Rui Alexandre Cardoso Ferreira

Protocolos de segurança em redes sem fios

Departamento de Matemática Pura

Faculdade de Ciências da Universidade do Porto

Dezembro de 2006

Rui Alexandre Cardoso Ferreira

Protocolos de segurança em redes sem fios

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Engenharia Matemática.

Departamento de Matemática Pura

Faculdade de Ciências da Universidade do Porto

Dezembro de 2006

"Three may keep a secret, if two of them are dead."

Benjamin Franklin

Aos meus pais

Agradecimentos

Agradeço profundamente aos meus orientadores, Prof. Doutor Jorge Almeida e António Machiavelo, pelo apoio inestimável que me prestaram durante a elaboração desta dissertação. Sublinho o indiscutível apoio científico que me facultaram, tendo sido um autêntico privilégio trabalhar com eles.

Agradeço também à minha família e a todos os meus amigos, que sempre tiveram palavras de incentivo nos bons e maus momentos que um trabalho destes sempre comporta.

Resumo

As denominadas *redes sem fios* ("wireless networks") estão actualmente na "moda", no sentido em que existe cada vez mais um maior número de utilizadores deste modo de comunicação. Numa troca de informação entre partes, em geral, é imperativo a confidencialidade e autenticidade das mensagens. Ao longo desta dissertação iremos descrever/estudar os protocolos de segurança existentes no mercado actual das redes sem fios.

Abstract

The usually called *wireless networks* are currently in "fashion" in the sense that there is a large number of users within this type of comunication. In an exchange of any kind of information, generally, it is imperative to have confidentiality and authenticity of the messages. Throughout this dissertation, we will describe/study the security protocols that are implemented on the wireless devices that exist in the market.

Conteúdo

Introdução

Emissões de rádio e, posteriormente, emissões televisivas definiram o mundo sem fios ("wireless") durante duas gerações. Esta capacidade das ondas de rádio e dos sinais de TV "chegarem a todo o lado" e poderem ser ouvidas e vistos por qualquer pessoa, trouxe grandes benefícios ao público em geral desde o início do século vinte. Do ponto de vista de um *receptor*, esta capacidade de emissão ("broadcast") é muito atractiva, embora para um *emissor* possa trazer grandes desvantagens.

Os militares foram pioneiros na utilização das ondas de rádio para comunicar e consequentemente os primeiros a serem afectados pelo facto de poderem ser ouvidos por "toda a gente". De modo a protegerem as suas comunicações por rádio, os militares adaptaram códigos secretos que já vinham sendo usados durante vários anos para protegerem mensagens escritas. Durante a Guerra Fria (1950 a 1980), grandes avanços foram feitos na elaboração de medidas de segurança para os canais de comunicação.

Devido às melhorias verificadas nas tecnologias *wireless*, assim como à sua descida no preço, hoje em dia quase toda a gente as utiliza, por exemplo, em telemóveis ou em computadores portáteis através de uma denominada *rede sem fios* ("Wireless Network").

Esta dissertação tem como principal objectivo o estudo dos protocolos de segurança usados nas denominadas[1] *Wi-Fi LANs*. Este modo de operação é bastante utilizado pelas pessoas para estabelecerem uma ligação entre computadores, como por exemplo em empresas ou nas suas próprias casas. Tipicamente, um cartão (específico para comunicações *wireless*) é inserido num computador de modo a que este possa enviar mensagens para outros computadores ou para a Internet via um canal de rádio de baixo alcance para um *ponto de acesso*. Isto significa que uma pessoa pode trabalhar na sua

[1] *Wireless LAN* é o termo usado para redes utilizando ondas rádio de baixo alcance e alta velocidade. Wi-Fi é um tipo de *wireless LAN*.

secretária ou numa sala de conferências, no seu escritório em casa ou no seu quarto, etc... sem o inconveniente de "andar com os fios atrás".

Conduzidos pelo nosso objectivo, este trabalho divide-se em três partes essenciais:

1) O primeiro modo de segurança a ser implementado nas redes Wi-Fi, designado por WEP, tinha como principal objectivo proteger as mensagens, conferindo-lhes autenticidade e confidencialidade. Tinha também a missão de impedir o acesso à rede de pessoas não desejadas, isto é, pessoas sem o direito legítimo de pertencerem a uma determinada comunicação.

   Nesta primeira parte, introduziremos os conceitos necessários para uma boa descrição de como o WEP era implementado para os objectivos a que se propôs. Seguidamente, estudaremos alguns ataques a que o WEP está sujeito, concluindo com uma descrição detalhada do ataque mais "feroz" [5] (este ataque permite recuperar a chave secreta), que foi devidamente implementado com sucesso em 2001 por Stubblefield, Ioannidis e Rubin [16].

2) Numa segunda parte, iremos descrever o que foi feito tendo em vista a eliminação dos problemas existentes no WEP. Descreveremos, em detalhe, as novas soluções de segurança: TKIP e CCMP-AES.

3) Finalmente, apresentamos e discutimos as ideias presentes num artigo que pretende apontar algumas fragilidades existentes num dos novos métodos de segurança, nomeadamente no TKIP.

Ao leitor indicamos que deverá estar familiarizado com alguma terminologia[2] das áreas da *Criptografia* e da *Computação*. Deve também ser possuidor de formação em Matemática, nomeadamente, ter conhecimentos de *Teoria dos Números* e *Probabilidades* ao nível de uma licenciatura.

[2]Em alguns casos, quando tal se justifique, serão dadas referências para consulta de determinada terminologia.

Capítulo 1

O método de segurança WEP

1.1 O protocolo IEEE 802.11

O estabelecimento de uma ligação entre um utilizador e uma *rede LAN* ("Local Area Network") é feito através de uma sequência de componentes de *hardware* e *software*, que estão conectadas através de um interface claramente definido. A sua implementação numa rede é definida pelo conceito de *camadas* ("layers"). Cada camada tem uma determinada função e é responsável por determinadas actividades. As camadas "perto" do utilizador são denominadas de *camadas superiores* e as camadas "perto" da LAN de *camadas inferiores*.

O termo *rede sem fios LAN* ("Wireless LAN") refere-se genericamente às camadas mais inferiores da rede, isto é, às camadas de ligação (físicas). A norma IEEE[1] 802 lida com estas camadas para um vasto número de diferentes tecnologias LAN. O IEEE 802.11 define a especificidade de se tratar de uma rede sem fios.

O IEEE 802.11 possui dois modos de operação no que diz respeito à comunicação entre aparelhos *wireless*:

Infra-estrutura ("Infrastructure mode") Este modo de operação prevê um ponto de acesso ("Access Point") que coordena a rede sem fios, estando na maioria das vezes conectado a uma rede com fios.

Ad-hoc Este modo de operação não necessita de pontos de acesso, pois cada dispositivo

[1]IEEE é a abreviatura para Institute of Electrical and Electronics Engineers.

wireless transmite directamente para qualquer outro.

A maioria das pessoas utiliza o modo infra-estrutura, pois dessa maneira poderão conectar-se a uma infra-estrutura com fios, como por exemplo, uma ligação de Internet. É nosso intuito neste trabalho descrever/estudar os protocolos de segurança implementados no IEEE 802.11 (no modo de operação infra-estrutura). O leitor interessado em estudar mais aprofundadamente o IEEE 802.11, poderá consultar por exemplo [13].

1.1.1 O WEP no IEEE 802.11

Quando foi concebido o IEEE 802.11 para redes (locais) sem fios, este apenas definia um modo de segurança, designado por *WEP* ("Wired Equivalent Privacy"), que como o próprio nome indica foi concebido para garantir um nível de segurança equivalente ao de uma rede com fios.

Vamos começar por fazer uma breve descrição do WEP (para uma descrição mais detalhada, ver [7]), que nos permitirá fazer uma discussão dos ataques que lhe são inerentes.

O WEP faz uso de uma chave secreta (habitualmente de 104 bits[2]) distribuída pelas partes que querem comunicar. A forma como esta chave é distribuída não é especificada em IEEE 802.11 (ver [4], pág. 77). O protocolo permite ainda trocar esta chave, embora numa determinada sessão esta chave seja a mesma para o envio dos diferentes pacotes. Dada a importância de que a chave seja diferente em cada transmissão (que adiante será melhor compreendida), o WEP define um *vector de inicialização* (IV) de 24 bits, o que significa que no total a chave terá 128 bits.

Vamos ver agora de que consiste a cifragem feita pelo WEP:

- Primeiro, é calculado um valor [3] de 32 bits de verificação de integridade da mensagem, denotado por $c(M)$ [o algoritmo utilizado neste cálculo é um CRC ("Cyclic

[2]O IEEE 802.11 só especificou chaves de 40 bits devido a restrições de exportação impostas pelo governo dos Estados Unidos (note-se que chaves com este número de bits são susceptíveis a um ataque de força bruta). No entanto, muitos fabricantes de sistemas de 802.11 fizeram versões não-*standard*, usando chaves de 104 bits, no intuito de fortalecerem a sua segurança [4].

[3]Neste trabalho também designaremos este valor por *soma de verificação de integridade*.

Redundancy Check") [7]]. Seguidamente, concatena-se esse valor com a mensagem, obtendo o texto a ser cifrado ("Plaintext") $P = \langle M, c(M) \rangle$. Note-se que $c(M)$, e portanto P, não dependem da chave secreta.

- Neste segundo passo, ciframos P usando o algoritmo RC4 (ver secções 1.2.1 e 1.2.3). O algoritmo gera uma sequência finita de bytes pseudo-aleatórios ("Keystream") em função do IV e da chave secreta SK, que denotamos por RC4(IV, SK).

- Finalmente, fazemos o XOR[4] (denotado por $\oplus$) de P com RC4(IV, SK) para obter o texto cifrado ("Chiphertext")

$$C = P \oplus \text{RC4}(IV, SK),$$

que é posteriormente transmitido junto com o IV, via ondas de radio.

Observação: O número de bytes gerado pelo RC4 depende do número de bytes que tem cada pacote - normalmente varia entre 100 e 1500.

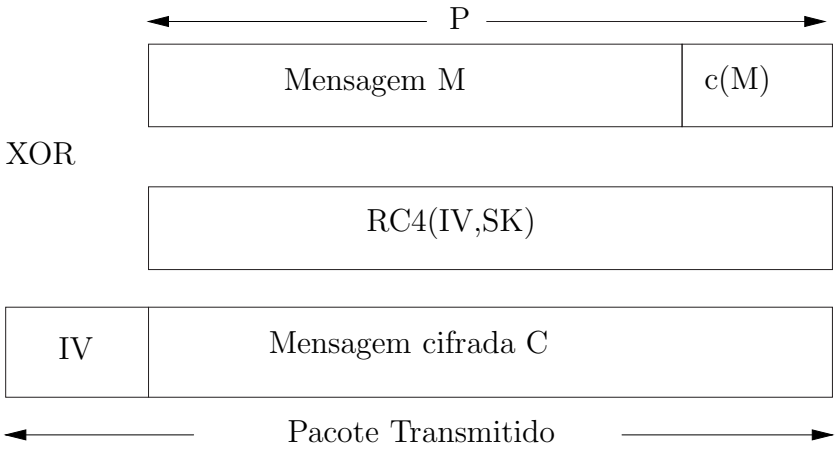


Figura 1.1: Pacote cifrado pelo WEP

De modo a obter a mensagem original, o receptor calcula a sequência pseudo-aleatória

[4]A operação XOR combina dois bytes e obtém outro da seguinte maneira: compara bits correspondentes em cada byte e, se eles forem iguais, o resultado é zero; se forem diferentes, o resultado é um. Por exemplo $00110101 \oplus 11100011 = 11010110$.

de bytes RC4(IV, SK) e faz XOR com o texto cifrado, obtendo

$$\begin{aligned} P' &= C \oplus \text{RC4}(IV, SK) \\ &= (P \oplus \text{RC4}(IV, SK)) \oplus \text{RC4}(IV, SK) \\ &= P \oplus (\text{RC4}(IV, SK) \oplus \text{RC4}(IV, SK)) \\ &= P. \end{aligned}$$

Finalmente, o receptor separa P' na forma $\langle M', c'(M') \rangle$, calcula $c(M')$ e verifica que coincide com $c'(M')$. Isto assegura que só pacotes com soma de verificação de integridade válida serão aceites pelo receptor.

1.2 Ataques ao WEP

1.2.1 A cifra sequencial RC4

Em 1987 Ron Rivest projectou a cifra sequencial RC4 [5], que é utilizada em aplicações de *software* e que é de muito simples implementação em *hardware*. O algoritmo foi tornado público em 1994 por alguém que permaneceu no anonimato. Pelo facto do RC4 ser utilizado em diversos protocolos de segurança e ser uma cifra extremamente simples, atraiu a atenção de diversos investigadores na área, mas até agora ninguém descobriu um ataque ao RC4 que esteja perto de ser prático.

O RC4 consiste de duas fases (ver fig.1.2 em baixo): a primeira, que se denota por KSA ("Key-Scheduling Algorithm"), que transforma uma chave K (cujo comprimento normalmente varia entre 5 e 32 bytes e é representado por l) numa permutação S do conjunto dos bytes $\{0, \ldots, N-1\}$, em que $N = 256$; a segunda, que se denota por PRGA ("Pseudo-Random Generation Algorithm"), que usa S para gerar uma sequência pseudo-aleatória de bytes.

1.2.2 Os ataques de Borisov, Goldberg e Wagner

[5] RC4 significa a quarta cifra projectada por Ron Rivest ("Rivest Chipher 4").

KSA(K)	PRGA(K)
Inicialização:	Inicialização:
For $i = 0, \ldots, N-1$	$i = 0$
$S[i] = i$	$j = 0$
$j = 0$	Geração do byte:
Mistura:	$i = i + 1$
For $i = 0, \ldots, N-1$	$j = j + S[i]$
$j = j + S[i] + K[i \bmod l]$	$S = S \circ (i, j)$
$S = S \circ (i, j)$	Byte pseudo-aleatório $= S[S[i] + S[j]]$

Figura 1.2: O Key-Scheduling Algorithm e o Pseudo-Random Generation Algorithm (Todas as somas são feitas mod N e $S \circ (i, j)$ denota a permutação das posições i e j em S).

Antes de descrever os ataques em questão [2], começaremos com uma pequena nota: o objectivo deste trabalho é descrever/estudar as propriedades criptográficas dos protocolos de segurança implementados em redes sem fios. Portanto, não estamos aqui interessados em estudar a dificuldade em montar estes ataques na prática. Diremos apenas que podemos assumir que "atacantes" motivados conseguem ter acesso às transmissões feitas via ondas de rádio, encerrando por aqui esse assunto.

Os três principais objectivos a serem alcançados pelo WEP são:

Confidencialidade O protocolo de segurança deve ser capaz de evitar que um "intruso", ou seja, um indivíduo que não esteja autorizado a participar da comunicação, seja capaz de ler dados transmitidos.

Controle de acesso O protocolo deve evitar que um individuo "não desejado" tenha acesso à rede sem fios. A norma 802.11 inclui a opção de podermos simplesmente descartar todos os pacotes que não cheguem cifrados pelo WEP. Isto em princípio garante que só utilizadores de posse de uma chave secreta WEP possam fazer parte da comunicação.

Integridade dos dados transmitidos O protocolo deve garantir que o conteúdo das mensagens se mantém inalterado durante a transmissão. O WEP utiliza a função soma de verificação de integridade para o almejar.

