de stockage. Bob utilise la clé secrète **S** calculé depuis le mot de passe partagé $\mathbf{PW_i}$ pour authentifier Alice tout en garantissant la fraicheur du cryptogramme ; tandis que Alice utilise le RCA **T** et son inverse $\mathbf{T^{-1}}$ générés durant la phase d'authentification pour authentifier Bob.

Par conséquent, le système proposé est sécurisé contre les attaques (stolen verifier attack).

2.8.6. Attaque par réflexion

Une attaque par réflexion ne peut être effectuée que si et seulement si des exécutions parallèles du même protocole sont autorisées. Par exemple, si un participant est un hôte Internet, il peut accepter des sessions de plusieurs autres participants. Dans notre cas une attaque par réflexion ne peut être effectuée même si le protocole autorise plusieurs exécutions parallèles pour deux raisons - aucun défit ne circule en claire durant la phase d'authentication, même si un défi peut être dévoilé ou prédit, une telle attaque ne peut être accomplie vue la différence de la façon d'authentification d'Alice auprès de Bob (Par un mécanisme symétrique) et l'authentification de Bob auprès d'Alice (Par un mécanisme asymétrique).

2.8.7. La confidentialité persistante

L'idée de la confidentialité persistante est que quand une clé à long terme **(S)** est compromise, les clés de session qui ont été précédemment établies à l'aide de cette clé à long terme ne doivent pas être compromises aussi. Le concept crucial pour fournir la confidentialité persistante est le principe de la clé publique éphémère (dans notre cas nous avons utilisé le RCA **T** et son inverse $\mathbf{T^{-1}}$) ; celle-ci est une clé publique **T** qui est utilisée uniquement pendant la durée du protocole d'établissement de clé et est ensuite détruite avec sa clé privée $\mathbf{T^{-1}}$ correspondante. Si la clé publique est utilisée uniquement pour authentifier la clé de session, et que la clé de session ne peut pas être récupérée sans la clé privée éphémère, alors la clé $\mathbf{K_i}$ ne peut être récupérée depuis le cryptogramme $\mathbf{Y = T\ (Ki,\ K_{i-1})}$ sans la connaissance de la clé privée $\mathbf{T^{-1}}$. Par conséquent, notre protocole fournit la confidentialité persistante.

3. Analyse comparative

Afin d'illustrer les avantages du système proposé, une étude comparative a été réalisée par rapport à plusieurs schémas d'authentification existants basée sur les CA. Il est clair que l'approche proposée assure toutes les propriétés énumérées tout en nécessitant moins de coût de calcul.

3.1. Analyse de sécurité

La comparaison est effectuée par rapport aux propriétés de sécurité et la complexité d'implémentation. Le tableau VI.4 présente plusieurs aspects de

sécurité assurée par le schéma proposé par rapport à la plupart des schémas d'authentification à base des CA-base connus proposées dans [V.1], [V.2] et [V.3].

Tableau VI. 4 - Security aspects comparison

SECURITY CRITERION	CPAKE [V.1]	RPAKE [V.2]	CA-KEP [V.3]	PROPOSED
PERFECT FORWARD SECRECY	No	Yes	Yes	Yes
KNOWN KEY SECURITY	No	Yes	Yes	Yes
STOLEN SESSION KEY ATTACK	No	Yes	Yes	Yes
OFFLINE-PASSWORD GUESSING	No	Yes	Yes	Yes
RESISTANCE TO STOLEN VERIFIER ATTACK	No	Yes	No	Yes
RESISTANCE TO REPLAY ATTACK	Yes	Yes	Yes	Yes
MUTUAL AUTHENTICATION	Yes	Yes	Yes	Yes
RESISTANCE TO MAN IN THE MIDDLE ATTACK	Yes	Yes	Yes	Yes
USER IMPERSONATING ATTACK	Yes	Yes	Yes	Yes
SERVER COUNTERFEIT ATTACK	Yes	Yes	Yes	Yes
RESISTANCE TO FORGERY ATTACK	Yes	Yes	Yes	Yes

3.2. Analyse d'implémentation

Etant donné que les opérations CA utilisent les opérateurs binaires simples qui sont exécutées indépendamment pour chaque bit, le mécanisme d'évolution basé sur les CA a un parallélisme inhérent, et fournit donc une implémentation extrêmement rapide et efficace, en particulier sur la conception du matériel. Le système proposé ne nécessite pas de cryptage des données explicites, ni du côté du client, ni sur le serveur. Cela empêche également les attaques par déni de service basées sur la surcharge du serveur avec des demandes de service parasites.

Ces faits conduisent à une grande vitesse d'authentification des performances de fonctionnement en temps réel. En utilisant une analyse conceptuelle à la fois de l'approche proposée et celles qui existent déjà, le tableau comparatif IV.5 montre que l'approche proposée prend avantage en termes de coût de communication et de calcul.

Tableau VI. 5 - Comparaison des performances

MUTUAL AUTHENTICATION PHASE	CPAKE [V.1]	RPAKE [V.2]	CA-KEP [V.3]	PROPOSED
NOMBRE DE FLUX DE MESSAGES	4	2	4	2
INSTANCES DES FCT. DE HACHAGE	10	6	4	0
NOMBRE DE NOMBRE ALEATOIRE	6	2	2	0
INSTANCES D'EXP. MODULAIRE	0	0	4	0
INSTANCES DE CRYPT. SYMETRIQUES	2	0	0	2
INSTANCES DE CRYPT. ASYMETRIQUE	0	2	4	2
CLES PARTAGEES A LONG TERMES	S_K	-	P_K, S_K	-
KEY CONFIRMATION PHASE				
NOMBRE DE FLUX DE MESSAGES	4	2	2	1
INSTANCES DES FCT. DE HACHAGE	-	6	4	2
TOTAL				
NOMBRE DE FLUX DE MESSAGES	8	4	6	**3**
INSTANCES DES FCT. DE HACHAGE	-	12	8	**2**

4. Comparaison pratique

Une comparaison très pratique est présentée dans cette section entre les protocoles d'authentification à base de clé secrètes partagées à long terme et l'échange de clés authentifié sous forme de tableau VI.6.

Tableau VI. 6 - Comparaison pratique

	Authentification à base de clé secrète à long terme	**Echange de clés authentifié à base de mot de passe**
Objectifs	- On peut viser l'objectif d'authentification d'entité sans avoir besoin de celui d'établissement de clé.	- Les objectifs d'authentification d'entité et d'établissement de clé sont inséparables.
	- L'objectif de la confidentialité persistante n'est pas toujours souhaitable.	- L'objectif de la confidentialité persistante est toujours souhaitable.
Données intermédiaires	- On peut ajouter un mot de passe ou un code PIN afin de renforcer la sécurité.	- On peut ajouter une clé partagée à long terme (secrète S_k /publique P_k).
	- On peut transformer un mot de passe à faible entropie en une clé à long terme.	- On peut transformer un mot de passe à faible entropie en une clé à long terme.
Application	- Elle ne peut être utilisée que dans une architecture client/serveur. n (>2) tiers. Mais ne peut pas être appliquée dans une application décentralisée.	- Elle peut être utilisée dans un environnement décentralisé ; ainsi que dans une architecture client-serveur.
Phase d'enregistrement	- La phase d'enregistrement est obligatoire pour	- La phase d'enregistrement

	délivrer le secret à l'utilisateur légitime.	n'est pas obligatoire.

Conclusion et perspectives

Conclusion et perspective de recherche

Conclusion

Dans cette thèse nous sommes focalisés sur le problème d'authentification de l'entité et l'échange de clés cryptographiques entre deux utilisateurs qui souhaitent communiquer ou établir une session de communication sécurisée.

La première approche propose un nouveau système d'authentification autonome à distance basé sur des automates cellulaires qui garantit l'authentification et l'accord de clé de session de manière rapide et hautement sécurisée. En ce qui concerne les systèmes existants basés sur CA, le projet proposé utilise des opérations CA simples et parallèles sans outils cryptographiques supplémentaires tels que les fonctions de hachage ou de cryptage. Le système offre un haut niveau de sécurité et de performance avec un coût de calcul optimisé, alors que l'analyse effectuée et les expériences montrent qu'il résiste aux attaques connues. Le schéma proposé gère l'authentification des utilisateurs distants, l'accord de clé de session et l'authentification mutuelle entre l'utilisateur et le serveur en utilisant un modèle de calcul simple et élégant. Les expériences de calcul effectuées pour mesurer les performances de l'exécution montrent que le schéma offre les meilleures performances par rapport aux crypto-systèmes à base des CA existants.

Dans la deuxième approche, en tirant toujours avantage des CA nous avons conçu un schéma pour gérer les situations où aucune clé secrète n'est partagée entre les deux parties communicantes (protocole d'échange de clés similaire à Diffie-Hellman) en utilisant la réversibilité des automates cellulaires bidimensionnels de deuxième ordre. Cela nous a permis de construire un schéma d'authentification et d'échange de clé qui a pour objectif de fournir la confidentialité persistante, en se basant seulement sur un petit mot de passe partagé entre les deux utilisateurs sans falloir passer par la phase d'enregistrement auprès du fournisseur du service.

Perspectives de recherche

Annexes

Annexes

Sommaire

Annexe A
Acronymes

1. Annexe A : Les acronymes

2D: Two-dimensional.

2PKDP: Two-Party Key Distribution Protocol.

A:

AP: Access Point.

APDU: Application Protocol Data Unit.

API: Application Programming Interface.

AS: Authentication Server.