Vamos no entanto ver que nenhum destes três objectivos foi eficientemente alcançado.

1.2.2.1 O risco da reutilização de RC4(IV, SK)

Um dos objectivos a atingir pelo protocolo de segurança WEP é o da confidencialidade dos dados a transmitir; para isso utiliza a cifra sequencial RC4.

Suponhamos que duas mensagens M_1 e M_2 são cifradas usando o mesmo vector de inicialização IV. Ficamos assim com duas mensagens cifradas C_1 e C_2, tais que:

$$C_1 = P_1 \oplus \text{RC4}(IV, SK)$$
$$C_2 = P_2 \oplus \text{RC4}(IV, SK)$$

Fazendo XOR entre C_1 e C_2, obtemos:

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus \text{RC4}(IV, SK)) \oplus ((P_2 \oplus \text{RC4}(IV, SK))) \\ &= P_1 \oplus P_2. \end{aligned}$$

Por outras palavras, fazendo XOR de dois pacotes cifrados com o mesmo IV, obtemos o XOR dos dois que haviam sido cifrados $P_1 \oplus P_2$. Isto conduz a alguns ataques: por exemplo, se conhecermos P_1 (analogamente para P_2), obtemos facilmente P_2, uma vez que $P_1 \oplus P_1 \oplus P_2 = P_2$. Mas, conhecer P_1 não é em geral possível, no entanto, grande parte dos textos a serem cifrados possuem redundância, o que permite aplicar técnicas conhecidas de Criptanálise (como por exemplo análise de frequência) e poder recuperar P_1 e P_2 a partir de $P_1 \oplus P_2$ [3].

Note-se a importância de se evitar a reutilização de RC4(IV, SK). Agora percebemos o porquê da inserção do IV antes da chave secreta: *ter um IV diferente para cada pacote enviado* e dessa maneira evitar a reutilização de RC4(IV, SK). O problema é que esse objectivo não foi conseguido, vejamos porquê: a norma (original) IEEE 802.11 não especificava como deveriam ser gerados os IV's, embora recomendasse a sua mudança por cada pacote enviado. Ora, intuitivamente seria de esperar que uma geração aleatória dos IV's fosse uma boa solução, no entanto neste modo de operação, é muito provável obtermos dois IV's iguais - que chamamos de **colisão** - muito rapidamente! Isto deve-se ao chamado *paradoxo do aniversário*:

Suponhamos que o aniversário de uma pessoa possa ser, com igual probabilidade, em qualquer dos dias do ano. Se r ($r \leq 365$) pessoas forem escolhidas ao acaso, a probabilidade de que todas façam anos em dias diferentes é dada pela fórmula

$$P = \frac{n(n-1)(n-2)\ldots(n-r+1)}{n^r},$$

com $n = 365$, e portanto, a probabilidade de existirem pelo menos duas a fazerem anos no mesmo dia é dada por $1 - P$. É surpreendente que num grupo de $r = 40$ pessoas, a probabilidade de pelo menos duas delas terem nascido no mesmo dia do ano seja superior a 85%!

Transportando esta análise para o nosso estudo, pondo $n = 2^{24}$ conclui-se que ao fim de aproximadamente 4850 pacotes[6] teremos uma colisão, com probabilidade superior a 50%.

A melhor maneira de gerar os IV's parece ser a de ir incrementando de 1 por cada pacote enviado. Note-se no entanto que os IV's têm comprimento de 24 bits e portanto uma colisão é garantida após o envio de $2^{24} = 16777216$ pacotes. O IEEE 802.11b consegue transmitir cerca de 500 pacotes de 1500 bytes por segundo, ou seja, em aproximadamente 9 horas o espaço dos IV's estará esgotado e portanto vai haver uma colisão.

A partir do momento em que o atacante consegue o texto antes de estar cifrado P para uma determinada mensagem, imediatamente consegue obter o valor da chave pseudo-aleatória. Relembre-se que

$$C = P \oplus \mathrm{RC4}(IV, SK),$$

e portanto

$$RC4(IV, SK) = \underbrace{C}_{conhecido} \oplus \underbrace{P}_{conhecido}.$$

Com esta chave pseudo-aleatória, o atacante consegue decifrar qualquer mensagem que use este IV. Ao longo do tempo, o atacante pode construir uma tabela de chaves pseudo-aleatórias para cada IV. Considerando que cada pacote tem 1500 bytes, para o total de 2^{24} IV's necessita-se aproximadamente de 24 GB de espaço em disco, o que para um computador actual não constitui um impedimento. Note-se ainda que, para os cartões PCMCIA que utilizam um contador para gerar os IV's e que começam sempre a contar

[6] Fizemos as contas tendo obtido exactamente 4823 pacotes.

do 0 sempre que são reiniciados, este ataque torna-se ainda mais prático, pois a tabela acima referida seria construída para os primeiros milhares de IV's, uma vez que numa instalação com muitos clientes (do 802.11) vão existir colisões nos primeiros milhares de IV's.

1.2.2.2 Autenticação de Mensagens

O protocolo WEP utiliza um valor de verificação de integridade para se assegurar da veracidade dos pacotes que estão a ser transmitidos, que como vimos anteriormente é determinado por um CRC de 32 bits. Vamos no entanto ver que o CRC, do modo como é implementado pelo WEP, é insuficiente para impedir que um atacante modifique a mensagem - CRC's são concebidos para o efeito de detectarem erros aleatórios aquando da transmissão das mensagens; eles são vulneráveis a ataques maliciosos, vejamos como:

Uma das propriedades de qualquer CRC é a seguinte:

Propriedade 1.2.1 *A soma de verificação de integridade $c(M)$ é uma função linear da mensagem, isto é, tem-se a seguinte distribuição sobre a operação XOR:*

$$c(x \oplus y) = c(x) \oplus c(y), \text{ para quaisquer escolhas de } x \text{ e } y.$$

Uma consequência desta propriedade é que se torna possível fazer modificações a um texto cifrado C, obtendo um novo texto cifrado C', que ao ser decifrado se torna numa nova mensagem M' tal que $c(M')$ é válido:

Suponhamos que interceptamos uma mensagem cifrada C antes de chegar ao seu destino. Temos que C advém de uma mensagem M que desconhecemos e verifica a seguinte igualdade

$$C = \text{RC4}(IV, SK) \oplus \langle M, c(M)\rangle.$$

Seja Δ um valor arbitrário e $C' = C \oplus \langle \Delta, c(\Delta)\rangle$. Então,

$$\begin{aligned} C' &= C \oplus \langle \Delta, c(\Delta)\rangle \\ &= \text{RC4}(IV, SK) \oplus \langle M, c(M)\rangle \oplus \langle \Delta, c(\Delta)\rangle \\ &= \text{RC4}(IV, SK) \oplus \langle M \oplus \Delta, c(M) \oplus c(\Delta)\rangle \\ &= \text{RC4}(IV, SK) \oplus \langle M', c(M \oplus \Delta)\rangle \\ &= \text{RC4}(IV, SK) \oplus \langle M', c(M')\rangle. \end{aligned}$$

Podemos assim substituir a transmissão original onde iriam C e IV por C' e IV. O receptor irá obter a mensagem M' com o $c(M')$ válido. Desta maneira podemos fazer as modificações que quisermos sem sermos detectados, falhando portanto o WEP no objectivo de conseguir a integridade dos dados transmitidos.

Uma consequência do que acabamos de ver é que é possível decifrar mensagens emitidas via *wireless*. A ideia do ataque é a seguinte: quando capturamos uma mensagem cifrada, não conseguimos obter (instantaneamente) a mensagem original, a não ser que estivéssemos de posse da chave secreta. Mas há uma "entidade" que o consegue fazer - o ponto de acesso ("Access Point"). O objectivo é então fazer com que o ponto de acesso decifre mensagens e as envie para um destino que o atacante controle[7] - esta técnica é conhecida por re-direcção do IP ("IP redirection"). O ataque consiste assim em alterar o endereço de destino de um determinado pacote para um que o atacante controla. Assim, o ponto de acesso irá decifrar o pacote e enviá-lo (via Internet) para a sua nova "morada", onde o atacante poderá ler a mensagem.

Mostraremos agora que o WEP não fornece um controle de acesso seguro a uma rede sem fios. Como já foi observado em cima, a soma de verificação de integridade tem a seguinte propriedade:

Propriedade 1.2.2 *Para qualquer mensagem M, a função $c(M)$ não depende da chave secreta.*

Se o atacante conseguir obter um texto P que havia sido cifrado, ele irá conseguir injectar mensagens na rede, pois

$$P \oplus C = P \oplus (P \oplus RC4(IV, SK)) = RC4(IV, SK),$$

ou seja, o atacante descobre a chave pseudo-aleatória correspondente a um determinado IV, e assim pode injectar novos textos cifrados C' para quaisquer mensagens M' (em virtude da propriedade anterior, $c(M')$ pode ser calculado sem o conhecimento da chave secreta),

$$C' = RC4(IV, SK) \oplus \langle M', c(M') \rangle.$$

Note-se que esta (falsa) mensagem usa o mesmo IV que a original, no entanto os pontos de acesso que utilizam o WEP verificam a seguinte:

[7]Este ataque só funcionará no caso do ponto de acesso funcionar como um *router* de IP com ligação à Internet, que é um cenário bastante usual.

Propriedade 1.2.3 *Podem-se reutilizar IV's sem desencadear qualquer tipo de alarme ao receptor.*

Desta maneira, sempre que se tenha um IV e a sua correspondente chave pseudo-aleatória RC4(IV,SK), podemos reutilizá-la indefinidamente, ultrapassando assim o mecanismo de controlo de acesso do WEP[8].

Vamos ver como esta descrição pode ser usada para um atacante se autenticar, sem saber a chave secreta: quando um utilizador (por exemplo de um computador portátil) se quer conectar a um ponto de acesso, tem de provar a sua identidade. Como analogia, quando assinamos um cheque, estamo-nos a autenticar para com o receptor do mesmo, pois ele usará a assinatura para provar ao banco que nós realmente escrevemos nesse cheque. Numa rede LAN, todo o aparelho tem (em princípio) um único número chamado de morada MAC ("MAC address"). Qualquer transmissão numa rede LAN de um aparelho para outro contém a sua morada MAC, portanto a identidade de um emissor pode ser verificada. Mas como poderemos saber se alguém não forjou uma mensagem com uma morada MAC falsa? Uma tentativa de ultrapassar este problema é autenticar o aparelho quando ele se junta à rede LAN e "combinar" sobre um código secreto que será usado para autenticar todas as mensagens subsequentes. Como só o verdadeiro aparelho e o ponto de acesso sabem esse código, cada mensagem é validada como autêntica quando é recebida. Isto é o objectivo da autenticação.

Infelizmente, no IEEE 802.11 WEP, existe uma fase na qual o utilizador faz a autenticação do seu aparelho, não existindo autenticação das mensagens subsequentes, sendo assim impossível saber se estas vieram realmente do utilizador ou de um impostor.

De volta ao ataque, quando um utilizador (aparelho) se quer autenticar perante o ponto de acesso, este envia-lhe uma mensagem (não cifrada) de 128 bytes ("challenge message") que o utilizador tem de cifrar, usando o WEP, e reencaminhá-la para o ponto de acesso, que depois envia uma mensagem de sucesso ou de insucesso, conforme o utilizador tenha de facto cifrado com a chave secreta do WEP ou não, respectivamente. Soa lindamente, mas repare-se que o ponto de acesso ao enviar a mensagem de 128 bytes, acabou de "dar" ao atacante texto antes de estar cifrado P, e o utilizador ao

[8]Relembre-se que apesar de recomendar fortemente a não reutilização de IV's, a norma 802.11 não exigia que estes mudassem para cada pacote enviado e, portanto, um receptor teria de aceitar IV's repetidos ou então correr o risco de falhas na inter-comunicação.

reencaminhar essa mesma mensagem já cifrada acabou de "dar" o texto depois de cifrado ao atacante C. Portanto, o atacante calcula automaticamente os primeiros 128 bytes da chave pseudo-aleatória $RC4(IV, SK) = C \oplus P$, para um determinado IV. Como todos os processos de autenticação têm as mensagens com o mesmo tamanho (portanto o atacante não necessita de conhecer mais bits da sequência $RC4(IV, SK)$), a chave pseudo-aleatória calculada anteriormente servirá para um intruso se autenticar sempre que quiser.

1.2.3 O ataque de Fluhrer, Mantin e Shamir [FMS]

O ataque que descreveremos a seguir é, de facto, o mais brutal contra o protocolo de segurança WEP. Ele permite recuperar a chave secreta - é o *climax* da Criptanálise!

Este ataque pressupõe que se consiga obter o primeiro byte (sem estar cifrado) de cada pacote, de modo a que se consiga calcular o primeiro byte da chave pseudo-aleatória (mais à frente ficará clara esta necessidade), utilizando técnicas estudadas anteriormente. Mas como é dito em [16], normalmente o primeiro byte (sem estar cifrado) de cada pacote corresponde ao valor hexadecimal "AA", e portanto é possível determinar o primeiro byte da chave pseudo-aleatória para cada pacote.

Comecemos por introduzir alguma notação. Para indicar o estado do KSA (ver figura 1.2) depois de utilizadas as palavras $[0, \ldots, r]$ (isto é, depois de $r + 1$ passagens ("rounds")), utilizamos o índice r em S_r, i_r e j_r. Em particular, a permutação final do KSA denota-se por S_{N-1}.

Como foi visto anteriormente, o WEP "apresenta" ao RC4 uma chave K (digamos de comprimento l-bytes), que é a concatenação de um vector (que é conhecido) de 3 bytes (IV) com uma chave secreta (SK):

$$K = [IV[0]\ IV[1]\ IV[2]\ SK[0] \ldots SK[l-3-1]] = [K[0]\ K[1] \ldots K[l-1]].$$

O primeiro byte gerado pelo PRGA depende apenas de 3 dos bytes de S_{N-1}, que são: $X = S_{N-1}[1]$, $Y = S_{N-1}[S_{N-1}[1]]$, $Z = S_{N-1}[S_{N-1}[1] + S_{N-1}[S_{N-1}[1]]]$.

i	0	1	$\cdots$	X	$\cdots$	$X+Y$	$\cdots$	$N-1$
$S_{N-1}[i]$	$S_{N-1}[0]$	X	$\cdots$	Y	$\cdots$	Z	$\cdots$	$S_{N-1}[N-1]$

Suponhamos agora que o KSA atinge um estado em que

$$i \geq \max\{1, S_i[1], S_i[1] + S_i[S_i[1]]\}.$$

Então, o índice i já não atinge mais nenhuma vez estas posições. E quanto ao índice j? Considerando-o aleatório, a probabilidade de que ele não atinja nenhuma das três posições nas restantes $N-1-i$ passagens é igual a $\left((1-\frac{1}{N})^{N-1-i}\right)^3 > \left((1-\frac{1}{N})^{N}\right)^3 \approx 5\%$ (relembre-se que $N = 256$).