ATR: an Answer To Reset.

C:

CA: Certificate Authority / Cellular Automata.

CAD: Card Acceptance Device.

CA-KEP: a secure CA based 2-party Key Exchange Protocol.

CARA: Cellular Automata based Remote-user Authentication scheme.

CARMA: Cellular Automata based Remote Mutual Authentication scheme.

CPAKE: CA based Password-only Authenticated Key Exchange.

CSR: Certificate Signing Request.

D:

DES: Data Encryption Standard.

DH: Diffie-Hellman.

DoS: Denial of Service.

E:

ECA: Elementary Cellular Automata.

EKE: Encrypted Key Exchange.

F:

FIFO: First In, First Out.

FIPS: Federal Information Processing Standards.

I:

IBM: International Business Machines.

IEC: International Electro-technical Commission.

ISO: International Organization for Standardization**.**

J:

JCRE: Java Card Runtime Environment.

JCVM: Java Card Virtual Machine.

K:

KDC: Key Distribution Center.

L:

LAN: Local Area Network.

LCA: Linear Cellular Automata.

LTF: Local transition function.

M:

MAC: Message Authentication Code / Media Access Control.

MACA: Multiple Attractor Cellular Automata.

MCA: Memory Cellular Automata.

MD: Message Digest

O:

OCSP: Online Certificate Status Protocol.

OSI: Open Systems Interconnection.

OTP: One Time Password.

P:

PAKE: Password Authenticated Key Exchange.

PCA: Programmable cellular Automata.

PIN: Personal Identification Number.

PKCS: Public Key Cryptography Standards.

PKI: Public Key Infrastructure.

PKINIT: Public Key Cryptography for Initial Authentication.

R:

RA: Registration Authority.

RCA: Reversible Cellular Automata.

RFID: Radio-Frequency IDentification.

RPAKE: RCA-based Password Authenticated Key Exchange.

RPC: Remote Procedure Call.

RSA: Rivest-Shamir-Adleman.

S:

SCA: Singular Cellular Automata.

SHA: Secure Hash Algorithm.

SSO: Single-Sign-On.

T:

TGS: Ticket Granting Service.

TGT: Ticket Granting Ticket.

TPDU: Transmission Protocol Data Unit.

U:

USB: Universal Serial Bus.

X:

XOR: eXclusive OR.

Annexe B
Carte à puce et Java Card

2. Annexe B : Carte à puce et Java Card

2.1. La carte à puce et ses points forts

La carte à puce vise à stocker et à protéger des informations personnelles d'une simple capacité de stockage et de contrôle d'accès à ces données personnelles.

2.1.1. Les avantages de la carte à puce

L'intérêt des cartes à puce est un résultat des avantages qu'ils procurent. L'un des avantages est bien sûr intégré dans leur puissance de calcul. La sécurité, la portabilité et la facilité d'utilisation sont les autres avantages clés des cartes à puce.

Le processeur, la mémoire et le support Entré/Sortie de cartes à puce sont emballées dans un seul circuit intégré embarqué dans une carte en plastique.

La carte à puce est résistante à l'attaque car elle n'a pas besoin de dépendre des ressources externes potentiellement vulnérables. Les éléments de sécurité des cartes à puce sont encore renforcés par des fonctions cryptographiques. Les données stockées dans la carte peuvent être cryptées pour protéger sa vie privée dans la mémoire physique et les données échangées entre la carte et le monde extérieur peuvent être signés et cryptés.

En outre l'accès à une carte à puce nécessite habituellement du titulaire de la carte d'entrer le code PIN, ce qui empêche la carte de n'être utilisée que par une personne autorisée. Dans l'ensemble, il serait bien plus difficile à craquer au niveau d'une carte à puce qu'à celui d'un ordinateur de bureau traditionnel.

Un autre avantage des cartes à puce est leur portabilité inhérente. Vous pouvez porter une carte à puce dans votre portefeuille de la même manière que vous portez des cartes de crédit. En raison de cette caractéristique, les cartes à puce conservent les données disponibles chaque fois que nécessaire, en tant que détenteur de la carte se déplace d'un endroit à un autre.

Les cartes à puce sont également très pratiques à utiliser. Pour commencer une transaction, vous devez insérer la carte dans un dispositif d'acceptation des cartes, et vous retirez la carte de l'appareil lorsque le travail est fait.

En termes de fiabilité les cartes à puce ont connu un grand succès, cela est dû au fait qu'elles respectent le standard ISO 7816 définissant les contraintes environnementales que doit supporter une carte à puce (par exemple humidité, température, électricité statique, flexibilité).

2.1.2. Les domaines d'application

Les cartes à puce sont souvent utilisées pour le stockage sécurisé des données et pour authentifier et garantir les transactions. Elles ont une puissance et un impact dans le commerce et l'industrie. Cette section fournit des exemples d'applications pour l'utilisation de la carte à puce.

Dans le domaine des cartes de paiement. Toutefois, les banques ne seront pas les premières à s'intéresser à ce « système d'information » de poche. Les compagnies de téléphone y trouvent un moyen idéal pour le paiement des communications dans les cabines publiques. Les appareils ne sont plus détériorés pour en retirer la monnaie qu'ils contiennent, les communications sont payées d'avance à l'achat de la carte et constituent ainsi une réserve de trésorerie et enfin, la surface disponible sur la carte permet l'insertion de messages publicitaires.

La carte à puce est désormais utilisée dans beaucoup d'autres domaines que le paiement. Ses capacités de stockage d'informations et protection de ces informations sont mises en valeur par des applications telles que la carte de santé, renfermant des données médicales personnelles, ou les cartes de fidélité. Ses facultés d'identification et d'authentification (par le code PIN ou d'autres moyens morphologiques) sont exploités directement pour l'identification de personnes, dans une entreprise pour le contrôle d'accès ou prochainement peut-être dans la vie publique comme carte d'identité.

Enfin, les cartes à puce ont connu un nouvel essor avec la téléphonie mobile. Après l'adoption généralisée de la norme GSM et du format SIM de la carte à puce, elles sont utilisées à la fois comme élément d'identification et support applicatif. Les cartes SIM permettent alors à l'opérateur de téléphone de proposer des services personnalisés sur le téléphone.

On le voit, la carte à puce offre de plus en plus de fonctionnalités. Devant la multiplication dans la vie courante de ces cartes, il est tentant d'aboutir à des cartes pouvant faire cohabiter plusieurs services. Il s'agit ainsi de cartes multi-applicatives hébergeant divers programmes pouvant éventuellement communiquer entre eux. Mais pour gagner la confiance du public, ces cartes

intelligentes (smart card) doivent assurer, avec une architecture beaucoup plus ouverte et complexe, le même niveau de sécurité que leurs prédécesseurs.

2.1.3. Les différents types des cartes à puce

Plusieurs critères peuvent être pris en compte pour classifier les cartes à puce ; selon l'architecture, selon l'interface de communication…

2.1.3.1. Carte à mémoire et carte à micro-processeur

Les premières cartes à puce produites en grandes quantités étaient les cartes à mémoire. Les cartes à mémoire ne sont pas vraiment intelligentes parce qu'elles ne contiennent pas de microprocesseur.

Elles embarquent une puce de mémoire et une logique non programmable.

Généralement, les cartes à mémoire n'ont pas un processeur pour traiter des données, le traitement des données est effectué par un simple circuit capable d'exécuter quelques instructions préprogrammées.

Un tel circuit a des fonctions limitées et ne peut pas être reprogrammé.

A cet effet, la carte à mémoire ne peut pas être réutilisée, lorsque la valeur de la carte est dépensée, la carte est éliminée. En fonction des exigences de sécurité des données stockées, l'accès aux données peut être protégé.

Cependant, les cartes à mémoire peuvent être contrefaites facilement. Par contraste les cartes à microprocesseur, comme leur nom l'indique, contiennent un processeur. Elles offrent une sécurité considérablement augmentée et une capacité multifonctionnelle. Avec une carte à microprocesseur, les données ne sont jamais à la disposition directe des applications externes. Le microprocesseur gère le traitement des données et l'accès mémoire selon un ensemble donné de conditions (mot de passe, cryptage, etc.) et les instructions des applications externes.

En général, le terme smart card fait référence à la fois aux cartes à mémoire et aux cartes à microprocesseur. Cependant, certaines publications préfèrent nommer seulement les cartes à microprocesseur « smart card », en raison de l'intelligence fournie par le processeur embarqué. Le terme carte à puce est utilisée à la fois pour les cartes à mémoire et les cartes à microprocesseur. Dans cette thèse, le terme « smart card » se réfère aux cartes à microprocesseur.

2.1.3.2. Carte à puce avec et sans contact

Les cartes avec contact doivent être insérées dans un dispositif d'acceptation de cartes (CAD), et elles communiquent avec le monde extérieur à l'aide de l'interface de communication via huit points de contact.

Comme une carte de contact doit être insérée dans le CAD dans le bon sens et dans une orientation exacte, les cartes à puce sans contact sont populaires dans des situations nécessitant des transactions rapides. Systèmes de transport public et contrôle d'accès pour les bâtiments sont des applications excellentes pour les cartes sans contact.

Cependant les cartes sans contact ont leurs propres inconvénients. Les cartes sans contact doivent être à une certaine distance nécessaire pour échanger les données avec le CAD. Comme la carte peut se déplacer hors de la portée très rapidement, seules des données limitées peuvent être transmise en raison de la courte durée de la transaction. Il est possible que les données transmises soient interceptées sans que le titulaire de carte le sache.