Definição 1.2.1 *Dizemos que o RC4 está num estado x-resolvido se, durante o KSA acontecer:*

1) $1,\ S_x[1] < x;$

2) $S_x[1] + S_x[S_x[1]] = x.$

Definição 1.2.2 *Uma chave diz-se x-boa se conduzir o RC4 a um estado x-resolvido.*

Suponhamos então que o RC4 está num estado *x-resolvido* e que conhecemos as primeiras x palavras (bytes) da chave do KSA (isto é, $K[0], \ldots, K[x-1]$). Então, podemos simular as primeiras x passagens do KSA, ficando assim a conhecer a permutação S_{x-1} e os índices i_{x-1} e j_{x-1}. Na passagem x temos que $i_x = x$ e $j_x = j_{x-1} + S_{x-1}[x] + K[x]$, ou seja,

$$K[x] = \underbrace{j_x}_{desconhecido} - \underbrace{j_{x-1}}_{conhecido} - \underbrace{S_{x-1}[x]}_{conhecido},$$

sendo assim necessário determinar j_x de modo a calcularmos o valor da chave secreta $K[x]$. Mas $S_x[x] = S_{x-1}[j_x]$ e portanto $j_x = S_{x-1}^{-1}[S_x[x]]$, onde S^{-1} denota a inversa da permutação S.

Mas como obter $S_x[x]$? Por hipótese, com probabilidade de pelo menos 5%, ter-se-á no final do KSA,

$$S_{N-1}[1] = S_x[1]$$
$$S_{N-1}[S_{N-1}[1]] = S_x[S_x[1]]$$
$$S_{N-1}[S_{N-1}[1] + S_{N-1}[S_{N-1}[1]]] = S_x[S_x[1] + S_x[S_x[1]]].$$

Determinemos o primeiro byte z gerado pelo PRGA:

$$\begin{aligned} i &= 1 \\ j &= S_x[1] \\ z &= S_x[S_x[1] + S_x[S_x[1]]], \text{pois } x \neq 1 \text{ e } x \neq S_x[1] \\ &= S_x[x]. \end{aligned}$$

Portanto,

$$K[x] = S_{x-1}^{-1}[z] - j_{x-1} - S_{x-1}[x].$$

Neste ponto é necessário estimar o número de chaves x-boas de modo a obter o byte correcto. Os autores em [5] afirmam que com 60 chaves[9] x-boas, a probabilidade de obter correctamente o byte x é maior que 50%. No apêndice A apresentamos algumas simulações computacionais, onde pretendemos dar a indicação da veracidade daquele valor.

Começando com o vector inicial (IV) e repetindo o processo descrito, podemos recuperar, um por um, os bytes secretos da chave. Temos agora que estimar quantos pares (IV,z), em que z designa o primeiro byte pseudo-aleatório correspondente ao IV, são necessários para recuperar completamente a parte secreta da chave. Existem essencialmente três maneiras distintas dos cartões *wireless* gerarem os IV's [16]:

1) Aleatoriamente;

2) Utilizando um contador;

3) Alternar entre dois IV's.[10]

Em [5], aparece a seguinte tabela,

[9]Este valor foi obtido por simulação computacional. Mais informação pode ser encontrada em http://www.derkeiler.com/Newsgroups/sci.crypt/2003-02/1597.html.

[10]Esta classe de cartões não é susceptível ao ataque que estamos a descrever, embora seja vulnerável aos ataques de Borisov *et al.* [2].

Comprimento IV	Probabilidade	Número de IVs necessários
3	$4,57 \times 10^{-5}$	1310000
4	$4,50 \times 10^{-5}$	1330000
5	$1,65 \times 10^{-4}$	364000
6	$1,64 \times 10^{-4}$	366000
7	$2,81 \times 10^{-4}$	213000
8	$2,80 \times 10^{-4}$	214000
9	$3,96 \times 10^{-4}$	152000
10	$3,94 \times 10^{-4}$	152000
11	$5,08 \times 10^{-4}$	118000
12	$5,04 \times 10^{-4}$	119000
13	$6,16 \times 10^{-4}$	97500
14	$6,12 \times 10^{-4}$	98100
15	$7,21 \times 10^{-4}$	83200
16	$7,18 \times 10^{-4}$	83600

que resume o estudo do ponto 1); a segunda coluna representa a fracção de chaves x-boas, $x \in \{3, \ldots, 16\}$. O número de pacotes necessários, ou seja, a terceira coluna, obtém-se fazendo o seguinte cálculo: $60 \cdot \frac{1}{probabilidade}$. Note-se que o número máximo de pacotes acontece para $x = 4$, e portanto esse número é suficiente para recuperar toda a chave secreta. Registe-se ainda que na versão "melhorada" do WEP, que utiliza IV's de 16 bytes, o ataque requer ainda um menor número de pacotes, cerca de 83600!!

Analisemos agora 2). Existem dois tipos de contadores utilizados nos cartões *wireless* para gerarem os IV's. São eles:

Pequeno Contador Final ("Little Endian Counter") Neste tipo de contador, o primeiro byte do IV é incrementado mais rapidamente.

Grande Contador Final ("Big Endian Counter") Este tipo de contador incrementa mais rapidamente o último byte do IV.

Para "atacar" este tipo de cartões, considerem-se IV's da forma (x, 255, V, ...), com $x \in \{l', \ldots, l-1\}$, onde l' representa o número de bytes do IV (relembre-se que l representa

o número de bytes da chave secreta K). Calculemos as duas primeiras passagens do KSA:

$$\begin{aligned} i &= 0 \\ j &= S[0] + K[0] \\ &= 0 + x \\ &= x \\ S[i] &\leftrightarrow S[j] \end{aligned}$$

i	0	1	2	$\cdots$	x	$\cdots$	255
$S[i]$	x	1	2	$\cdots$	0	$\cdots$	255

$$\begin{aligned} i &= 1 \\ j &= j_0 + S_0[1] + K[1] \\ &= x + 1 + 255 \\ &= x \\ S[i] &\leftrightarrow S[j] \end{aligned}$$

i	0	1	2	$\cdots$	x	$\cdots$	255
$S[i]$	x	0	2	$\cdots$	1	$\cdots$	255

Na próxima passagem, $x > 1, S_2[1]$ e portanto, desde que j não seja 0 nem 1 até à passagem x (se o for, esse IV não serve), o RC4 estará num estado x-resolvido, uma vez que $x > S_x[1]$ e $S_x[1] + S_x[S_x[1]] = 0 + S_x[0] = x$.

Temos assim que, se os IV's forem gerados por um *Pequeno Contador Final*, o atacante pode esperar por IV's da forma $(x, 255, V, \ldots)$, com $l' \leq x \leq l-1$ e V's diferentes. O atacante consegue assim um "bom" par (IV, z) para cada x, depois de percorrer todas as combinações possíveis dos dois primeiros bytes e consegue 60 "bons" pares esperando no máximo $2^8 \cdot 2^8 \cdot 60 \approx 4000000$ de pacotes.

Se os IV's forem gerados por um *Grande Contador Final*, então o número de pacotes para o ataque depende de l'. Para $l' = 3$ (que é o caso do WEP), pode-se esperar por todas as combinações dos dois últimos bytes do IV (2^{16}) e multiplicar pelo tamanho da

chave l. Isto dá (para $l = 16$, ou seja, uma chave de 128 bits) aproximadamente 1050000 de pacotes.

Pouco tempo depois de [5] ter sido publicado, Stubblefield, Ionnidis e Rubin [16] implementaram este ataque, tendo conseguido recuperar uma chave WEP de 128 bits, em poucas horas (tempo este que pode ser optimizado para alguns minutos, dependendo também do número de utilizadores de uma determinada rede local sem fios).

Capítulo 2

WPA, RSN e 802.11i

Em virtude do que foi visto nas secções anteriores, obviamente que novas e melhores medidas de segurança teriam de ser concebidas e implementadas nas redes sem fios. Nas próximas secções descreveremos os novos protocolos que substituem o WEP e pretendem providenciar verdadeira segurança.

O IEEE contém um subgrupo designado de *Associação de normas* ("Standards Association" - SA). De entre várias normas, o IEEE-SA é responsável pela "família" IEEE 802, que está dividida em grupos de trabalho, cada um dos quais produz normas numa área específica. O grupo ".11" produz normas para redes locais sem fios *WLAN* ("Wireless Local Area Network").

A norma original IEEE 802.11 foi ratificada em 1997, tornando-se uma norma internacional em 1999. O trabalho é contínuo e actualizações à norma são feitas com regularidade. Algumas destas, tais como 802.11a e 802.11b [4], estão completas, enquanto que outras se encontram em desenvolvimento. Note-se que actualizações tais como IEEE 802.11b não constituem novas normas, são apenas suplementos à norma existente.

As normas permitem aos fabricantes produzir produtos que têm características físicas conhecidas. Por exemplo, um computador portátil e um ponto de acesso não podem comunicar um com o outro a não ser que usem rádio frequências e métodos de modulação compatíveis. Uma norma especifica este tipo de situações em detalhe. São, portanto, muito úteis aos fabricantes, pois criam uma descrição técnica exacta através da qual os produtos podem ser concebidos. No entanto, do ponto de vista do utilizador final,

isto é, do indivíduo que compra o produto, existe uma preocupação diferente: o IEEE 802.11 pode informá-lo sobre as características do produto, mas não garante que um produto comprado ao vendedor A funcione completamente com um produto comprado ao vendedor B.

O IEEE 802.11 é uma longa e complicada norma; apesar dos melhores esforços das pessoas envolvidas na sua concepção, podem haver áreas que sejam ambíguas ou não totalmente definidas. Existem, também, um certo número de particularidades que são opcionais, consequentemente os fabricantes podem fazer escolhas diferentes para os seus produtos. Para evitar problemas de inter-funcionamento, a "Wi-Fi Alliance" foi formada, por um grupo de grandes fabricantes, e assim surgiu a sigla "Wi-Fi".

Para obter certificação, um fabricante tem de submeter o seu produto a testes contra um conjunto de "normas de ouro" de produtos Wi-Fi. A "Wi-Fi Alliance" criou os seus próprios testes, baseados no IEEE 802.11 - algumas características do IEEE 802.11 não são necessárias para uma certificação Wi-Fi, assim como existem outras que são adicionais à norma. Em suma, Wi-Fi define um subconjunto do IEEE 802.11 com algumas extensões.

2.1 O que são o IEEE 802.11i e o WPA?

O que foi adicionado à norma, tendo em vista a criação de uma nova geração de segurança é chamado de IEEE 802.11i.

O IEEE 802.11i define um novo tipo de rede sem fios, chamado de *rede robusta de segurança* ("Robust Security Network" - RSN). Para se juntar a um RSN, um aparelho *wireless* tem de ter um número de novas capacidades, que descreveremos nas próximas secções. Num *verdadeiro* RSN, o ponto de acesso só admite que se conectem aparelhos que disponham do RSN (RSN-"capable mobile devices"). No entanto, a maioria dos cartões Wi-Fi não podiam ser actualizados para o RSN, pois os protocolos criptográficos necessários não eram suportados pelo hardware e estavam para além das capacidades de actualização por software. Havia, portanto, a necessidade de criar uma "ponte" entre o que havia (WEP) e o RSN, uma vez que, por exemplo, os consumidores finais não estariam na disposição de simplesmente "deitar fora" o que tinham e mudar para a nova

solução de segurança.

Para contornar o problema, o grupo de trabalho "i" começou a desenvolver uma solução para a segurança, baseada nas capacidades dos produtos Wi-Fi existentes. Isto conduziu ao aparecimento do TKIP ("Temporal Key Integrity Protocol"), que descreveremos mais adiante. Enquanto a norma RSN não estava totalmente concluída, a Wi-Fi Alliance adoptou, do anteprojecto RSN, uma nova solução de segurança, chamada de "Wi-Fi Protected Access" (WPA), especificando apenas o TKIP[1]. Com isso, os produtos existentes puderam ser actualizados via *software*, assim como novos produtos já incluíam o WPA.

2.2 Contexto de segurança

O grupo IEEE 802.11i tinha dois objectivos para uma rede sem fios: criar uma nova solução de segurança que, em particular, oferecesse real protecção a todos os ataques conhecidos. Foi assumido que a nova solução iria substituir completamente o WEP, ao longo do tempo. A primeira e mais importante medida tomada foi a da separação do processo de autenticação de um utilizador da protecção de mensagens (integridade e privacidade). A autenticação é o processo pelo qual um utilizador prova que é elegível de se juntar a uma rede (e que a rede é legitima); a protecção de mensagens assegura que, quando um utilizador (e a rede) está autenticado, pode comunicar sem risco de intersecção, modificação ou qualquer outro tipo de risco para a segurança. A separação da autenticação da protecção de mensagens permite conceber uma solução que vai desde pequenos sistemas até grandes corporações. No entanto, as duas partes têm de ser inter-ligadas num *contexto de segurança*[2]. Muita da arquitectura do RSN[3] refere-se ao estabelecimento e manutenção de um contexto de segurança entre aparelhos sem fios LAN. A espinha dorsal deste contexto reside (como em todos os protocolos de segurança) na chave secreta.

[1] O RSN suporta a cifra AES, para além do TKIP.

[2] Este conceito refere-se a um processo de autenticação, seguido de um sistema de segurança (de tempo limitado) que confere direitos aos participantes.

[3] Usamos RSN aqui como no resto da secção, pois é o modelo global para a segurança. O WPA é derivado do RSN, portanto os mesmos comentários podem ser aplicados ao WPA.

2.2.1 Chaves

A segurança de um protocolo criptográfico assenta fortemente na(s) chave(s) secreta(s), ficando completamente perdida se a(s) chave(s) forem copiadas ou roubadas.

No RSN, o contexto de segurança é definido pela posse de chaves de duração limitada. Ao contrário do WEP, no RSN existem muitas chaves diferentes, fazendo parte de uma *hierarquia de chaves* ("Key hierarchy"), e a maioria destas chaves não são conhecidas antes do processo de autenticação estar concluído. De facto, a criação das chaves é feita em tempo real enquanto o contexto de segurança é estabelecido depois da autenticação. Estas podem ser actualizadas de tempos em tempos, mas são sempre destruídas quando o contexto de segurança é encerrado.

Durante a fase da autenticação, um utilizador tem de provar a sua identidade, demonstrando que está de posse de um segredo. Ultrapassando este teste, poderá receber as outras chaves - no RSN, depois de ultrapassada correctamente a fase da autenticação, recebem-se ou criam-se as chaves que são usadas na cifragem e protecção dos dados. Estas chaves denominam-se por *chaves temporais* ou *chaves de sessão*, uma vez que apenas são utilizadas enquanto um contexto de segurança estiver estabelecido, isto é, no momento em que uma comunicação for encerrada, simplesmente podem-se "deitar fora" estas chaves.

A autenticação é baseada numa informação secreta partilhada entre partes que não pode ser criada automaticamente. A base de todos os métodos de autenticação é a de que a entidade que vai ser autenticada possua à partida alguma informação especial, que é designada de *chave mestra* ("Master Key"). É fulcral utilizar a chave mestra de maneira a que nunca seja descoberta por alguém não desejado. Regra geral, a chave mestra nunca é usada directamente, sendo apenas usada para criar as chaves temporais (o WEP, claro, violou esta regra, utilizando a chave mestra tanto na autenticação como na cifragem).

Observação: O leitor interessado em saber como são criadas estas chaves, pode consultar o capítulo 10 do livro [4].

2.2.2 Camadas de segurança

No WEP, todas as medidas relacionadas com segurança estavam definidas na mesma norma. Algumas funções, como por exemplo a cifragem, são assuntos bastante locais e apenas relevantes ao *hardware* dos Wi-Fi LAN's que estão a comunicar. Outros assuntos, particularmente decisões a quem deve ser permitido acesso à rede, são de extrema importância e têm de ser consistentes sobre toda a rede.