Une carte à interface hybride est simplement une carte combinant les technologies sans contact et celles avec contact. Elle dispose donc de deux possibilités de communication ce qui en fait la carte idéale.

2.1.4. Les protocoles de communication

Une carte à puce classique communique avec le lecteur en mode semi-duplex, cette manière de procéder nécessite d'avoir un canal de communication pour savoir qui est autorisé à émettre et qui est autorisé à recevoir des informations. Pour parvenir à cela il faut déterminer une relation de type client/serveur, où le client initie la communication et le serveur reçoit l'information. Si ces règles ne sont pas respectées, il peut alors se produire deux situations qui ont des effets gênants lors de la communication :

- Les deux parties émettent au même moment, ce qui crée des collisions et entraine des pertes de données.
- Les deux parties se mettent en attente d'un message au même moment et dans ce cas on a une situation de Dead lock.

La communication entre la carte et le lecteur se fait suivant le modèle client/serveur où le lecteur joue le rôle de client et c'est d'ailleurs ce dernier qui fournit la carte en énergie.

La communication client/serveur se fait au travers d'une pile protocolaire que l'on retrouve dans la Figure A.1. Où nous avons représenté les différentes

couches de cette pile en m'alignant au mieux sur le modèle OSI. Dans ce modèle, à chaque couche correspond un protocole de communication. Dans la suite de cette section nous parlerons des deux premières couches à savoir la couche application et la couche transport.

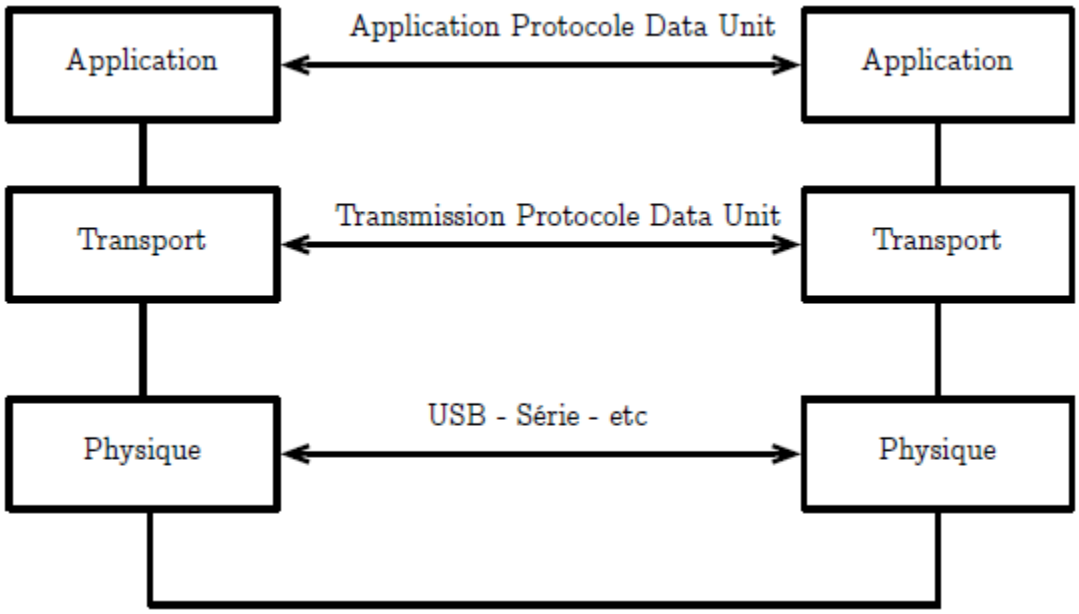


Figure A. 1 - – Pile de communication entre le lecteur et la carte

2.1.4.1. La couche physique

La couche physique est normalisée par l'ISO 7816-3 (qui définit les signaux électroniques et les protocoles de transmission). Elle définit notamment une fréquence d'horloge comprise entre 1 MHz et 5 MHz.

2.1.4.2. La couche transport - TPDU

Le protocole utilisé pour le transport des données est le protocole TPDU. Tous les échanges de données pour ce protocole sont également définis dans l'ISO 7813-3. Ce protocole définit deux modes principaux de transmission qui sont :

- le T=0 qui utilise une transmission par octet ;
- le T=1 qui utilise une transmission par paquet.

Il y a aussi certaines cartes qui utilisent le T=14 qui est réservé aux protocoles propriétaires. On trouve également des cartes qui utilisent T=CL pour les communications sans contact. Cette norme définit aussi la sélection du type de protocole si plusieurs protocoles sont disponibles ainsi que le format d'ATR qui est le message envoyé par la carte au lecteur juste après sa réinitialisation.

2.1.4.3. La couche application – APDU

Le protocole ici utilisé échange des données au format APDU qu'on retrouve également normalisé dans l'ISO 7816-3. Il existe principalement deux types d'APDU :

Tableau A. 1 - APDU de commande

Entête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Champ de données	Le

Tableau A. 2 - APDU de réponse

Corps optionnel	En-queue obligatoire	
Champ de données	SW1	SW2

- les commandes APDU qui sont émises par le CAD en direction de la carte ;
- les réponses APDU qui sont émises par la carte en direction du CAD.

Comme le modèle de communication entre le CAD et la carte est un modèle client/serveur où le CAD est le client et la carte le serveur, alors la carte est toujours en attente d'une requête en provenance du CAD sous forme d'une APDU de commande et retournera en conséquence toujours une APDU de réponse. En somme, une commande APDU sera toujours couplée avec une réponse APDU.

La commande APDU est constituée (voir le Tableau A.1) d'un entête et d'un corps. L'entête obligatoire contient quatre champs correspondant à un octet ayant une signification précise :

- CLA, l'octet de classe qui identifie la catégorie de la commande et de la réponse APDU
- INS, l'octet d'instructions qui indique l'instruction à exécuter,
- P1 et P2, deux octets contenant les paramètres de l'instruction ;

Le corps de la commande est variable et peut être omis :

- Lc, l'octet qui spécifie la taille du champ de données ;
- Le Champ de données qui contient les données à envoyer à la carte pour exécuter l'instruction spécifiée dans l'entête
- Le, l'octet correspondant au nombre d'octets attendu par le CAD pour le champ de données de la réponse APDU de la carte.

La réponse APDU est constituée des éléments du Tableau A.2

- Un corps optionnel de taille variable
- Le champ de données de taille Le déterminé dans la commande APDU correspondante.
- Une zone de fin obligatoire comportant deux octets :
- SW1 et SW2, deux champs d'un octet précisant l'état de la carte après exécution de la commande APDU. Par exemple, 0x9000 signifie que l'exécution s'est déroulée jusqu'à son terme et avec succès.

Comme certaines parties des APDU sont optionnelles, il y a quatre cas possibles d'échanges :

- Cas 1 : Aucune donnée n'est échangée entre le CAD et la carte mis à part l'entête de commande APDU et l'en queue de la réponse APDU.
- Cas 2 : Aucune donnée (dans le champ de données de la commande APDU) n'est envoyée à la carte. Le corps de la commande contient le champ Le qui spécifie le nombre d'octets que la carte doit fournir en retour dans le champ de données de la réponse APDU correspondante.
- Cas 3 : Des données de taille Lc octets sont fournies à la carte (dans le champ données de la commande) et la carte ne renvoie dans la réponse que son (en-queue).
- Des données sont échangées dans le champ de données de la commande et de la réponse APDU.

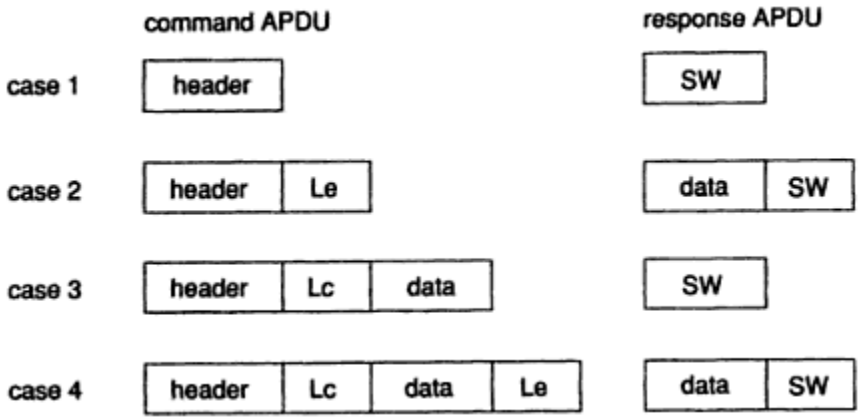


Figure A. 2 - Cas des commandes / réponses APDU.

2.2. La technologie Java pour les cartes à puces

La plateforme Java Card a pour but de faire fonctionner la technologie Java sur des équipements fortement contraints tels que les cartes à puce ou d'autres équipements avec peu de mémoire et peu de puissance de calcul. Une Java Card

est donc une carte à puce sur laquelle on est capable de charger et d'exécuter des applications Java du type applets ou servlets (depuis la version 3.0 de la spécification). Contrairement aux cartes à puce traditionnelles, ce sont des cartes ouvertes, c.-à-d. que les programmes qui y sont contenus ne sont pas nécessairement fournis par l'émetteur de la carte.

La technologie Java Card peut être considérée comme étant une plateforme fournissant un environnement sécurisé pour cartes à puce, interopérable et multi-applicative qui jouit des avantages du langage Java.