Por estas razões, era necessário identificar e implementar camadas ("Layers") "administrativas" na solução de segurança. Três camadas estão claramente identificadas. De facto, estas camadas não são específicas das redes sem fios LAN, sendo aplicáveis a qualquer sistema de segurança LAN. Uma vantagem de escolher este modelo de camadas é que o RSN pode ser adaptado a arquitecturas de segurança existentes que foram desenvolvidas para outros propósitos. As três camadas de segurança são as seguintes:

- Camada da rede local sem fios ("Wireless LAN layer").
- Camada do controlo de acesso.
- Camada de autenticação.

A camada da rede local sem fios é responsável, entre outras coisas, pela cifragem e decifragem da informação sempre que um contexto de segurança é estabelecido.

O trabalho da camada de controlo de acesso é o de gerir o contexto de segurança. Tem de impedir a passagem de informação de e para um inimigo (aqui, um inimigo é definido como alguém que não tem um contexto de segurança estabelecido). Esta camada é volúvel, podendo-se imediatamente mudar o nosso estatuto de inimigo para amigo quando as autenticações ocorrem e o contexto de segurança é estabelecido. A camada de controlo de acesso comunica com a camada de autenticação para saber se pode abrir um contexto de segurança e participa na criação das chaves temporais.

Na camada da autenticação são feitas provas de identidade, que podem ser aceites ou rejeitadas. Com efeito, esta camada tem poder de veto sobre qualquer um que se queira juntar à rede LAN e delega poder à camada do controlo de acesso assim que aprova alguém para se juntar à rede LAN.

A camada da rede local sem fios encontra-se no dispositivo *wireless* do ponto de acesso, enquanto que no aparelho móvel do utilizador tipicamente é implementada nos cartões Wi-Fi e seu *software* associado.

A camada de controlo de acesso usualmente reside completamente no ponto de acesso. Embora em pequenos sistemas a camada de autenticação possa estar também no ponto de acesso, em sistemas maiores é usualmente implementada num servidor de autenticação, separado dos pontos de acesso. No caso do utilizador, as camadas de controlo de acesso e de autenticação são usualmente implementadas no sistema operativo. Note -se que é muito importante que o utilizador autentique também a rede para se assegurar que não se está a juntar a uma rede falsa, criada por um atacante.

Como foi visto, existem três camadas de segurança; isto representa um problema quando se concebem sistemas em que é necessária a cooperação de várias camadas. Esta foi, aliás, uma das razões pela qual se tentou definir no WEP todas as questões de segurança na camada da rede local sem fios. Na concepção do RSN, este problema foi ultrapassado pela referência a normas desenvolvidas fora do IEEE 802.11, especificamente para as camadas de autenticação e controlo de acesso. Nos poucos casos em que estas normas precisavam de ser modificadas, o grupo IEEE 802.11i contactou os outros grupos de modo a serem efectuadas.

Parecia haver um candidato perfeito para a camada de controlo de acesso: enquanto o trabalho progredia na elaboração da norma de segurança, outro grupo, o IEEE 802.1X, estava a terminar uma norma, concebida especificamente para lidar com o controlo de acesso (IEEE, 2001). Foi assim seleccionado o IEEE 802.1X como o mais apropriado e com quase aprovação total.

A escolha para a camada de autenticação foi mais problemática, isto porque, existiam muitos candidatos possíveis. No final, a decisão foi a de que o IEEE 802.11i não especificaria nenhuma camada de autenticação obrigatória, mas o RSN seria concebido de modo a poder incorporar quaisquer "bons" métodos existentes. A palavra "bons" sublinha o facto de que a norma impõe condições nas capacidades de segurança dos vários métodos. Por exemplo, todos os métodos têm de suportar autenticação mútua.

Observação: Neste trabalho, vamos remeter-nos ao estudo da camada da rede local

sem fios, estudando os protocolos de segurança usados para uma transmissão segura da informação. O leitor interessado em estudar os novos processos de autenticação e controlo de acesso, poderá consultar, por exemplo, os capítulos 8 e 9 de [4].

2.3 TKIP

O TKIP ("Temporal Key Integrity Protocol") foi concebido essencialmente para colmatar as deficiências de segurança do WEP, sendo portanto compatível com as implementações usando o RC4. O TKIP introduz uma série de medidas com vista a eliminar cada uma das fragilidades "diagnosticadas" ao WEP. Vamos, nas próximas secções, enunciá-las e descrever o que foi feito para as (tentar) eliminar.

2.3.1 Integridade da Mensagem

A integridade de uma mensagem é uma parte essencial de qualquer protocolo de segurança, uma vez que, se um atacante conseguir modificar uma mensagem, ele pode comprometer um sistema de comunicação de várias maneiras. O WEP tinha um método para detectar modificações, a soma de verificação de integridade, que como vimos, não oferecia uma grande protecção. Relembre-se que esta soma não dependia da chave secreta, mas no novo método depende: a ideia é combinar todos os bytes da mensagem e uma chave secreta de modo a produzir um valor de verificação de integridade, designado MIC (ver secção 2.3.3.1). Assim um atacante não pode recalcular este valor a não ser que tenha conhecimento da chave secreta.

Existem vários métodos considerados seguros para calcular o MIC, mas para o TKIP existia um problema: todos esses métodos requeriam a introdução de um novo algoritmo criptográfico ou cálculos envolvendo multiplicações rápidas. O último caso ainda foi ponderado, mas geralmente era necessária pelo menos uma multiplicação para cada grupo de quatro bytes, o que implicava, para um pacote típico de 1514 bytes, pelo menos 379 multiplicações. No entanto, o microprocessador da maioria dos cartões Wi-Fi não é muito "poderoso"; tipicamente não tem *hardware* que execute multiplicações rapidamente, o que reduziria drasticamente a rapidez do fluxo da informação. Uma

proposta para ultrapassar este problema foi a de fazer o cálculo do MIC utilizando *software* apropriado, uma vez que se estivéssemos a usar um computador portátil com um processador recente, estes cálculos não envolveriam praticamente tempo nenhum. O problema eram os pontos de acesso, dos quais a maioria não teria esta mesma capacidade de processamento.

A solução teria assim de passar por um método tão seguro quanto os conhecidos, mas que ao mesmo tempo não dependesse de operações de multiplicação ou de um novo algoritmo criptográfico. Isto não é, de forma alguma, fácil de concretizar! No entanto, uma boa solução foi proposta pelo criptógrafo Niels Ferguson, utilizando um método que ele chamou de Michael, o qual não usa operações de multiplicação. O Michael pode ser implementado nos pontos de acesso existentes, sem arruinar a sua velocidade de processamento de informação. No entanto, há um preço a pagar, pois o Michael é vulnerável a ataques de força bruta. Assim, de modo a evitá-los, o Michael introduz um novo conceito, designado por *contra-medidas* ("countermeasures"), de que daremos uma breve descrição em 2.3.3.1.

Iremos estudar com mais detalhe o Michael (secção 2.3.3.1); salientamos para já que o método calcula um valor (MIC) que é adicionado à mensagem antes da cifragem, e que é depois verificado pelo receptor depois da decifragem. Este valor providencia integridade da mensagem, que como vimos não era efectiva no WEP.

O Michael opera em MSDUs (veja-se nota em baixo); o cálculo para obter o MIC é feito num MSDU em vez de em cada MPDU.

Nota 2.3.1 (MSDUs *versus* MPDUs) *Quer na norma, quer neste trabalho, aparecem referências a MSDUs ("MAC Service Data Unit") e MPDUs ("MAC Protocol Data Unit"). Ambos se referem a um pacote de dados, com uma morada de partida e de destino (e potencialmente outras coisas). O MSDU é o pacote que vai do software do computador do utilizador para a camada da rede local sem fios, saindo daqui os MPDUs para a antena. Nas transmissões, os MSDUs são enviados pelo sistema operativo para a camada da rede local sem fios e são convertidos em MPDUs de modo a serem enviados via ondas de rádio. Nas recepções, os MPDU's chegam através da antena e são convertidos em MSDUs antes de serem "entregues" ao sistema operativo.*

Existe um ponto muito importante a mencionar: um MSDU pode ser dividido em várias partes para produzir vários MPDUs, num processo chamado de fragmentação.

Estes MPDUs são reunidos novamente num único MSDU no outro lado da comunicação. Isto é feito de modo a que, se a transmissão for perdida devido a algum factor, só se precisará reenviar o MPDU, em vez de todo o MSDU.

Uma das vantagens é que não é necessário adicionar um valor MIC a cada fragmento (MPDU) da mensagem. Em contraste, a cifragem no TKIP faz-se em MPDU's.

Para o Michael é necessária uma chave secreta própria, que tem de ser diferente da chave usada para a cifragem. A criação de uma chave nestas condições é facilmente conseguida quando se geram chaves temporais a partir de uma chave mestra.

2.3.2 Selecção do IV e seu uso

Na secção 1.2 vimos o propósito do IV no WEP, analisando também as suas fragilidades no contexto de segurança, que passam essencialmente pelas seguintes:

- O comprimento do IV é muito pequeno, permitindo a reutilização de IV's.
- O IV não é específico para um determinado utilizador, portanto o mesmo IV pode ser usado com a mesma chave secreta em múltiplos aparelhos *wireless*.
- O modo como o IV é junto à chave secreta torna-o susceptível ao ataque FMS (secção 1.2.3).

O TKIP introduz uma série de novas regras para usar o IV. Essencialmente, existem três diferenças em como os IV's são usados comparativamente ao WEP:

- O comprimento do IV é aumentado de 24 para 48 bits.
- O IV tem um papel secundário como um contador sequencial, de modo a evitar ataques de repetição de mensagens (*Replay Attacks*)[4].
- Os IV's são construídos de modo a evitar certas "chaves fracas" (do ponto de vista do atacante corresponde a uma *chave boa*, como introduzido em 1.2.3).

[4]Para a definição deste ataque (assim como de outros), ver o capítulo 4 de [4].

2.3.2.1 Comprimento do IV

O IV no WEP tinha um comprimento de 24 bits, o que implicava que ao fim de 16777216 pacotes haveria de certeza uma colisão. Isto é inaceitável, pois na prática este número de pacotes pode ser enviado em poucas horas. Discutimos também o ataque FMS, mostrando como o WEP lhe é vulnerável - essencialmente porque os IV's aparecem primeiramente nos pacotes enviados (ver figura 1.1), dando assim informação ao atacante de quando se está diante de chaves fracas. Alguns vendedores tentaram reduzir a potencialidade deste ataque evitando alguns valores dos IV's que produzissem chaves fracas. No entanto, esta estratégia reduz ainda mais o número total possível de IV's, melhorando um problema mas piorando o outro!

Aquando da concepção do TKIP, os especialistas recomendaram que o comprimento do IV fosse aumentado de modo a criar uma solução robusta. Houve alguma discussão em como o fazer, mas no fim, o grupo i de trabalho do IEEE 802.11 decidiu inserir 32 bits extra. Foi uma decisão um tanto ou quanto controversa, pois nem todos os vendedores podiam actualizar os seus antigos sistemas de modo a satisfazerem este requisito (embora a maioria pudesse).

Dissemos atrás que o IV teria passado de 24 para 48 bits, mas adicionando 32 a 24 bits ficamos com 56! O que acontece na prática é que só 48 bits são usados, sendo 8 deles (1 byte) "atirados fora" de modo a evitar chaves fracas. Com IV's deste comprimento elimina-se o primeiro problema visto em 2.3.2:

> Suponhamos que temos um aparelho a enviar 10000 pacotes por segundo, o que é perfeitamente possível. O espaço dos IV's de 24 bits ficaria esgotado em menos de meia hora, enquanto que o dos IV's de 48 bits em aproximadamente 900 anos.

A maneira de incorporar este (mais longo) IV na chave para o RC4 vai ser estudada na secção 2.3.3.2 (veja-se também a figura 2.2).

2.3.2.2 O IV como um contador sequencial - o TSC

O WEP não tinha qualquer protecção contra ataques de repetição de mensagens! Um inimigo podia copiar um pacote válido e enviá-lo outra vez mais tarde, na expectativa de que, se decifrado correctamente da primeira vez, provavelmente o seria novamente. Num ataque de repetição, o inimigo não tenta decifrar as mensagens mas, por exemplo, gravando mensagens em que o utilizador apaga ("delete") um determinado ficheiro e reenviando-as mais tarde, pode provocar que um ficheiro com o mesmo nome seja apagado, sem ter "quebrado" o processo de cifragem.

O TKIP tem um mecanismo para controlar este tipo de ataque, designado por TKIP *contador sequencial*, abreviadamente TSC ("TKIP sequence counter").

Na realidade, o TSC e o IV são a mesma coisa, ou seja, representam o mesmo valor, que começa sempre em 0 e é incrementado de 1 por cada pacote enviado. Como temos a garantia do IV nunca ser repetido para uma dada chave, podemos prevenir o ataque de repetição, ignorando quaisquer mensagens com um TSC que já tenha sido recebido. Estas regras significam que não é possível implementar um ataque de repetição gravando mensagens anteriores e enviá-las novamente mais tarde.

A maneira mais simples de prevenção a ataques de repetição seria a de "deitar fora" todas as mensagens recebidas nas quais o TSC não viesse incrementado de 1 relativamente à última mensagem. No entanto, existem várias razões pelas quais esta abordagem não funciona na prática. Primeiro, é possível que pacotes sejam perdidos numa transmissão devido a interferências e ruído, e portanto, se um pacote com TSC = 1234 fosse recebido e o próximo com TSC = 1235 não, o próximo pacote teria TSC = 1236, um valor que está incrementado de dois em relação ao último válido. Devido ao pacote perdido, todos os pacotes subsequentes seriam rejeitados, pois o TSC não tinha sido incrementado de 1.

Tendo em vista o que dissemos e ainda outro tipo de problemas, o TKIP faz uso do conceito de *janela de repetição* ("Replay Window"). O receptor guarda o maior valor de TSC recebido, assim como os últimos 16 valores do TSC recebidos. Quando um novo pacote é recebido, é categorizado num dos seguintes três tipos:

- Aceite: O TSC é maior que o mais alto dos recebidos.

- Rejeitado: O TSC é menor que o maior valor menos 16.
- Janela: O TSC é menor que o maior valor, mas maior que o menor valor (maior menos 16).

Para a categoria Janela, é verificado se um pacote com esse TSC já foi recebido anteriormente. Se foi, é rejeitado.

Este conjunto de regras é mais complicado do que aquele que descrevemos primeiramente, mas efectivamente previne os ataques de repetição, permitindo ao mesmo tempo a eficiência do protocolo.

2.3.2.3 Prevenção ao ataque FMS

Na secção 1.2.3 discutimos o ataque mais devastador ao WEP, que permite recuperar a chave secreta. Ron Rivest, que concebeu o RC4, recomendou uma solução bastante simples para este problema [14]: descartar os primeiros 256 bytes da chave pseudo-aleatória, que se manifestou ser impossível de concretizar no *hardware* existente, pois a sua configuração era a de utilizar a chave pseudo-aleatória a partir do primeiro byte gerado.

Assim, para prevenir este ataque, teriam de ser definidos outros métodos de defesa. O grupo que concebeu o TKIP focou-se em dois pontos:

- Tentar evitar chaves fracas.
- Alterar a chave secreta para cada pacote.

Em relação ao primeiro ponto, existe um problema em o concretizar: um criptógrafo pode afirmar que uma determinada chave é fraca, mas não pode afirmar que todas as outras são *fortes*. Assim, o que se fez foi fixar dois bits do IV (ver secção 2.3.3.2), evitando deste modo uma classe de chaves fracas conhecidas. Relembre-se que o IV foi estendido para 48 bits em vez dos antigos 24; assim, mesmo fixando dois bits, não existe o problema de um pequeno espaço de IV's.

Quanto ao segundo ponto[5], a ideia é impedir que o atacante consiga obter as amostras necessárias (relembre-se que com cerca de 60 chaves x-boas, um atacante poderia

[5]No WEP, a chave de cifragem mudava com cada pacote enviado porque continha o IV. No entanto, a parte secreta da chave (excluindo o IV) era constante.

começar a descobrir a chave secreta) para atacar qualquer chave (secção 2.3.3.2).

2.3.3 Implementação do algoritmo TKIP

Vamos agora "mergulhar" nos detalhes do algoritmo TKIP.

Suponhamos que as chaves mestras foram distribuídas e as chaves da sessão derivadas em ambas as partes que querem comunicar. A tarefa do TKIP é a de providenciar a validação da integridade dos dados recebidos, assim como "esconder" os dados que são transmitidos. Para isso, o TKIP implementa o seguinte:

- Geração e verificação de IV's;
- Geração e verificação de MIC's;
- Cifragem e decifragem.

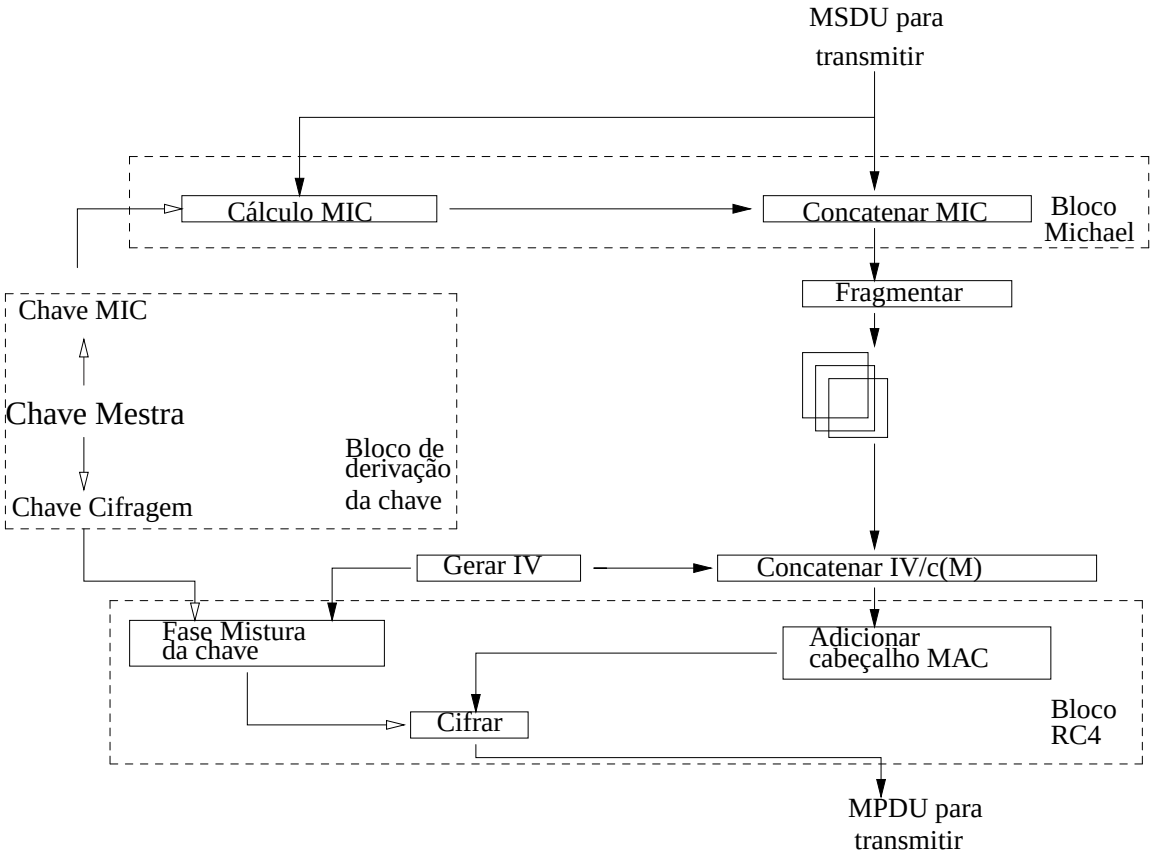
Do ponto de vista do transmissor, a articulação destas componentes com outras actividades do processo pode ser vista na figura seguinte.

Note-se que o MIC é calculado sobre, e concatenado (no último byte) ao MSDU, antes da fragmentação. Como resultado, o valor do MIC aparece apenas no último MPDU. A soma de verificação original do WEP ($c(M)$) ainda é calculada e concatenada a cada MPDU.

Do ponto de vista do receptor (veja-se a figura em baixo), o processo não é uma completa inversão do anterior (a decifragem não é a primeira operação a ser feita), pois em primeiro lugar faz-se uma verificação do TSC no intuito de prevenção aos ataques de repetição de mensagens. Há também a verificação do valor de $c(M)$ que também pode ser usado para rejeitar um pacote[6].

O MIC é verificado depois de todos os fragmentos terem sido recebidos e juntos formando o MSDU. Note-se que se o valor do MIC não for o correcto, o MSDU irá ser rejeitado e serão invocadas contra-medidas. Embora possível, é extremamente improvável que os fragmentos passem o teste da $c(M)$, tendo havido erros aleatórios durante a

[6]Tecnicamente, isto não corresponde a uma verificação de integridade, mas fornece uma rápida indicação do sucesso ou não da decifragem: decifrar um pacote com um valor errado de IV ou da chave conduz quase sempre a um valor errado de $c(M)$.



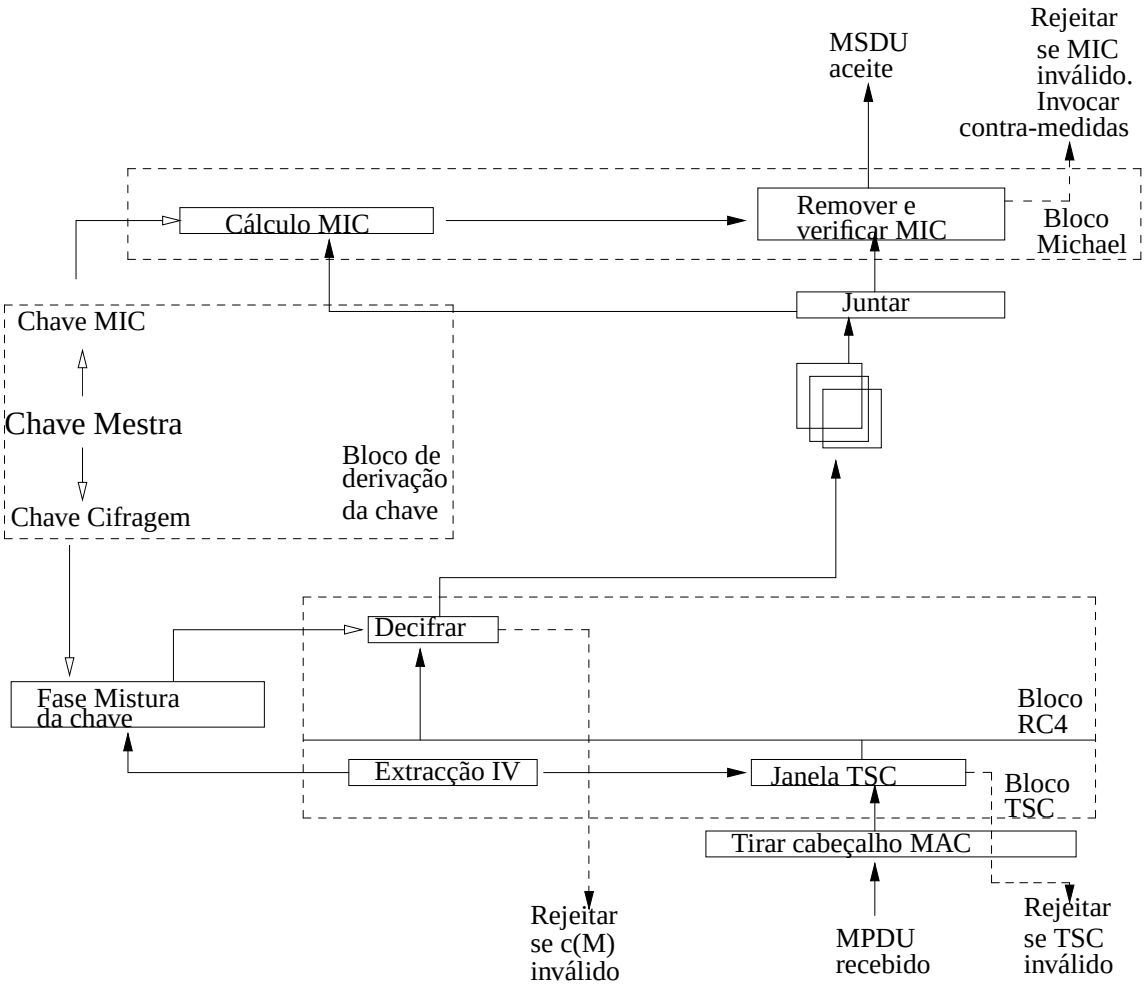
transmissão. Portanto, se o valor do MIC não estiver correcto, podemos "ter a certeza" de que se deveu a um ataque malicioso.

2.3.3.1 Integridade da mensagem-Michael

O Michael foi inventado por Neils Ferguson (2002) e foi concebido especificamente para colmatar algumas necessidades do TKIP, particularmente a necessidade de ser implementado usando um processador de baixo poder ("low power processor"), assim como o não recurso a multiplicações rápidas em *hardware*.

A "adopção" do Michael pela norma foi de alguma maneira controversa, isto porque, o algoritmo era novo e "novo" é uma má palavra para a comunidade criptográfica. Os criptógrafos gostam de algoritmos bem estudados e, para além disso, o nível de segurança medido pelo tamanho da chave (veja-se por exemplo página 61 e seguintes de [1] para uma descrição e exemplo deste conceito) é baixo - apenas 20 bits[7]. Em [4], a

[7]Note-se que existem outros métodos bastante seguros para calcular o MIC; o problema é que estes métodos não podiam ser implementados no equipamento existente.



consequência disto é, citando:

> Um valor do MIC escolhido aleatoriamente tem a probabilidade de $1/2^{20}$ de ser aceite como válido. Este valor (aproximadamente um num milhão) não se considera ser muito raro em normas criptográficas.

Posto isto, foram então elaboradas contra-medidas que, em poucas palavras, passam por desactivar as chaves existentes numa comunicação assim que um ataque é detectado, gerando em pouco tempo novas chaves. Os aparelhos que estavam em comunicação são assim impedidos de o fazer enquanto as novas chaves não são geradas[8].

Estas contra-medidas são efectivas para a prevenção de ataques em que há alterações

[8]Tipicamente, estas novas chaves podem ser geradas imediatamente, no entanto, o Michael usa a regra de que, se tiver ocorrido um ataque ao MIC nos últimos 60 segundos, as novas chaves só são criadas depois de expirar um período de 60 segundos. Isto limita o atacante a uma tentativa por minuto na rede inteira.

nas mensagens, mas conduz a um outro tipo de ataque, designado por *ataque de negação de serviço* ("denial-of-service attack[9]").

Vamos então descrever o algoritmo: o Michael calcula um valor (de verificação) de 8 bytes, chamado de *código de integridade da mensagem* ("message integrity code", abreviado por MIC). Este cálculo envolve apenas substituições, rotações e a operação XOR (não há multiplicações). A informação a ser protegida pelo MIC inclui a mensagem do utilizador e as moradas do "remetente" (MR) e do destinatário (MD) - ver tabela em baixo.

MD	MR	Mensagem do utilizador

A primeira parte do algoritmo consta em organizar os dados em palavras de 32 bits. Isto é feito tanto para a chave como para os dados em si. A chave de 64 bits é dividida em duas palavras de 32 bits, denominadas por K_0 e K_1. Esta divisão é exemplificada na figura seguinte, onde a chave de 64 bits é "tratada" como uma chave de 8 bytes, armazenada sequencialmente na memória.

Chave

0x1122334455667788

k7=11
k6=22
k5=33
k4=44
k3=55
k2=66
k1=77
k0=88

K_0=0x55667788; K_1=0x11223344.

Figura 2.1: Divisão da chave de 64 bits em duas de 32 bits.

[9] Há quem considere que este tipo de ataque é um "facto da vida" para as redes sem fios, outros não. O leitor mais interessado, pode consultar [4], págs 334-336.

Depois desta divisão, os dados a serem enviados têm também de ser divididos em palavras de 32 bits, portanto o seu comprimento tem de ser um múltiplo de 4 bytes. Caso seja necessário, adicionam-se alguns bytes para esse efeito. Note-se que estes bytes extra apenas servem para calcular o MIC, não sendo portanto enviados com a mensagem.

Suponhamos que temos então duas palavras chave K_0 e K_1 e um conjunto de palavras (os dados) $M_0, M_1, \ldots, M_{n-1}$.

O nosso objectivo é calcular o valor MIC de 64 bits, compreendendo duas palavras de 32 bits, V_0 e V_1, que serão adicionadas aos dados antes da cifragem.

O algoritmo é o seguinte:

1) Faça-se uma cópia da chave: $l = K_0$ e $r = K_1$.

2) Faça-se XOR da primeira palavra dos dados M_0 com l.

3) Aplique-se a função Michael (descrita um pouco mais à frente) aos valores l e r, obtendo assim dois novos valores.

4) Repita-se os passos 2 e 3 para os restantes blocos dos dados.

Os valores finais de l e r formam o MIC, isto é, obtém-se duas palavras V_0 e V_1 de 32 bits, respectivamente. Em linguagem de programação, esta sequência pode ser representada da seguinte maneira:

$(l, r) \leftarrow (K_0, K_1)$
for $i = 0$ to $n - 1$ do
  $l \leftarrow l \oplus M_i$
  $(l, r) \leftarrow$FnMichael(l, r)
$V_0 = l$
$V_1 = r$.

A função Michael, denotada por FnMichael, tem como entrada duas palavras (l, r), que são utilizadas no seguinte algoritmo (em linguagem de programação), para produzir duas novas palavras (l, r):

Entrada: (l, r)
$r \leftarrow r \oplus (l <<< 17)$
$l \leftarrow (l + r) \bmod 2^{32}$

$r \leftarrow r \oplus \text{XSWAP}(l)$

$l \leftarrow (l + r) \bmod 2^{32}$

$r \leftarrow r \oplus (l <<< 3)$

$l \leftarrow (l + r) \bmod 2^{32}$

$r \leftarrow r \oplus (l >>> 2)$

$l \leftarrow (l + r) \bmod 2^{32}$

Saída: (l, r),

onde $<<<$ ($>>>$) significa rotação para a esquerda (direita), respectivamente, e XSWAP significa permutar os primeiros 16 bits com os últimos 16 bits de uma palavra de 32 bits (ex: em hexadecimal, 12345678 transforma-se em 56781234).