2.2.1. Les avantages de la technologie Java Card

Les avantages de Java Card comme plateforme de gestion d'applications se situent d'une part au niveau des développeurs et d'autre part au niveau des fabricants de cartes. En effet, au niveau des développeurs d'applications la plateforme Java Card apporte les mêmes avantages que le Java par rapport aux autres langages de programmation existants.

La plateforme Java Card a permis de vulgariser le modèle de développement d'applications pour cartes à puce pour les raisons qui suivent :

- Elle introduit un langage de programmation simple : facile à prendre en main, à compiler, à déboguer, et à apprendre. De plus, tous les développeurs Java sont potentiellement capables de concevoir des applications Java Card, ce qui constitue de nos jours un nombre non négligeable de personnes.
- Elle est orientée objet ce qui permet de créer, de manipuler des objets et de les faire fonctionner entre eux. Offrant ainsi une meilleure modularité ainsi que la possibilité de réutiliser simplement du code déjà écrit.
- Elle encapsule la complexité sous-jacente et les détails du système des cartes à puce qui les rendaient complexes et donc difficiles à programmer.
- Elle présente une plateforme ouverte avec des API et un environnement d'exécution standardisé.
- Il existe des outils de développement puissants aussi bien libres que commerciaux, permettant de faciliter la conception et le chargement des applications au sein de la carte.

Tout comme Java, la plateforme Java Card est indépendante du matériel sur lequel elle s'exécute. En effet, il suffit qu'une carte à puce soit certifiée Java Card pour exécuter une applet.

Java Card est une plateforme pour cartes à puce capable de faire fonctionner plusieurs applications d'origines diverses. Ainsi, une fois la carte remise à son utilisateur, il est potentiellement possible d'ajouter ou de retirer des applications de la carte ; ce qui facilite la mise à jour des programmes sans avoir à changer les cartes.

Enfin, Java Card a été conçue autour de la norme ISO 7816, lui octroyant de pouvoir supporter les applications qui sont compatibles avec cette norme. Ainsi que d'être compatible avec toutes les cartes à puce Java, mais aussi avec tous les CAD existants.

2.2.2. La machine virtuelle pour la carte à puce Java (JCVM)

Une différence principale entre la Java Card Virtual Machine (JCVM) et la Java Virtual Machine (JVM) est que la JCVM est constituée par deux entités distinctes comme le montre la figure 8, l'une est logée sur une station de travail ou un ordinateur personnel (off-card VM) et qui contient un convertisseur, et l'autre est embarquée dans la carte (on-card VM) incluant un interpréteur de bytecode.

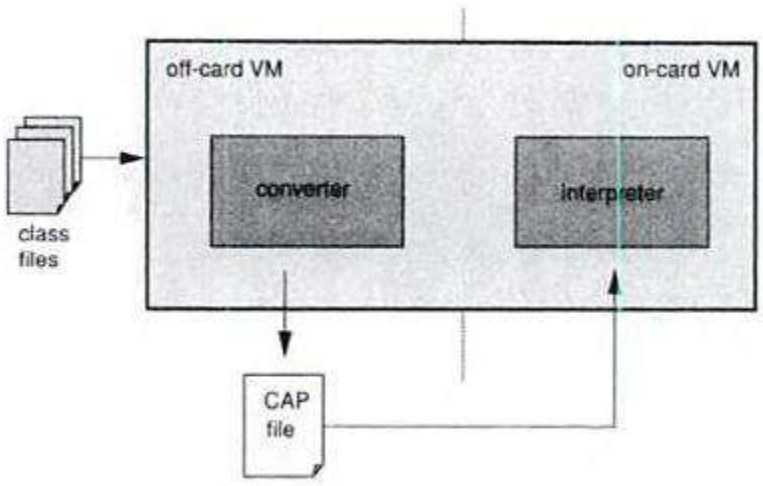


Figure A. 3 - Java Card Virtual Machine

2.2.2.1. Le convertisseur

Un ensemble de fichiers java est compilé par un compilateur (Javac) standard qui produit des fichiers (.class). Un programme nommé (converter) vérifie la comptabilité des classes avec le langage Java Card, charge et lie ces objets dans un package. Le résultat de ces opérations est stocké dans un fichier (.CAP).

2.2.2.2. L'interpréteur

Réalise l'exécution du code byte (.CAP), contrôle les allocations de mémoire et les créations d'objets.

2.2.3. L'installateur Interne (On-Card) et externe (Off-Card)

L'interpréteur Java Card ne charge pas lui-même les fichiers (.CAP), il exécute uniquement le code de ces fichiers.

L'installateur permet le téléchargement sécurisé de logiciels et d'applets sur la carte après que la carte a été produite et fournie à l'utilisateur. L'installateur interne coopère avec l'installateur externe à la carte. Ensemble, ils accomplissent la tâche de charger le contenu binaire du fichier CAP ou fichier de(s) classe(s). Cependant, l'installateur est un composant JCRE facultatif, mais sans lui tous logiciels de carte, y compris les applets, devraient être écrits dans la mémoire de la carte pendant le procédé de fabrication (Figure A.4).

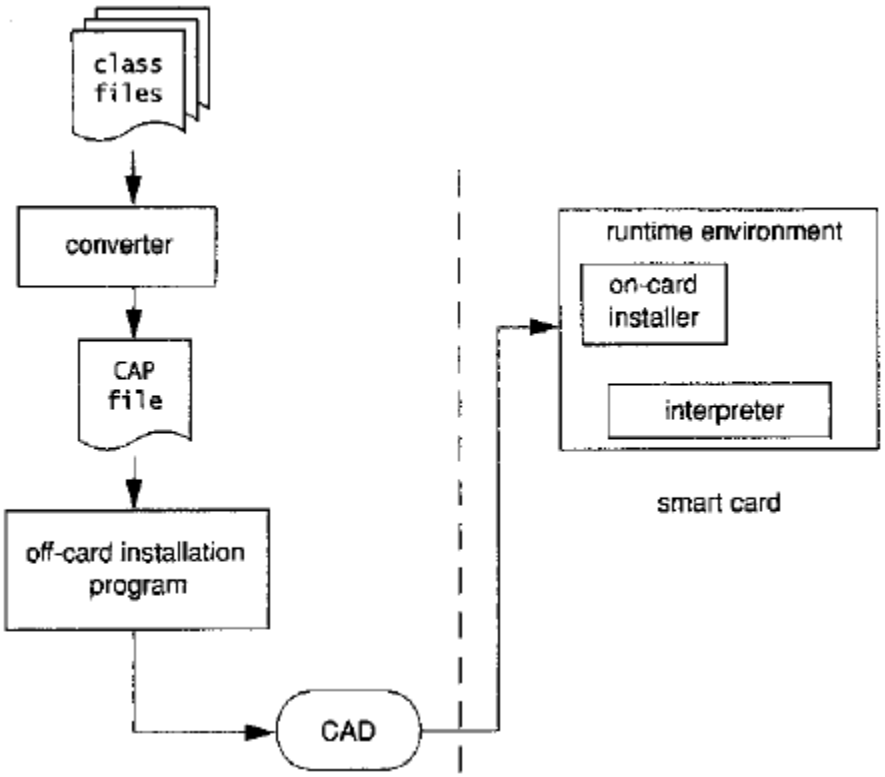


Figure A. 4 - Processus de chargement du fichier (.CAP)

2.2.4. Java Card Runtime Environment (JCRE)

Le JCRE est responsable de la gestion des ressources de la carte, des communications réseaux, de l'exécution et la sécurité des applets. Comme le montre la figure 10, le JCRE se situe au-dessus du hardware de la carte à puce et

du système natif. Le JCRE est composé de la JCVM, des APIs, d'extensions spécifiques à une industrie et des systèmes de classes.

Le JCRE distingue les applets des propriétés techniques de la carte à puce. Il fournit le système standard et les APIs pour les applets. Celles-ci sont donc plus simples à écrire et sont donc facilement portables sur différentes architectures de cartes à puce.

La couche inférieure du JCRE contient la JCVM et les méthodes natives. Elles fournissent le support à la JCVM et le système de classes pour la couche suivante. Le système de classes cadre le fonctionnement du JCRE et son rôle est similaire à un système d'exploitation. Il est responsable :

- De la gestion des transactions : Mécanisme permettant de rendre un ensemble d'opérations atomiques (ex : transaction bancaire).
- Des communications avec le CAD : La gestion des communications d'Applets, se fait via l'Application Protocol Data Unit (APDU).
- Du cycle de vie des applets : Le JCRE initialise l'applet après son installation. Il peut sélectionner alors l'applet pour lui signifier de s'exécuter, le dé sélectionner ou lui transmettre un signal APDU.

Le système de classes invoque les méthodes natives. Elles contiennent un ensemble de fonctions bas niveau qui permettent au JCRE de gérer les ressources physiques de la carte telles que les entrées-sorties, la gestion de la mémoire ou le coprocesseur cryptographique. Ces méthodes sont compilées en code exécutable dédié au processeur de la carte.

La structure de classes API définit les interfaces de programmation d'application, API. L'avantage majeur est qu'elle rend plus facile la création d'applet. Les développeurs d'applet peuvent concentrer leurs efforts sur la programmation vis-à-vis des contraintes de l'infrastructure de système de carte à puce grâce aux extensions d'API disponibles. Les applets ont accès aux services JCRE par des classes API.

La structure extension spécifique à une industrie fournit les bibliothèques complémentaires de services supplémentaires ou des modèles de système et de sécurité (ex : industries financières).