2.3.3.2 Mistura da chave por-pacote (*Per-Packet Key Mixing*)

A função *mistura da chave* cria uma nova chave para cada pacote transmitido. Foi introduzida no protocolo por duas razões:

- Proteger o RC4 de *chaves fracas*.
- Incorporar os bits extra do IV estendido.

A abordagem feita foi a de combinar a chave da sessão, o IV e a morada MAC da fonte com uma função de *compressão* ("hash function") de modo a obter uma *chave de mistura*. Incluir a morada MAC da fonte proporciona maior protecção uma vez que, quando duas partes comunicam entre si, têm uma determinada chave de sessão e supondo que ambos começavam com o valor de IV zero incrementando-o de um para cada pacote enviado, imediatamente ocorreriam colisões. No entanto, para duas partes comunicarem numa rede LAN, as suas moradas MAC têm de ser distintas, ou seja, mesmo na eventualidade de duas partes comunicarem utilizando a mesma chave de sessão e o mesmo IV, a chave de mistura vai ser diferente para cada parte.

Saliente-se que implementar uma operação de *compressão* ("hash") para cada pacote (a ser cifrado/decifrado) não é uma tarefa fácil para um processador de baixo poder. Assim, a *mistura* divide-se em duas fases. Na fase 1, a chave da sessão e a morada da fonte são combinadas através da função de *compressão*, mantendo-se o resultado

inalterado durante a sessão. Na fase 2 (executada para cada pacote), o IV é misturado com o resultado da primeira fase, de modo a produzirem uma chave de cifragem. Esta chave será depois usada para inicializar o RC4.

Vamos então descrever o algoritmo (veja-se também a figura 2.2) . Primeiro, algumas abreviaturas:

TSC = Contador sequencial do TKIP (48 bits)
TSCU = Últimos 32 bits do TSC (32 bits)
TSCL = Primeiros 16 bits do TSC (16 bits)
TA = Morada MAC do transmissor (48 bits)
TK = Chave temporal da sessão (128 bits)
P1K = O resultado da primeira fase (80 bits)
P2K = O resultado da segunda fase (128 bits); isto vem a ser a chave para o RC4.

As duas fases podem ser descritas através das seguintes funções:

P1K←Fase1(TA,TSCU,TK)
P2K←Fase2(P1K,TSCL,TK).

Quando o sistema é iniciado ou uma nova troca de chave ocorre, o TSC toma o valor 0. O sistema tipicamente calcula o valor de P1K e armazena-o de modo a utilizá-lo para gerar o P2K. O P1K tem de ser recalculado sempre que o TSCU muda, ou seja, por cada $2^{16} = 65536$ pacotes.

Tanto a fase 1 como a fase 2 necessitam de uma lista-byte de substituição, que chamamos de S-box[10]. Esta lista tem 512 palavras que estão distribuídas em duas (sub)listas com 256 palavras cada. Para fazer a substituição de uma palavra de 16 bits X, usamos o segundo byte de X como índice para a primeira lista e o primeiro byte de X como índice para a segunda. Depois faz-se XOR das duas palavras saídas das tabelas para produzir a palavra final de 16 bits. Esta operação é denotada no algoritmo pela função

$$i = S[j].$$

Os 512 valores para a S-box encontram-se listados na norma [8].

[10]É uma função bijectiva não linear.

A saída da fase 1 tem apenas 80 bits de comprimento, mas usa todos os 128 bits de TK no seu cálculo. O resultado é obtido através de um conjunto de cinco palavras de 16 bits, designadas de $P1K_0$, $P1K_1$, $P1K_2$, $P1K_3$ e $P1K_4$. A seguinte terminologia é usada no algoritmo:

TSC_1 = bits 16-31 do TSC (os 16 bits do meio)

TSC_2 = bits 32-47 do TSC (os 16 últimos bits)

TA_n = n-ésimo byte de TA (TA_0 = primeiro byte e TA_5 = último byte)

TK_n = n-ésimo byte da chave temporal TK (TK_0 = primeiro byte e TK_5 = último byte)

A expressão $X \cap Y$ denota a combinação de dois bytes numa palavra de 16 bits de maneira que: $X \cap Y = 256 * X + Y$

O símbolo + significa adição módulo 2^{16}

S[ ] denota o resultado da tabela de substituição de 16 bits.

FASE1_ PASSO1:

$P1K_0 = TSC_1$

$P1K_1 = TSC_1$

$P1K_2 = TA_1 \cap TA_0$

$P1K_3 = TA_3 \cap TA_2$

$P1K_4 = TA_5 \cap TA_4$.

FASE1_ PASSO2:

for i = 0 to 3

Begin

$P1K_0 = P1K_0 + S[P1K_4 \oplus (TK_1 \cap TK_0)]$

$P1K_1 = P1K_1 + S[P1K_0 \oplus (TK_5 \cap TK_4)]$

$P1K_2 = P1K_2 + S[P1K_1 \oplus (TK_9 \cap TK_8)]$

$P1K_3 = P1K_3 + S[P1K_2 \oplus (TK_{13} \cap TK_{12})]$

$P1K_4 = P1K_4 + S[P1K_3 \oplus (TK_1 \cap TK_0)] + i$

$P1K_0 = P1K_0 + S[P1K_4 \oplus (TK_3 \cap TK_2)]$

$P1K_1 = P1K_1 + S[P1K_0 \oplus (TK_7 \cap TK_6)]$

$P1K_2 = P1K_2 + S[P1K_1 \oplus (TK_{11} \cap TK_{10})]$

$P1K_3 = P1K_3 + S[P1K_2 \oplus (TK_{15} \cap TK_{14})]$

$P1K_4 = P1K_4 + S[P1K_3 \oplus (TK_3 \cap TK_2)] + 2*i + 1$

End

Quanto à saída da fase 2, o resultado é obtido através de um conjunto de seis palavras de 16 bits, denominadas por PPK_0, PPK_1, PPK_2, PPK_3, PPK_4 e PPK_5. Além da terminologia definida anteriormente, é necessária a seguinte:

$P1K_n$ = saídas da fase 1

TSC_0 = bits 0-15 do TSC (os primeiros 16 bits)

A expressão >>> (palavra) significa que a palavra de 16 bits sofre uma rotação de um bit para a direita

A expressão $\rightarrow$ (palavra) significa que a palavra de 16 bits sofre uma translação de um bit para a direita

& e | representam "e" e "ou" lógicos (bit a bit), respectivamente

$RC4Key_n$ designa o n-ésimo byte da chave usada para cifrar através do RC4.

FASE2_ PASSO1:

$PPK_0 = P1K_0$

$PPK_1 = P1K_1$

$PPK_2 = P1K_2$

$PPK_3 = P1K_3$

$PPK_4 = P1K_4$

$PPK_5 = P1K_4 + TSC_0$.

FASE2_ PASSO2:

$PPK_0 = PPK_0 + S[PPK_5 \oplus (TK_1 \cap TK_0)]$

$PPK_1 = PPK_1 + S[PPK_0 \oplus (TK_3 \cap TK_2)]$

$PPK_2 = PPK_2 + S[PPK_1 \oplus (TK_5 \cap TK_4)]$

$PPK_3 = PPK_3 + S[PPK_2 \oplus (TK_7 \cap TK_6)]$

$PPK_4 = PPK_4 + S[PPK_3 \oplus (TK_9 \cap TK_8)]$

$PPK_5 = PPK_5 + S[PPK_4 \oplus (TK_{11} \cap TK_{10})]$

$PPK_0 = PPK_0 + >>>(PPK_5 \oplus (TK_{13} \cap TK_{12}))$

$PPK_1 = PPK_1 + >>>(PPK_0 \oplus (TK_{15} \cap TK_{14}))$

$PPK_2 = PPK_2 + >>>(PPK_1)$

$PPK_3 = PPK_3 + >>>(PPK_2)$

$PPK_4 = PPK_4 + >>>(PPK_3)$

$PPK_5 = PPK_5 + >>>(PPK_4)$

FASE2_ PASSO3:

$RC4Key_0$ = Último byte (TSC_0)

$RC4Key_1$ = (Último byte (TSC_0) | 0x20) & 0x7F

$RC4Key_2$ = Primeiro byte (TSC_0)

$RC4Key_3$ = Primeiro byte ($PPK_5 \oplus \rightarrow (TK_1 \cap TK_0)$)

$RC4Key_4$ = Primeiro byte (PPK_0)

$RC4Key_5$ = Último byte (PPK_0)

$RC4Key_6$ = Primeiro byte (PPK_1)

$RC4Key_7$ = Último byte (PPK_1)

$RC4Key_8$ = Primeiro byte (PPK_2)

$RC4Key_9$ = Último byte (PPK_2)

$RC4Key_{10}$ = Primeiro byte (PPK_3)

$RC4Key_{11}$ = Último byte (PPK_3)

$RC4Key_{12}$ = Primeiro byte (PPK_4)

$RC4Key_{13}$ = Último byte (PPK_4)

$RC4Key_{14}$ = Primeiro byte (PPK_5)

$RC4Key_{15}$ = Último byte (PPK_5).

A saída final da fase 2 consiste de uma lista de 16 bytes (128 bits) que contém a chave para ser usada no RC4. Os primeiros 3 bytes desta chave são transmitidos como sendo o IV do WEP (24 bits). Note-se que o segundo byte do IV é uma repetição do primeiro, excepto que o bit 5 é igual a 1 e o bit 4 é igual a 0. Desta maneira, previne-se a geração de uma grande classe de chaves fracas ([8]).

Nesta secção pretendeu-se descrever a maneira de como, a partir das limitações (a nível de segurança) dos aparelhos que utilizavam o WEP, se conseguiu desenvolver um novo protocolo (de segurança) de modo a poder ser implementado nos produtos existentes. Todas as "fraquezas" (conhecidas) do WEP foram tidas em conta na concepção

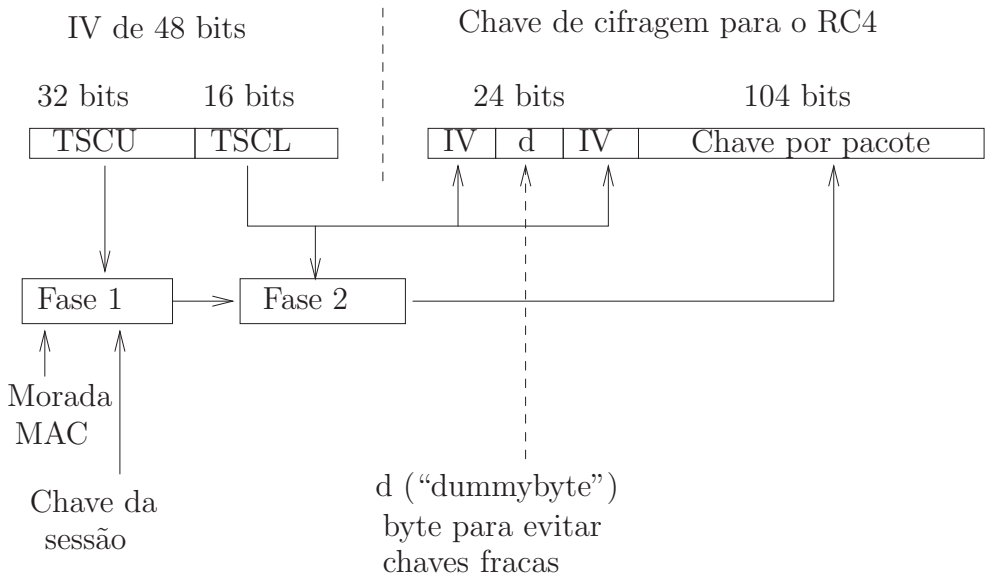


Figura 2.2: Criação da chave de cifragem para o RC4

desta nova solução: o TKIP. Finalizamos esta secção com uma frase retirada de [4]:

> "TKIP is a masterpiece of retro-engineering and provides real security in a way that WEP never could".

2.4 AES-CCMP

Nas secções anteriores descrevemos em detalhe o TKIP (obrigatório para o WPA), uma das opções para implementar cifragem e autenticação de mensagens pelo RSN. Nesta secção, vamos estudar o modo principal de o fazer no IEEE 802.11i, que é baseado numa cifra-bloco, designada de AES ("Advanced Encryption Standard").

Primeiro, vamos clarificar o que queremos dizer quando escrevemos "o RSN utilizando o AES". O AES não é um protocolo de segurança, mas sim uma cifra (de bloco). No RSN, o protocolo de segurança construído com base no AES é chamado de *Counter Mode-CBC MAC Protocol* (CCMP). O CCMP define um conjunto de regras que usam a cifra de bloco AES para cifrar e proteger os pacotes (*frames*) do IEEE 802.11. O AES está para o CCMP assim como o RC4 está para o TKIP.

2.4.1 Visão geral do AES

Como já foi dito, o AES é uma cifra de bloco[11]. Usando operações matemáticas e lógicas, o método combina uma chave e um bloco (de dados não cifrados) para produzirem um bloco de dados cifrados. A cifra é simétrica, isto é, usa a mesma chave tanto para cifrar como para decifrar.

O AES é baseado no algoritmo *Rijndael*, inventado por Joan Daeman e Vincent Rijmen. O algoritmo *Rijndael* permite a escolha do tamanho das chaves e dos blocos (separadamente). As opções são 128, 192 ou 256 bits. No caso do IEEE 802.11i, tanto as chaves como os blocos são restringidos a 128 bits de comprimento. Isto conduz a uma simplificação na implementação e "alivia" os utilizadores de terem de fazer ainda outra escolha durante a instalação.

2.4.2 Modos de Operação

Como foi dito, o AES cifra/decifra blocos de comprimento fixo. No entanto, na prática, as mensagens não têm um comprimento fixo, por exemplo, as mensagens Wi-Fi LAN são transmitidas em pacotes (*frames*) de diferentes tamanhos, variando entre 512 e 12000 bits. Assim, para usar o AES, tem de ser definida uma maneira de converter uma mensagem de tamanho arbitrário numa sequência de blocos com o mesmo tamanho, antes da cifragem. De maneira análoga, o método tem de permitir recuperar novamente a mensagem dos blocos cifrados. O método usado para esta conversão/recuperação entre as mensagens e os blocos é designado de *modo de operação* da cifra de bloco.

Existem diferentes modos que podem ser usados em conjunção com o AES. A escolha do modo é muito importante, pois tem implicações, tanto na complexidade de implementação, como na segurança. Modos "maus" podem abrir "buracos" na segurança, apesar da subjacente forte segurança da cifra AES.

O CCMP usa um modo chamado CCM (veja-se a secção seguinte), que é baseado num modo de contagem ("Counter Mode"). Antes de estudarmos este modo, consideremos o

[11] Uma cópia descritiva da cifra AES pode ser encontrada no sítio *http://csrc.nist.gov/encryption/aes/index.html*.

assunto *autenticidade das mensagens*. Para além do AES fornecer um método de cifrar dados, levando a que um atacante não os consiga ler, também é muito importante ter a certeza de que os dados não foram modificados, isto é, que a mensagem é autêntica. Para isto, inclui-se também um valor de verificação de integridade da mensagem (MIC). Tendo em conta eficiência, este valor deve ser calculado usando a cifra AES e, portanto, faz sentido que este modo de operação defina como obter cifragem e autenticação de mensagens.

O modo de contagem caracteriza-se da seguinte maneira: não utiliza directamente o AES para cifrar a informação desejada. Em vez disso, cifra um valor arbitrário chamado de *contador* e faz XOR do resultado com essa informação para produzir o texto cifrado. O contador é geralmente incrementado de um[12] para cada bloco processado sucessivamente (daí o nome!).