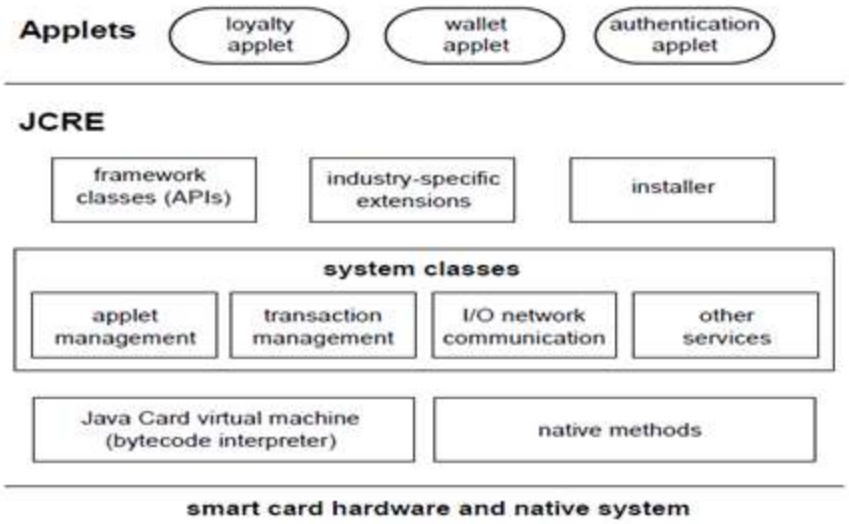


Figure A. 5 - On-card système architecture

2.2.5. Java Card APIs

Les API Java Card sont un ensemble de classes optimisées pour la programmation des cartes à puce en accord avec le modèle ISO 7816. Les API contiennent un noyau de trois packages (java.lang, javacard.framework et javacard.security) et un package d'extension (javacardx.crypto).

2.2.5.1. Le package java.lang

Ce package est un sous ensemble strict de son équivalent le package java.lang sur la plateforme Java. Ce package supporte la classe Object, qui est la superclasse de toutes les classes de langage Java et la classe Exception, qui est la superclasse pour l'exception et les classes d'exception pendant l'exécution. Donc c'est lui qui fournit le support fondamental du langage Java.

2.2.5.2. Le package javacard.framework

Ce package fournit la structure des classes et des interfaces pour le noyau fonctionnel des applets Java Card. La classe la plus importante qu'il définit est la classe de base Applet qui fournit la structure de l'exécution de l'applet en interaction avec le JCRE pendant la durée de vie de l'applet. Ce package fournit aussi la classe APDU nécessaire à la communication avec l'extérieur. Comme la classe java.lang.System de la plateforme Java n'est pas supportée, la plateforme Java Card propose le package javacard.framework.JCSystem qui fournit une interface au comportement du système. Ce package fournit d'autres classes dont entre autre la classe PIN.

2.2.5.3. Le package javacard.security

Ce package fournit les classes et les interfaces qui contiennent la fonction disponible pour implémenter une sécurité et une structure de cryptographie sur la Java Card. Les classes du paquet Javacard.security fournissent les définitions des algorithmes qui exécutent cette sécurité et les fonctions de cryptographie :

- Mise en œuvre de clés de chiffrement différentes par la classe KeyBuilder;
- Hachage des données par la classe MessageDigest ;
- Génération de données aléatoires par la classe RandomData ;
- Signature par la classe Signature ;
- Echanges de clé de session par la classe KeyAgreement.

2.2.5.4. Le package javacardx.crypto

Ce package contient la fonction qui implémente la sécurité et la structure de cryptographie sur la Java Card. C'est une extension du paquet précédent. Il contient la classe Cipher et l'interface KeyEncryption. Cipher est une classe qui fournit des méthodes pour chiffrer et déchiffrer des messages. KeyEncryption est une interface qui fournit la fonction qui permet aux clés d'être mises à jour d'une façon continue sécurisée.

2.2.6. Applets Java Card

L'applet est une application serveur, elle est sélectionnée par son identifiant unique ID, grâce à une commande APDU. Chaque applet s'exécute dans un espace qui lui est propre. Pour son interaction avec le JCRE elle doit implémenter les méthodes suivantes :

- select : Activation de l'applet sélectionnée ;
- deselect : Suspension de l'exécution de l'Applet active ;
- install / uninstall : pour installer ou supprimer une applet sur la carte ;
- process : pour traiter les commandes APDU et envoyer une réponse APDU ;
- register : L'enregistrement de l'applet auprès du JCRE.

2.2.6.1. Communication avec l'environnement extérieur

La Java Card est un module de traitement réactif à l'environnement. Une application externe qui va appeler les applets présents dans la carte. Cette application va envoyer des requêtes à la carte pour exécuter les applets qui lui sont nécessaires. Chaque applet présente dans une Java Card est spécifique au

programme client. Les applets communiquent avec l'application cliente grâce aux échanges de requêtes et de réponse par le protocole APDU.

Les cartes n'initient jamais la communication. L'application envoie une commande APDU via le lecteur (CAD). Le JCRE en recevant cette commande, sélectionne une nouvelle applet ou bien passe la commande à une applet courante (déjà sélectionnée). L'applet courante traite la commande et retourne une réponse APDU à l'application.

2.2.6.2. Cycle de vie d'une applet

Une applet est inactive tant qu'elle n'est pas sélectionnée pour être exécutée. La sélection est réalisée par le terminal. Une fois le code de l'applet chargé sur la carte et lié aux autres package se trouvant sur la carte, la vie de l'applet commence lorsqu'une instance de l'applet est créée et enregistrée au sein du JCRE. Une applet doit définir la méthode install () pour créer une instance d'applet et doit enregistrer l'instance au sein du JCRE en invoquant une des méthodes register ().

Lorsque le JCRE reçoit une commande SELECT APDU, il consulte sa table interne pour trouver l'applet dont l'ID correspond à celui spécifié dans la commande. S'il le trouve, le JCRE prépare la sélection de la nouvelle applet en deux étapes : d'abord, si une applet couramment sélectionnée est présente en mémoire, alors le JCRE la désélectionne en invoquant la méthode deselect (), l'applet l'exécute, ensuite il invoque la méthode select () pour informer la nouvelle applet de sa sélection.

La nouvelle applet effectue une opération d'initialisation avant d'être sélectionnée. L'applet retourne vrai à la méthode select () si elle est prête à devenir active. Sinon, l'applet retourne faux pour décliner sa participation. Une fois sélectionnée, le JCRE fait suivre toutes les commandes APDU (y compris SELECT) à la méthode process () de l'applet. Dans la méthode process (), l'applet interprète chaque commande APDU et exécute la tâche spécifiée par la commande. Pour chaque commande APDU, l'applet répond au CAD en envoyant une réponse APDU qui informe le CAD du résultat du traitement de la commande APDU. Ce dialogue commande-réponse continue jusqu'à ce qu'une nouvelle applet soit sélectionnée ou bien que la carte soit retirée du CAD (figure A.6).

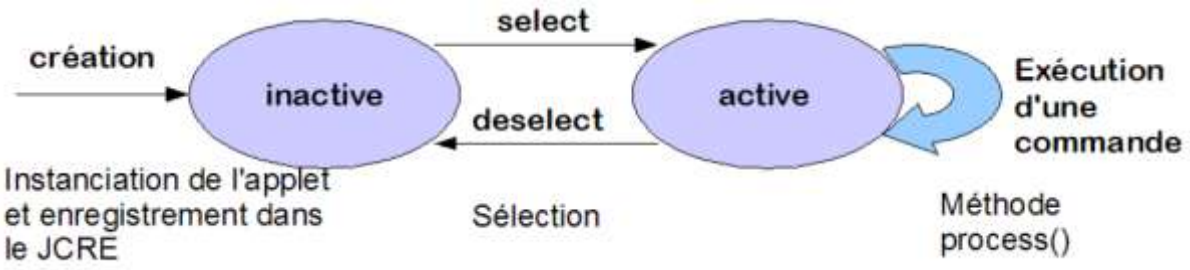


Figure A. 6 - cycle de vie d'une applet

Annexe C
Quelques définitions cryptographiques / Mathématique

3. Annexe C : Quelques définitions cryptographiques / mathématiques

3.1. Cryptanalyse différentielle

Tout d'abord il faut connaitre qu'il s'agit d'une attaque a clairs choisis. La cryptanalyse différentielle s'intéresse à l'évolution des différences $\mathbf{x_i + x'_i}$ pour deux clairs **x**, **x'**. On détermine que, si $\mathbf{x_i + x'_i = a}$, alors $\mathbf{x_{r-1} + x'_{r-1} = b}$ avec une probabilité non négligeable.

On utilise cela pour déterminer la clé inconnue $\mathbf{k_r}$ à partir de plusieurs messages **x** et de leurs chiffres $\mathbf{x_r}$ obtenus par $\mathbf{E_k}$.

Le principe général de cette attaque consiste à considérer des couples de clairs **X** et **X'** présentant une différence **ΔX** fixée et a étudier la propagation de cette différence initiale à travers le chiffrement.

On traite les couples d'entrée et de sortie comme des variables aléatoires que l'on note **X**, **Y**. **ΔX**, **ΔY**.

Les différences sont définies par une loi de groupe, en général le XOR bit à bit.

Cette attaque utilise la faiblesse potentielle de la fonction itérée **f** dans une dérivation à l'ordre 1.