Note-se que pelo facto do contador mudar para cada bloco, mesmo que dois blocos sejam iguais, eles serão combinados com contadores diferentes, produzindo assim blocos cifrados distintos. No entanto, este método utilizado em duas mensagens iguais (isto é, enviadas separadamente) produziria o mesmo resultado. Assim, na prática, o contador não começa em 1. Tipicamente é inicializado por um valor designado *nonce*[13] que muda para mensagens sucessivas.

O modo de contagem possui algumas propriedades interessantes. O processo de decifragem é exactamente o mesmo de cifragem, pois $A \oplus A = 0$, para qualquer valor (em bits) de A. Isto significa que só é necessário implementar o bloco de cifragem AES (e não o de decifragem). Outra propriedade útil, para algumas aplicações, é a de a cifragem poder ser feita completamente em paralelo no sentido em que, como todos os valores do *contador* são conhecidos à partida, podem-se ter vários dispositivos de cifragem AES e desse modo cifrar uma mensagem inteira numa única operação em paralelo. A última propriedade que aqui descrevemos é a de que não há problema da mensagem não poder ser particionada num número exacto de blocos. Basta pegar no último bloco (o de

[12]Na prática, o contador pode começar num valor arbitrário e pode ser incrementado de outro valor qualquer ou padrão. O importante é que o receptor que decifra as mensagens conheça o valor inicial e a regra do seu incremento.

[13]A palavra "nonce" pode ser interpretada como "n-once", ou seja, um valor n usado só uma vez. Criptograficamente falando, deve ser altamente improvável que este valor seja usado duas vezes com a mesma chave.

menor comprimento) e fazer XOR com o contador cifrado, usando apenas o número de bits que se necessitar. Desta maneira, o comprimento do texto cifrado ("ciphertext") é exactamente o mesmo da mensagem inicial.

O modo de contagem é usado há mais de 20 anos, estando portanto bem estudado, consequentemente usufruindo de bastante confiança por parte da comunidade criptográfica. A sua simplicidade e maturidade fizeram dele uma opção atractiva para o RSN. No entanto modos de contagem básicos não providenciam autenticidade das mensagens, apenas cifragem. Assim, para o RSN, novas capacidades tiveram de ser adicionadas.

2.4.2.1 Modo de Contagem + CBC MAC: CCM

O modo CCM foi criado[14] especificamente para ser usado no IEEE 802.11i RSN, embora seja aplicável a outros sistemas.

O CCM usa o modo de contagem em conjunção com um método de autenticação de mensagens designado *cipher block chaining* (CBC). O CBC é usado para produzir um MIC (ver secção 2.4.3.2) conduzindo assim à designação[15] CBC-MAC.

Na maioria dos métodos existentes que fazem tanto a cifragem como a autenticação das mensagens, está implícito o facto de que toda a mensagem irá ser cifrada. No entanto, no IEEE 802.11, só parte da mensagem precisa ser cifrada. O cabeçalho, que contém informação como por exemplo as moradas MAC, não pode ser "entregue" cifrado, pelo contrário, tem de ser enviado "em claro" de modo a haver operacionalidade entre os aparelhos. Por outro lado, é essencial que o receptor tenha a certeza que o cabeçalho não foi modificado. Por exemplo, um atacante podia alterar a morada da fonte e, portanto, o "atacado" acidentalmente reenviaria mensagens para o atacante em vez do transmissor original. Assim, o CCM permite que a cifragem seja feita numa sub-parte da mensagem que está autenticada com o CBC-MAC.

Uma regra importante em qualquer esquema de segurança é a de não utilizar a mesma

[14] Os seus inventores foram Doug Whiting, Russ Housley e Niels Ferguson.

[15] O termo usado no standard para o valor de verificação é *message authentication code* (MAC). Mas esta insígnia já é utilizada no IEEE 802.11 com o significado *medium access control*. Assim, para evitar confusões, temos vindo e continuaremos a utilizar MIC em vez de MAC.

chave para duas funções criptográficas separadas. Esta regra parece ser quebrada, em virtude da mesma chave ser usada tanto para a cifragem como para a autenticação. No entanto, embora a mesma chave seja usada, é utilizada em cada caso em conjunção com um vector inicial (IV), e este IV é construído de maneira diferente para o modo de contagem e para o CBC-MAC, conduzindo, com efeito, a duas chaves distintas. A efectividade desta separação foi demonstrada por criptógrafos (ver [6]).

2.4.3 Utilização do CCMP no RSN

As próximas secções descrevem a maneira como são cifrados os pacotes usando o CCMP. Notemos primeiro que o CCMP cifra dados em MPDUs (ver nota 2.3.1). Cada MPDU contém o seu próprio cabeçalho ("header"), onde estão as moradas da fonte e de destino, assim como outro tipo de informação. Vejamos os passos da cifragem de um MPDU, descritos em baixo e representados na figura seguinte:

1) Começa-se com um MPDU não cifrado, completado com o cabeçalho IEEE 802.11 MAC. O cabeçalho inclui as moradas da fonte e do destino, mas os valores de alguns campos não são para já usados, sendo-lhes atribuído o valor 0.

2) O cabeçalho MAC é separado do MPDU. Informação do cabeçalho é extraída e utilizada na criação do MIC (de 8 bytes). É também criado o cabeçalho de 8 bytes-CCMP para posterior inclusão no MPDU.

3) O MIC é agora calculado, de modo a proteger o cabeçalho CCMP, os dados e outras partes do cabeçalho IEEE 802.11. O MIC é concatenado aos dados.

4) A combinação dos dados e do MIC é cifrada. Depois disto, o cabeçalho CCMP é pré-concatenado (com o significado observado na figura).

5) Finalmente, o cabeçalho MAC é restituído na frente do novo MPDU, que está pronto para a transmissão.

2.4.3.1 Cabeçalho do CCMP

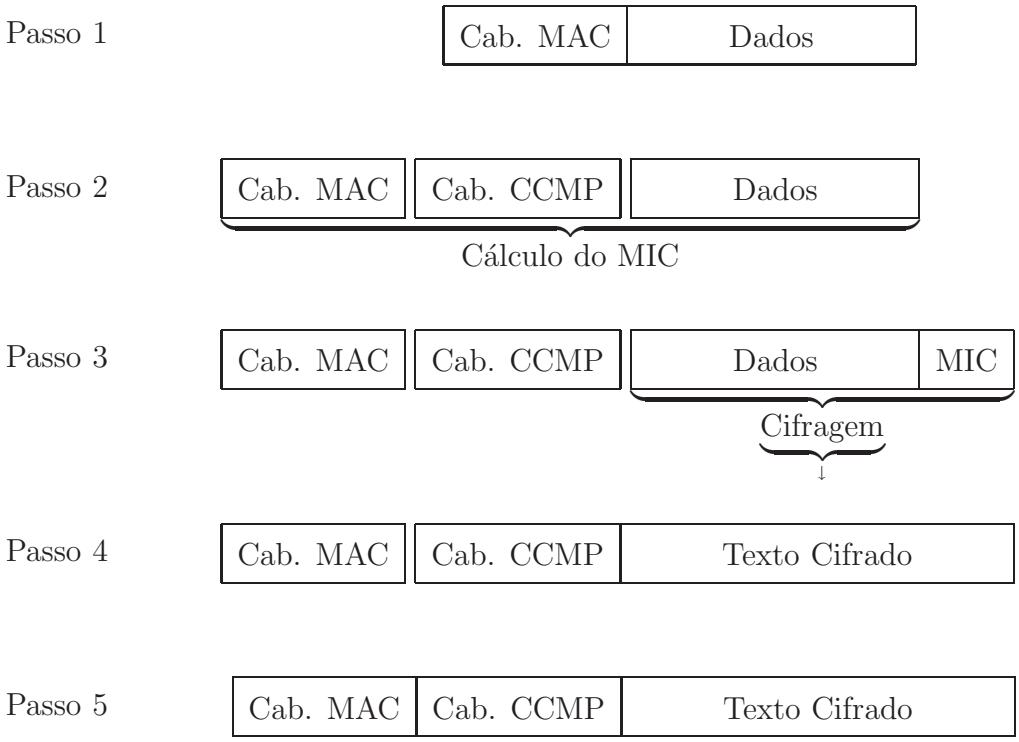


Figura 2.3: Passos no processamento de um MPDU.

O cabeçalho do CCMP é pré-concatenado ao texto cifrado e transmitido sem estar cifrado. Este cabeçalho tem a importante função de providenciar o *número do pacote*[16] ("packet number" - PN) de 48 bits que é usado para protecção a ataques de repetição de mensagem, como também permitir ao receptor derivar o valor do *nonce* que foi usado na cifragem. O formato do cabeçalho do CCMP é semelhante ao do cabeçalho do TKIP; o objectivo é simplificar a implementação nos pontos de acesso que necessitam de receber transmissões de um grupo misto, de TKIP e CCMP.

2.4.3.2 Cálculo do MIC

O cálculo do MIC é feito usando o CBC-MAC, que é conceptualmente simples:

1) Cifra-se o primeiro bloco da mensagem usando o AES.

2) Faz-se XOR do resultado com o segundo bloco e depois cifra-se o resultado.

3) Faz-se XOR do resultado com o bloco seguinte e cifra-se o resultado... e assim sucessivamente.

[16]Este valor é incrementado por cada pacote processado.

O MIC final é um bloco de 128 bits, mas como só são necessários 64 bits, para o CCMP descartam-se os primeiros 64 bits do resultado. No CCMP, o primeiro bloco para o cálculo do CBC-MAC não é "retirado" directamente do MPDU, sendo formado de uma maneira especial, utilizando um valor *nonce* de 104 bits. O formato do primeiro bloco é apresentado na figura seguinte, contendo um *nonce* e dois outros campos: **Flag** e **DLen**.

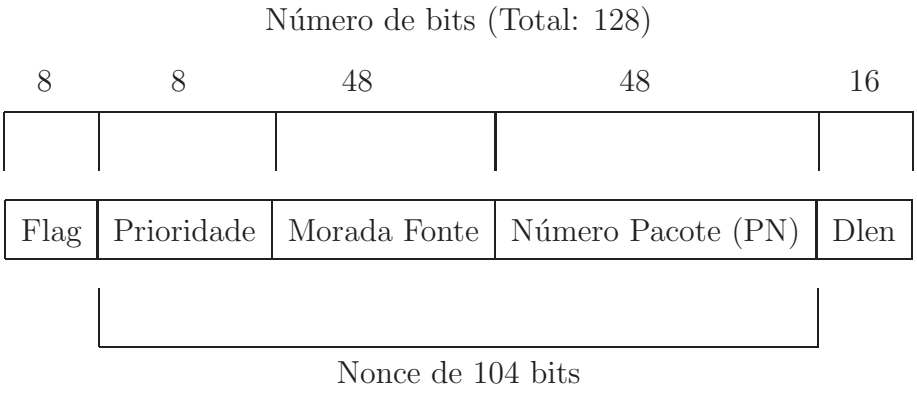


Figura 2.4: Construção do primeiro bloco para o CBC-MAC.

O *nonce* garante uma certa "frescura" ao processo, assegurando que cada cifragem usa dados nunca usados anteriormente (sob uma certa chave). Poderíamos pensar em usar simplesmente o PN para o *nonce* pois ele é incrementado para cada MPDU e portanto nunca se repete. No entanto, relembre-se que a chave é usada pelo menos por duas partes numa comunicação e cada uma destas partes pode, em algum momento, usar um PN que já foi utilizado por outra parte, violando a regra "usar uma vez por chave". Para evitar este problema, o *nonce* é formado combinando o PN com a morada MAC da fonte.

Um outro campo incluído no *nonce* é designado por *Prioridade* ("Priority"). Este campo prende-se com características dos dados transmitidos (áudio, vídeo, etc...).

Os outros dois campos que, juntamente com o *nonce*, formam o primeiro bloco para o CBC-MAC (Figura 2.4) têm as seguintes características: o campo Flag tem o valor fixo 01011001 e indica, entre outras coisas, que o MIC tem 64 bits. O campo Dlen indica o tamanho do texto a ser cifrado ("plaintext data").

Assim que este primeiro bloco estiver preparado, o valor MIC é calculado como

vimos no início desta secção, incorporando o cabeçalho MAC, o cabeçalho CCMP e a mensagem (Figura 2.3).

2.4.3.3 Cifrar um MPDU

Quando o MIC tiver sido calculado e concatenado à mensagem, começar-se-á a cifrar (usando o modo de contagem) o MPDU (note-se que o que se vai cifrar são os dados juntamente com o MIC - ver figura 2.3).

O *contador*[17] é construído de uma maneira quase idêntica ao do primeiro bloco para o MIC. De facto, o valor *nonce* é calculado de maneira igual ao do MIC e inclui a morada MAC da fonte, o número do pacote e o campo Prioridade. A diferença é que este valor é junto com dois campos: **Flag** e **Contador** ("Ctr"), um dos quais diferente do do MIC.

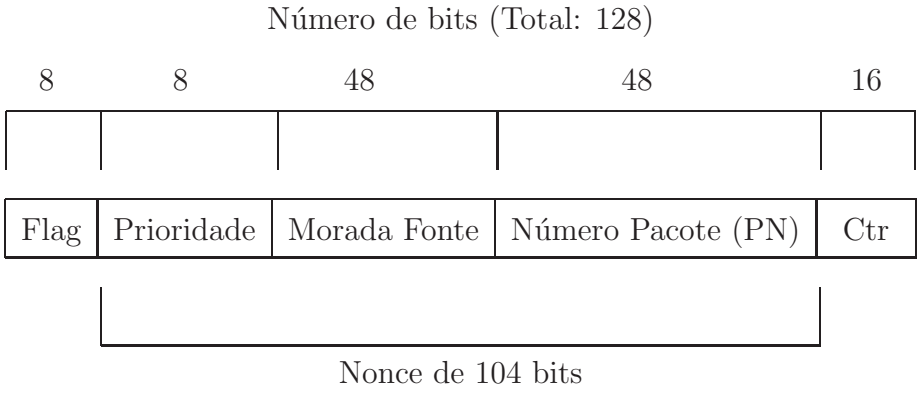


Figura 2.5: Construção do contador para o CCMP AES-modo de contagem.

O valor de Ctr começa em 1 e é aumentado de um à medida que o processo decorre. Como o valor *nonce* está fixo para cada MPDU, e o Ctr tem 16 bits de comprimento, tem-se a garantia de não haver valores do *contador* iguais, para mensagens com menos de 65536 blocos. Isto é o suficiente, mesmo para os "maiores" MPDU's permitidos no IEEE 802.11.

Uma vez iniciado o contador, a cifragem faz-se de acordo com o descrito na secção 2.4.2, isto é, cada valor sucessivo do contador é cifrado usando o AES, sendo depois feito XOR com o texto antes de cifrar de modo a produzir texto cifrado.

[17]Blocos construídos para serem usados no processo de cifragem.