La cryptanalyse différentielle utilise la comparaison du XOR de deux entrées avec le XOR des deux sorties correspondantes. On considérés **x'= (x'_1, x'_2 . . . x'_n)** et **x"= (x"_1, x"_2 . . . x"_n)** deux entrées et **y'= (y'_1, y'_2 . . . y'_n)** et **y"= (y"_1, y"_2 . . . y"_n)** les sorties correspondantes.

· On note **Δx = x'⊕ x"**

Δy = y' ⊕ y"

Si le système cryptographique était parfait alors la probabilité pour qu'un **Δy** provienne d'un **Δx** devrait être de **(½) ⁿ** ou n est le nombre de bits de **X**.

La cryptanalyse différentielle exploite le fait qu'il peut arriver qu'un **Δy** particulier arrive avec une très grande probabilité, **p ≫ (½) ⁿ**, d'un **Δx** particulier. Le couple (**Δx**, **Δy**) est appelée une différentielle.

3.2. CA additif

Cette famille d'automate cellulaire est introduite par O. Martin, A. Odlyzko et S. Wolfram [A.1]. Ces automates sont définis de la façon suivante :

- ***S = {0, 1, n-1}*** ; est l'ensemble d'alphabet fini.
- $f(s_0, s_1, \ldots) = \sum_i ai * si \pmod n$; où les coefficients a_i sont fixés et caractérisent l'automate.

L'exemple typique est celui où tous les coefficients ai sont égaux à 1 : la règle locale consiste alors à faire la somme modulo n de toutes les cellules du voisinage. Ainsi, en dimension 1, l'automate cellulaire élémentaire 90 est additif : sa fonction de transition consiste à faire la somme modulo 2 des états de la cellule et de ses deux voisines. Il suffit de connaître le comportement de cet automate sur la configuration c_0 partout égale à 0 sauf au centre où elle vaut 1 pour en déduire son comportement général par le principe de superposition (voir la figure A.7).

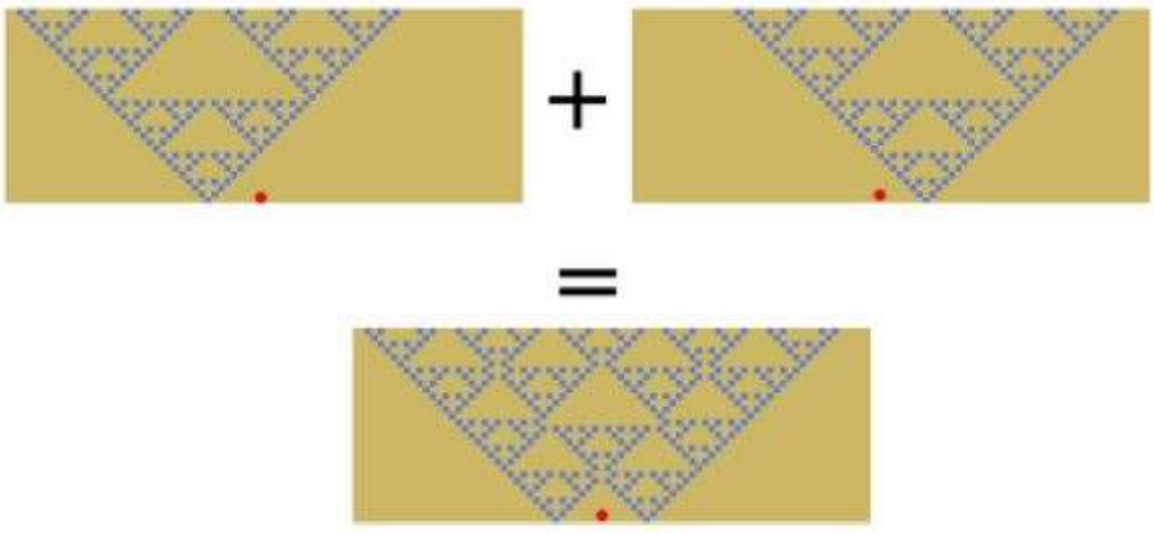


Figure A. 7 - CA additif

Bibliographie

Bibliographie

[I.1] A. W. Roscoe. The Theory and Practice of Concurrency. Prentice Hall, 1998.

[I.2] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology, 15(2):103-127, 2002.

[I.3] Roger Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. Communications of the ACM, 21(12):993-999, December 1978.

[I.4] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. Communications of the ACM, 24(8):533-536, August 1981.

[I.5] Colin Boyd. Security architectures using formal methods. IEEE Journal on Selected Areas in Communications, 11(5):694-701, 1993.

[I.6] Li Gong. Variations on the themes of message freshness and replay. In 6th IEEE Computer Security Foundations Workshop, pages 131-136. IEEE Computer Society Press, June 1993.

[I.7] Li Gong. A security risk of depending on synchronized clocks. ACM Operating Systems Review, 26(1):49-53, January 1992.

[I.8] Chris J. Mitchell. Making serial number based authentication robust against loss of state. ACM Operating Systems Review, 34(3):56-59, July 2000.

[I.9] Ulf Carlsen. Cryptographic protocol flaws - know your enemy. In 7th IEEE Computer Security Foundations Workshop, pages 192-200. IEEE Computer Society Press, June 1994.

[I.10] Stuart G. Stubblebine and Virgil D. Gligor. On message integrity in cryptographic protocols. In IEEE Symposium on Research in Security and Privacy, pages 85-104. IEEE Computer Society Press, 1992.

[I.11] Wenbo Mao and Colin Boyd. On the use of encryption in cryptographic protocols. In P. G. Farrell, editor, Codes and Cyphers - Cryptography and Coding IV, pages 251-262, 1995.

[I.12] Paul Syverson. A taxonomy of replay attacks. In 7th IEEE Computer Security Foundations Workshop, pages 187-191. IEEE Computer Society Press, June 1994.

[I.13] Mike Burmester. On the risk of opening distributed keys. In Y. Desmedt, editor, Advances in Cryptology - Crypto '94, pages 308-317. Springer-Verlag, 1994. Lecture Notes in Computer Science Volume 839.

[I.14] Ray Bird, I. Gopal, Amir Herzberg, Philippe A. Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic design of a family of attack-resistant authentication protocols. IEEE Journal on Selected Areas in Communications, 11(5):679-693, June 1993.

[I.15] P. Karn and W. Simpson. Photuris: Session-Key Management Protocol. The Internet Society, March 1999. RFC 2522.

[I.16] Catherine Meadows. A formal framework and evaluation method for network denial of service. In 12th IEEE Computer Security Foundations Workshop, pages 4-13. IEEE Computer Society Press, June 1999.

[I.17] Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium. Internet Society, February 1999.

[I.18] Dave Otway and Owen Rees. Efficient and timely mutual authentication. ACM Operating Systems Review, 21(1):8-10, January 1987.

[I.19] Thomas Aura. Strategies against replay attacks. In 10th IEEE Computer Security Foundations Workshop, pages 59-68. IEEE Computer Society Press, 1997.

[I.20] Alfred J. Menezes, Minqhua Qu, and Scott A. Vanstone. Some new key agreement protocols providing implicit authentication. In Workshop on Selected Areas in Cryptography (SAC'95), pages 22-32, 1995.

[I.21] Tsutomu Matsumoto, Youichi Takashima, and Hideki ImaI. On seeking smart public-key-distribution systems. Transactions of the IECE of Japan, (2):99-106, February 1986.

[I.22] John Kelsey, Bruce Schneier, and David Wagner. Protocol interactions and the chosen protocol attack. In B. Christianson et al., editors, Security Protocols - 5th International Workshop, pages 91-104. Springer-Verlag, 1998. Lecture Notes in Computer Science Volume 1361.

[I.23] Dieter Gollmann. What do we mean by entity authentication? In IEEE Symposium on Security and Privacy, pages 46-54. IEEE Computer Society Press, 1996.

[I.24] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. Proceedings of the Royal Society of London, A426:233-271, 1989.

[I.25] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, Advances in Cryptology - Crypto '93, pages 232-249. Springer-Verlag, 1993. Lecture Notes in Computer Science Volume 773.

[I.26] Dieter Gollmann. Insider fraud (position paper). In B. Christianson et al., editors, Security Protocols - 6th International Workshop, pages 213-226. Springer-Verlag, 1999. Lecture Notes in Computer Science Volume 1550.

[I.27] Dieter Gollmann. What do we mean by entity authentication? In IEEE Symposium on Security and Privacy, pages 46-54. IEEE Computer Society Press, 1996.

[I.28] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchange. Designs, Codes and Cryptography, 2:107-125, 1992.

[I.29] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution - the three party case. In 27th ACM Symposium on Theory of Computing, pages 57-66. ACM Press, 1995.

[I.30] ISO. Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture ISO 7498-2, 1989. International Standard.

[I.31] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.

[I.32] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22(6):644-654, November 1976.

[I.33] Boyd, C., & Mathuria, A. (2013). Protocols for authentication and key establishment. Springer Science & Business Media. ISO 690.

[I.34] I. F. Blake, G. Seroussi, and N. P. Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999.

[I.35] T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, and Roger Needham. Reducing risks from poorly chosen keys. ACM Operating Systems Review, 23(5):14-18, December 1989.

[I.36] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In IEEE Symposium on Research in Security and Privacy, pages 72-84. IEEE Computer Society Press, 1992.

[I.37] Ray Bird, I. Gopal, Amir Herzberg, Philippe A. Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic design of a family of attack-resistant authentication protocols. IEEE Journal on Selected Areas in Communications, 11(5):679-693, June 1993.

[I.38] ISO. Information Technology - Security Techniques - Entity Authentication - Part 2: Mechanisms Using Symmetric Encipherment Algorithms ISO/IEC 9798-2, 2nd edition, 1999. International Standard.

[I.39] M. Satyanarayanan. Integrating security in a large distributed system. ACM Transactions on Computer Systems, 7(3):247-280, August 1989.

[I.40] John Clark and Jeremy Jacob. Attacking authentication protocols. High Integrity Systems, 1(5):465-473, August 1996.

[I.41] Philippe Janson and Gene Tsudik. Secure and minimal protocols for authenticated key distribution. Computer Communications, 18(9):645-653, September 1995.

[I.42] Colin Boyd. Towards a classification of key agreement protocols. In 8th IEEE Computer Security Foundations Workshop, pages 38-43. IEEE Computer Society Press, 1995.

[I.43] Garman, J. (2003). Kerberos: The Definitive Guide.

[I.44] David Pointcheval & (Robert Erra), «Cryptologie et Sécurité ; Théorie, Enjeux, Pratique(s) », Livre, Université ESIEA Paris, 2000.

[I.45] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In D. Coppersmith, editor, Advances in Cryptology - Crypto '95, pages 236-247. Springer-Verlag, 1995. Lecture Notes in Computer Science Volume 963.

[I.46] ISO. Information Technology - Security Techniques - Key Management – Part 2: Mechanisms Using Symmetric Techniques ISO/IEC 11770-2, 1996. International Standard.

[I.47] Paul Syverson. Limitations on design principles for public key protocols. In IEEE Symposium on Security and Privacy, pages 62-72. IEEE Computer Society Press, 1996.

[I.48] National Institute of Standards and Technology. Entity Authentication Using Public Key Cryptography, February 1997.

[I.49] ISO. Information Technology - Security Techniques - Entity Authentication Mechanisms - Part 5: Mechanisms using Zero Knowledge Techniques ISO/IEC 9798-5, 1999. International Standard.

[I.50] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, Advances in Cryptology - Crypto '86, pages 186-194. Springer-Verlag, 1987. Lecture Notes in Computer Science Volume 263.

[I.51] L. Guillou and J.-J. Quisquater. A practical zero knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Gunther, editor, Advances in Cryptology - Eurocrypt '88, pages 123- 128. Springer-Verlag, 1988. Lecture Notes in Computer Science Volume 330.

[I.52] C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, Advances in Cryptology - Crypto '89, pages 239-252. Springer-Verlag, 1990. Lecture Notes in Computer Science Volume 435.

[I.53] Suguru Yamaguchi, Kiyohiko Okayama, and Hideo Miyahara. Design and implementation of an authentication system in WIDE internet environment. In IEEE Region 10 Conference on Computer and Communications Systems, pages 653-657, Hong Kong, September 1990.

[I.54] Tzonelih Hwang and Yung-Hsiang Chen. On the security of SPLICE/AS - the authentication system in WIDE internet. Information Processing Letters, 53:97-101, 1995.

[II.1] S. Wolfram. A New Kind of Science. Wolfram Media, Inc., US, United States, 2002.

[II.2] M. Gardner. The Fantastic Combinations of John Conway's New Solitaire Game `Life'. Sci. Am., 223:120{123, 1970.

[II.3] S. Wolfram. Theory and Applications of Cellular Automata. World Scientific, Singapore, 1986. ISBN 9971-50-124-4 pbk.

[II.4] Bakhshandeh, A., & Eslami, Z. (2013). An authenticated image encryption scheme based on chaotic maps and memory cellular automata. Optics and Lasers in Engineering, 51(6), 665-673.

[II.5] Del Rey, Á. M. (2005, June). Design of a Cryptosystem Based on Reversible Memory Cellular Automata. In ISCC (pp. 482-486).

[II.6] Faraoun, K. M. (2014). Design of fast one-pass authenticated and randomized encryption schema using reversible cellular automata. Communications in Nonlinear Science and Numerical Simulation, 19(9), 3136-3148.

[II.7] Von Neumann, J. (1951). The general and logical theory of automata. Cerebral mechanisms in behavior, 1(41), 1-2.

[II.8] Neumann, J. V., & Burks, A. W. (1966). Theory of self-reproducing automata.

[II.9] Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. ACM SIGACT News, 32(1), 60-65.

[II.10] Manning, F. B. (1977). An approach to highly integrated, computer-maintained cellular arrays. Computers, IEEE Transactions on, 100(6), 536-552.

[II.11] Preston Jr, K., Duff, M. J., Levialdi, S., Norgren, P. E., & Toriwaki, J. I. (1979). Basics of cellular logic with some applications in medical image processing. Proceedings of the IEEE, 67(5), 826-856.

[II.12] Atrubin, A. J. (1965). A one-dimensional real-time iterative multiplier. Electronic Computers, IEEE Transactions on, (3), 394-399.

[II.13] Cole, S. N. (1966, October). Real-time computation by n-dimensional iterative arrays of finite-state machines. In Switching and Automata Theory, 1966. IEEE Conference Record of Seventh Annual Symposium on (pp. 53-77). IEEE.

[II.14] Nishio, H. (1981). Real time sorting of binary numbers by one-dimensional cellular automata. Technical Report, Kyoto University.

[II.15] Fischer, P. C. (1965). Generation of primes by a one-dimensional real-time iterative array. Journal of the ACM (JACM), 12(3), 388-394.

[II.16] Deutsch, E. S. (1972). Thinning algorithms on rectangular, hexagonal, and triangular arrays. Communications of the ACM, 15(9), 827-837.

[II.17] Sternberg, S. R. (1980). Language and architecture for parallel image processing. Pattern recognition in practice, 35.

[II.18] Toffoli, T. (1977). Cellular automata mechanics.

[II.20] Delorme, M. and Mazoyer, J. [1999], Cellular Automata, A Parallel Model, Kluwer Academic Publishers.

[II.21] S. Wolfram. Cryptography with cellular automata. In Advances in Cryptology - CRYPTO '85, volume 218, pages 429-432. Springer-Verlag, 1986.

[II.22] S. Wolfram. Random sequence generation by cellular automata. Advances in Applied Mathematics, 7(2):163-169, 1986.

[II.23] W. Meier and O. Staffelbach. Analysis of pseudo random sequences generated by cellular automata. Proceedings of Eurocrypt '91, pages 186-199, 1991.

[II.24] H. Gutowitz. Cryptography with dynamical systems. Cellular Automata and Cooperative Phenomena, Eds: E. Goles and N. Boccara, pages 237-274, 1993.

[II.25] H. Gutowitz. Method and apparatus for encryption, decryption, and authentication using dynamical systems. U.S. Patent 5365589, 1994.

[II.26] E. Biham and A. Shamir. Di differential cryptanalysis of des-like cryptosystems. In CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, pages 2-21, London, UK, 1991. Springer-Verlag.

[II.27] M. Tomassini and M. Perrenoud. Stream cyphers with one- and two-dimensional cellular automata. In PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, pages 722-731, London, UK, 2000. Springer-Verlag.

[II.28] P. Bouvry, F. Seredynski, and A. Y. Zomaya. Application of cellular automata for cryptography. In PPAM, pages 447-454, 2003.

[II.29] M. Seredynski and P. Bouvry. Block cipher based on reversible cellular automata. Evolutionary Computation, 2004. CEC2004. Congress on, 2:2138-2143 Vol.2, 19- 23 June 2004.

[II.30] Petre Anghelescu, Silviu Ionita, and Emil Sofron. Block encryption using hybrid additive cellular automata. In HIS '07: Proceedings of the 7th International Conference on Hybrid Intelligent Systems, pages 132-137, Washington, DC, USA, 2007. IEEE Computer Society.

[II.31] S. Nandi, B.K. Kar, and P. P. Chaudhuri. Theory and applications of cellular automata in cryptography. IEEE Transactions on Computers, 43(12):1346-1357, 1994.

[II.32] S. Sen, C. Shaw, D. R. Chowdhuri, N. Ganguly, and P. P. Chaudhuri. Cellular automata based cryptosystem (cac). In ICICS '02: Proceedings of the 4th International Conference on Information and Communications Security, pages 303-314, London, UK, 2002. Springer-Verlag.

[II.33] F. Bao. Cryptanalysis of a partially known cellular automata cryptosystem. IEEE Trans. Comput., 53(11):1493-1497, 2004.

[II.34] S. R. Blackburn, S. Murphy, and K. G. Paterson. Comments on "theory and applications of cellular automata in cryptography". IEEE Trans. Comput., 46(5):637-638, 1997.

[II.35] R. Tao and S. Chen. On finite automaton public-key cryptosystem. Theoretical Computer Science, 226(1-2):143-172, 1999.

[II.36] P. Guan. Cellular automata public key cryptosystem. Complex Systems, 1:51-57, 1987.

[II.37] J. Kari. Cryptosystems based on reversible cellular automata. Manuscript, 1992.

[II.38] S. Wolfram. Computation theory of cellular automata. Communications in Mathematical Physics, 96(1):15-57, 1984.

[II.39] W. Li, N.H. Packard, and C.G. Langton. Transition phenomena in cellular automata rule space. Physica D: Nonlinear Phenomena, 45(1-3):77-94, 1990.

[II.40] W. Li and N. Packard. The structure of the elementary cellular automata rule space. Complex Systems, 4:281-297, 1990.

[II.41] P. Kurka. Topological and symbolic dynamics. Collection SMF. Société Mathématique de France, 2003.

[II.42] P. Kurka. Languages, equicontinuity and attractors in cellular automata. Ergodic Theory and Dynamical Systems, 17:417-433, 4 1997.

[II.43] K. Culik III and S. Yu. Undecidability of CA classification schemes. Complex Systems, 2(2):177-190, Apr 1988.

[II.44] G.A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. Mathematical systems theory, 3(4):320-375, 1969.

[II.45] D. Richardson. Tessellations with local transformations. Journal of Computer and System Sciences, 6(5):373-388, 1972.

[II.46] S. Amoroso and Y.N. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. Journal of Computer and System Sciences, 6(5):448-464, October 1972.

[II.47] J. Kari. Reversibility of 2D cellular automata is undecidable. Physica D: Nonlinear Phenomena, 45(1-3):379-385, 1990.

[II.48] J. Kari. Reversibility and surjectivity problems of cellular automata. Journal of Computer and System Sciences, 48(1):149-182, 1994.

[II.49] T. Toffoli and N.H. Margolus. Invertible cellular automata: A review. Physica D: Nonlinear Phenomena, 45(1-3):229-253, 1990.

[II.50] G.Y. Vichniac. Simulating physics with cellular automata. Physica D: Nonlinear Phenomena, 10(1-2):96-116, 1984.

[II.51] Damgård, I., "A Design Principle for Hash Functions," Advances in Cryptology Crypto 89, Lecture Notes in Computer Science, vol. 435, p. 416427, 1990.

[II.52] Daemen, J., Govaerts, R., and Vandewalle, J., "A Framework for the Design of One-Way Hash Functions Including Cryptanalysis of Damgard's One-Way Function Based on a Cellular Automaton," Advances in Cryptology Asiacrypt 91, Lecture Notes in Computer Science, vol. 739, p. 82-96, 1993.

[II.53] Mihaljevic, M., Zheng, Y., and Imai, H., “A Cellular Automaton Based Fast One-way Hash Function Suitable for Hardware Implementation,” Lecture Notes in Computer Science, vol. 1431, p. 217-233. Springer-Verlag, 1998.

[II.54] Dasgupta, P., Chattopadhyay, S., and Sengupta, I., “An ASIC for Cellular Automata Based Message Authentication,” 12th International Conference on VLSI Design, 1999.

[II.55] Mukherjee, M., Ganguly, N., and Chaudhuri, P.P., “Design of Cellular Automata Based Message Authentication” International Conference on Computer Communication, Mumbai, 2002.

[II.56] Lee, M., Hong, D, and Kim, D., "Cryptanalysis of Mukherjee-Ganguly-Chaudhuri's Message Authentication Scheme," Computational Intelligence and Security, 2006 International Conference on, vol. 2, p. 1311-1314, Nov. 3-6 2006.

[II.57] Zhang, X., Lu, R., Zhang, H., & Xu, C. (2014). A New Public Key Encryption Scheme based on Layered Cellular Automata. TIIIS, 8(10), 3572-3590.

[III.1] Lamport, L. (1981). Password authentication with insecure communication. Communications of the ACM, 24(11), 770-772.

[III.2] Chang CC, Wu TC. Remote password authentication with smart cards. IEE Proceedings-Computers and Digital Techniques 1991; 138(3):165–168.

[III.3] Fan, C., Chan, Y., Zhang, Z.: Robust remote authentication scheme with smart cards. Comp & Sec 24(8) (2005) 619-628.

[III.4] Juang, W.S.: Efficient multi-server password authenticated key agreement using smart card. IEEE Trans. On Cons. Ele 50(1) 2004 251-255.

[III.5] Shen, J. J., Lin, C. W., Hwang, M. S.: A modified remote user authentication scheme using smart cards. IEEE Trans. on Cons. Ele. 49(2) (2003) 414-416.

[III.6] Leu, Jenq-Shiou, and Wen-Bin Hsieh. Efficient and secure dynamic ID-based remote user authentication scheme for distributed systems using smart cards. Information Security, IET 8.2 (2014): 104-113.

[III.7] Li, X., Liao, J., Zhang, J., Niu, J., & Kumari, S. (2014, October). A secure remote user mutual authentication scheme using smart cards. In Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE (pp. 89-92). IEEE.

[III.8] Pippal, R. S., Jaidhar, C. D., & Tapaswi, S. (2013, February). A novel smart card mutual authentication scheme for session transfer among registered devices. In Advance Computing Conference (IACC), 2013 IEEE 3rd International (pp. 1540-1547). IEEE.

[III.9] Chen, Bae-Ling, Wen-Chung Kuo, and Lih-Chyau Wuu. Robust smart-card-based remote user password authentication scheme. International Journal of Communication Systems 27.2 (2014): 377-389.

[III.10] Tripathy, S., Chowdhury, A. R., & Nandi, S. (2006). CARA: Cellular Automata Based Remote-User Authentication Scheme. In Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on (pp. 1-4). IEEE.

[III.11] Tripathy, S., Nandi, S., & Chowdhury, A. R. (2006, December). CARMA: Cellular Automata Based Remote Mutual Authentication Scheme. In Information Technology, 2006. ICIT'06. 9th International Conference on (pp. 190-191). IEEE.

[III.12] Jeon, J. C., & Yoo, K. Y. (2006). Authentication based on singular cellular automata. In Cellular Automata (pp. 605-610). Springer Berlin Heidelberg.

[III.13] Seo, K. J., Jeon, J. C., Shin, S. H., & Yoo, K. Y. (2008, July). CA Based RFID Authentication Protocol for Privacy and Anonymity. In Wireless and Mobile Communications, 2008. ICWMC'08. The Fourth International Conference on (pp. 278-281). IEEE.

[IV.1] Baghor, S., & Faraoun, K. M. (2016). A Novel Fast and Secure Mutual Remote User Authentication Protocol with Session Key Agreement Using Second Order Cellular Automata.

[V.1] Bhattacharryya, D. K., & Nandi, S. (2000). CA based password-only authenticated key exchange. In Signal Processing Systems, 2000. SiPS 2000. 2000 IEEE Workshop on (pp. 820-827). IEEE.

[V.2] Tripathy, S., & Nandi, S. (2006, September). RPAKE: RCA-based Password Authenticated Key Exchange. In 2006 Annual IEEE India Conference (pp. 1-5). IEEE.

[V.3] Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2012). CA-KEP: A Secure CA Based 2-Party Key Exchange Protocol. Journal of Information Assurance and Security, 7(3), 193-206.

[V.4] M Mihaljevid, Y Zhang and H Imai, "A family of fast dedicated one-way hash function based on Linear CA over GF(Q)," in IEICE Transactions Fundamentals, Vol. E82 - A, No.1, pp.1 - 8, Jan. 1999.

[V.5] D P Jablon , ”Strong Password-Only Authenticated Key Exchange ”, in ACM SIGCOMM, Computer Comm. Review, pp 5-26, 1997.

[V.6] D K Bhattacharyya, et al. ”Efficient One-way Hash Function Design using Cellular Automata”, in Proc. of the 4th Int’nl Conf. on Advanced Computing, Chennai, 1997.

[V.7] B Schneier, Applied Cryptography, 2nd Ed, John Wiley & Sons, 1996.

[V.8] D.P. Jablon, "Strong password-only Authenticated Key Exchange," ACM SIGCOMM computer Comm Review pp. 5-26, 1997.

[V.9] V. Boyko, P. MacKenzie and S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman," Proc. Eurocrypt 2000, pp. 156-171.

[V.10] P. MacKenzie, S. Patel and R. Swaminathan, “Password authenticated key exchange based on RSA” Proc. Asiacrypt 2000, - pp. 599-613.

[V.11] J. Kari, "Reversibility and surjectivity problems of cellular automata," Journal of Computer and System Sciences, vol. 48(1), pp. 149-182, 1994.

[V.12] S. Amoroso, Y. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, Journal of Comput. System Sci. vol. 6, pp. 448, 1972

[V.13] S M Bellovin, M Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 72–84, 1992.

[V.14] R Lu, Z Cao. Simple Three-party Key Exchange Protocol. Computers & Security, 26(1), pp. 94–97, 2007.

[V.15] H F Huang. A Simple Three-party Password-based Key Exchange Protocol. International Journal of Communication Systems, 22(7), pp. 857–862, July 2009.

[V.16] H S Kim, J Y Choi. Enhanced Password-based Simple Three-party Key Exchange Protocol. Computers and Electrical Engineering, 35(1), pp. 107–114, January 2009.

[V.17] D C Lou, H F Huang. Efficient Three-party Password based Key Exchange Scheme. International Journal of Communication Systems, 24(4), pp. 504–512, 2011.

[VI.1] Kari, J. (1992). Cryptosystems based on reversible cellular automata. Manuscript, August.

[VI.2] P P Choudhury, D R Choudhury, S Nandi, S Chattopadhay. Additive Cellular Automata: Theory and Application, Vol. I. IEEE, 1997.

[VI.3] J. Daemenn., R. Govaerts and J. Vandewalle, "A framework for the Design of One-Way Hash Function Based On a Cellular Automaton," LNCS 739, Asiacrypt 91, 1993.

[VI.4] D K Bhattacharyya, et al. ”Efficient One-way Hash Function Design using Cellular Automata”, in Proc. of the 4th Int’nl Conf. on Advanced Computing, Chennai, 1997.

[VI.5] S. Amoroso, Y. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, Journal of Comput. System Sci. vol. 6, pp. 448, 1972.

[A.1] Martin, O., Odlyzko, A. M., & Wolfram, S. (1984). Algebraic properties of cellular automata. Communications in mathematical physics, 93(2), 219-258.