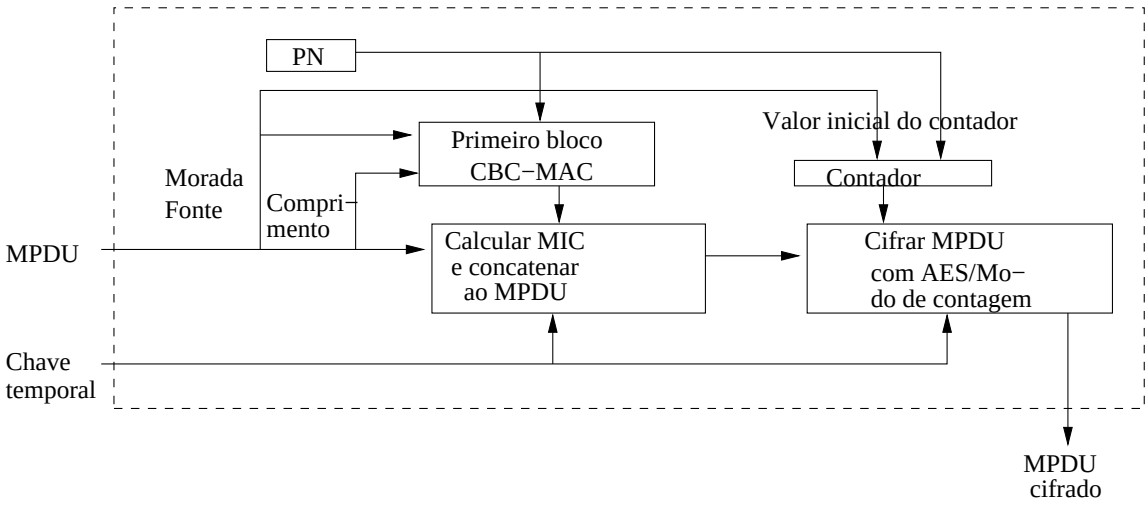


Figura 2.6: Bloco de cifragem no CCMP

2.4.3.4 Decifrar um MPDU

Quando um MPDU cifrado é "entregue" ao receptor, o primeiro passo a fazer é ter a chave correcta para o decifrar. Esta decifragem é só um passo de um processo que se chama *descapsulação* ("decapsulation"). O PN é enviado (sem estar cifrado) no cabeçalho CCMP, portanto, a primeira coisa que o receptor faz é ler o PN e compará-lo com o último pacote recebido. Se o número for menor ou igual ao último recebido, então o pacote é descartado.

Assumindo que o passo anterior não ocorreu, o próximo passo é decifrar usando o modo de contagem-AES. Para isso, é necessário calcular o valor inicial do contador, que tem de coincidir com o valor usado na cifragem. O PN é combinado com a morada da fonte e com o campo prioridade para criar o *nonce*. Isto é depois combinado com o (conhecido) valor Flag e com o valor inicial Ctr, de modo a criar o contador inicial. Note-se que este contador inicial pode ser calculado por um atacante mas, em princípio, se não souber a chave secreta, não lhe servirá de nada. A decifragem procede-se da mesma maneira da cifragem. No final deste processo obtém-se, em particular, o valor MIC que tem de se verificar se é o correcto. O modo de o fazer é o mesmo aquando do seu cálculo pelo emissor (ver secção 2.4.3.2). É claro que se a mensagem original não tiver sido modificada, o receptor (detentor da chave secreta) vai obter o valor correcto.

Se o valor não coincidir, há uma evidência de ataque e assim o pacote irá ser descartado.

Finalmente, quando o MPDU tiver sido decifrado, o MIC e o cabeçalho CCMP são retirados de modo aos restantes dados poderem ser restituídos com os que vão chegando de modo a formar o MSDU original.

Um grande número de sistemas Wi-Fi foram desenvolvidos baseados no algoritmo de cifragem RC4. Foi assim para o WEP, tendo sido depois incluído no TKIP de modo a que fosse possível fazer actualizações dos produtos existentes no mercado até então. Posteriormente, quando o grupo IEEE 802.11 começou (em 2000) a pensar em desenvolver uma nova solução de segurança do zero, escolheu a cifra AES. Um dos motivos desta escolha foi o facto de nessa altura outra agência se debater com o mesmo problema (entenda-se a escolha de uma cifra), nada mais nada menos que a "National Institute for Science and Technology" (NIST). Devido ao extenso processo de revisão efectuado, conduzindo posteriormente à selecção do AES por parte da NIST, esta cifra era tida como bastante segura, levando a que fosse também adoptada para o IEEE 802.11. Esta secção expôs a maneira como o AES foi incorporado na solução de segurança RSN.

Capítulo 3

Ataque ao TKIP

Nas secções 2.3 e 2.4 descrevemos o TKIP e o AES-CCMP, respectivamente. Como vimos, através do TKIP foram implementadas uma série de novas medidas com vista a colmatar as deficiências existentes no protocolo de segurança WEP. O AES-CCMP foi criado de raiz, podendo-se, em certo sentido, dizer que é mais seguro que o TKIP.

Neste capítulo vamos descrever um ataque ao protocolo de segurança TKIP, seguindo as ideias apontadas em [11]. Este ataque revela algumas fraquezas na concepção do protocolo, podendo assim comprometer todo o sistema.

3.1 Fraquezas da chave temporal no TKIP

Nesta secção pretendemos descrever e fazer alguma análise a um ataque à *função de compressão*[1], estudada na secção 2.3.3.2. Como consequência, pode ser possível recuperar a chave temporal que é utilizada pelo TKIP.

Começamos por assumir que o atacante possui algumas (menos de 10) chaves RC4, obtidas sob o mesmo TSCU (os autores referem que a obtenção de tais chaves depende da implementação. Eles afirmam que o seu principal intuito é destacar pontos fracos existentes no TKIP [11]).

Com esta hipótese em mente, mostraremos que um atacante pode determinar a chave temporal (TK), e portanto, decifrar qualquer pacote tal como um legítimo utilizador.

[1]Por esta função entenda-se todo o algoritmo da *mistura da chave*.

Observação: O valor TSCU só é alterado ao fim de 2^{16} pacotes, portanto, P1K corresponde a um valor fixo durante esse período.

Este ataque faz uso do facto de que 8 bits da TK podem ser calculados directamente a partir de uma chave RC4. Os valores PPK obtidos na fase 2 são conhecidos através da chave RC4, em particular o valor PPK_5. Relembrando o passo 3 da fase 2 do algoritmo, temos que

$$\mathrm{RC4Key}_3 = \text{Primeiro byte } (\mathrm{PPK}_5 \oplus \rightarrow (\mathrm{TK}_1 \cap \mathrm{TK}_0)),$$

sendo portanto possível obter o bit menos significativo de TK_1, assim como os sete bits mais significativos de TK_0.

O resto da secção descreve o ataque (dividido em seis partes), mostrando como podemos obter o resto da chave temporal TK. Nas figuras seguintes, as setas pretas indicam que sabemos o valor que transportam, as setas a tracejado indicam que desconhecemos o valor e duas setas em conjunto indicam que existem duas escolhas possíveis para o seu valor.

Parte 1 - Descobrir TK_{10} e TK_{11}

A figura 3.1 corresponde à parte do algoritmo necessária para descobrirmos o valor de TK_{10} e de TK_{11}.

A ideia consiste em percorrer o algoritmo do fim para o início, isto é, como sabemos os valores de PPK_3, PPK_4 e PPK_5, podemos dar valores a TK_{10} e a TK_{11}, obtendo assim um conjunto de valores para $\mathrm{P1K}_4$. Veremos em seguida como obter os valores correctos de TK_{10} e TK_{11}.

As setas na figura[2] indicam valores de entrada e saída.

A primeira operação a fazer é uma rotação de uma casa para a direita de PPK_3 e PPK_4. A inversa da adição mod 2^{16} é a subtracção mod 2^{16}. Assim, podemos calcular até ao ponto em que os valores dependem de TK_{10} e TK_{11}. Atribuindo valores a $\mathrm{TK}_{10} \| \mathrm{TK}_{11}$, permite-nos fazer XOR com o valor da S-box e, juntamente com duas subtracções mod 2^{16}, calcular $\mathrm{P1K}_4$ (relembre-se que o valor de TSC_0 é conhecido, pois é enviado sem

[2] A notação utilizada em [11] para representar $\mathrm{TK}_i \cap \mathrm{TK}_j$ é $\mathrm{TK}_i \| \mathrm{TK}_j$. Optamos por mantê-la nesta parte do documento.

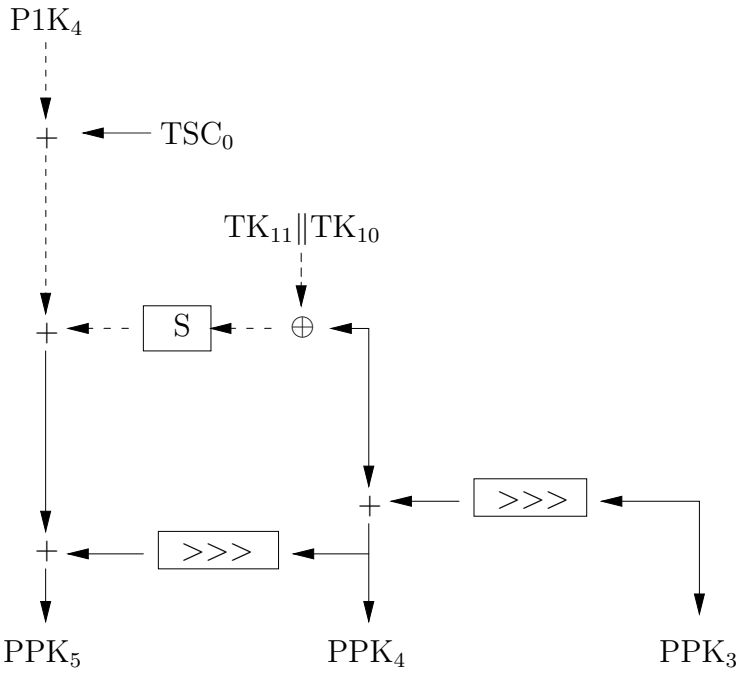


Figura 3.1: Parte da fase 2 necessária para calcular TK_{10} e TK_{11}.

estar cifrado em cada pacote). Temos assim uma bijecção entre os valores de $TK_{11}\|TK_{10}$ e os valores de $P1K_4$.

Os autores afirmam em [11] e passamos a citar, o seguinte:

> Using two or three RC4-keys should be enough to eliminate all but the correct values of TK_{10} and TK_{11}.

Não havendo nada escrito no artigo que justifique tal afirmação, decidimos fazer alguma investigação no intuito de confirmarmos (ou não) a afirmação supracitada.

Comecemos por admitir que estamos na posse de duas chaves RC4. Então temos duas funções bijectivas, digamos f e g, do conjunto dos pontos de $TK_{11}\|TK_{10}$ no conjunto dos pontos de $P1K_4$. Pela observação feita anteriormente sabemos que estas duas funções têm a seguinte propriedade:

$$f(a) = b \text{ e } g(a) = b, \text{ para alguns } a, b \in \{0,1\}^{16}. \tag{3.1}$$

É claro que sem mais restrições, o par (a, b) não é o único a verificar tal condição. Assim, decidimos fazer um estudo probabilístico da situação, que descrevemos a seguir.

Sejam f e g duas funções escolhidas aleatoriamente do conjunto

$$F = \{h : \{0,1\}^{16} \to \{0,1\}^{16} : h \text{ é bijectiva}\},$$

e que satisfazem a propriedade (3.1). Pretendemos determinar a probabilidade de não existirem mais pontos a satisfazerem essa propriedade. Escrevendo a e b na base decimal, temos que (veja-se apêndice B)

$$f = \begin{pmatrix} 1 & 2 & \dots & a & \dots & 2^{16} \\ f(1) & f(2) & \dots & b & \dots & f(2^{16}) \end{pmatrix},$$

e

$$g = \begin{pmatrix} 1 & 2 & \dots & a & \dots & 2^{16} \\ g(1) & g(2) & \dots & b & \dots & g(2^{16}) \end{pmatrix}.$$

Formemos um nova função

$$h = \begin{pmatrix} b & f(1) & f(2) & \dots & f(a-1) & f(a+1) & \dots & f(2^{16}) \\ b & g(1) & g(2) & \dots & g(a-1) & g(a+1) & \dots & g(2^{16}) \end{pmatrix}.$$

Reordenando (se necessário) as colunas de h, podemos obter uma nova bijecção

$$h' = \begin{pmatrix} 1 & 2 & \dots & 2^{16}-1 \\ h(1) & h(2) & \dots & h(2^{16}-1) \end{pmatrix},$$

A probabilidade pretendida equivale a determinar a probabilidade desta função não ter pontos fixos. Para isso, considere-se $n = 2^{16} - 1$ e observe-se que o número de casos favoráveis a esse acontecimento é dado por D_n (B.2). Assim a probabilidade pretendida é:

$$P = \frac{D_n}{n!} = \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^n \frac{1}{n!}\right) \approx e^{-1} \approx 37\%.$$

Este valor fica aquém das expectativas. Isto levou-nos a contactar (via e-mail) os autores do artigo no intuito de obter alguma informação adicional que nos conduzisse a um "resultado mais satisfatório". Neste contacto (informal) que mantivemos, foi-nos dito que não fizeram os cálculos que acabamos de apresentar mas apenas implementações computacionais, que demonstraram os resultados apresentados no artigo.

Prosseguindo a nossa investigação, tentamos fazer cálculos análogos aos anteriores, mas agora na hipótese de termos três chaves RC4. Estes cálculos tornam-se fastidiosos. Decidimos então fazer uma simulação computacional, que apresentamos no apêndice C. Os resultados nela obtidos indicam valores mais condizentes às afirmações proferidas pelos autores em [11]. Por exemplo, pondo $n = 2^5$, o valor sugerido pela simulação computacional para a probabilidade pretendida foi, aproximadamente, 97%.

Parte 2 - Descobrir TK_8 e TK_9

A figura 3.2 corresponde à parte do algoritmo necessária para descobrirmos o valor de TK_8 e de TK_9.

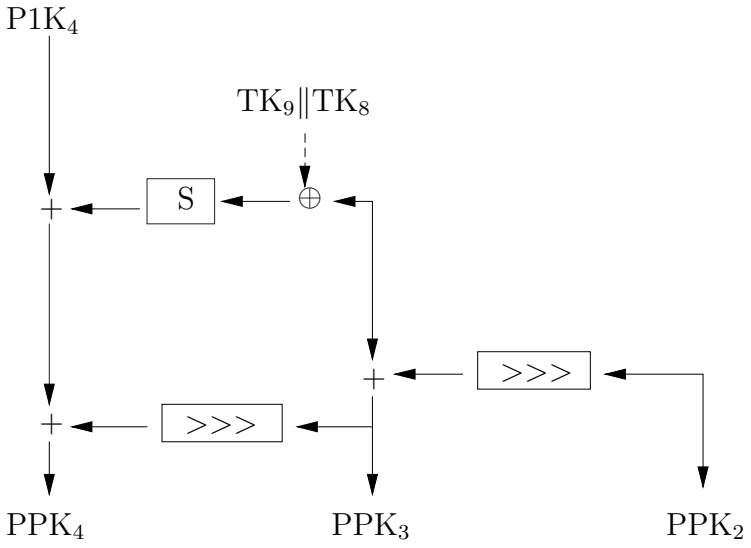


Figura 3.2: Parte da fase 2 necessária para calcular TK_8 e TK_9.

Note-se que no passo anterior, ao descobrirmos TK_{10} e TK_{11}, obtivemos o valor correcto de $P1K_4$. Portanto, os valores de TK_8 e TK_9 podem ser determinados correctamente, sem recurso a nenhuma tentativa.

Parte 3 - Descobrir TK_6 e TK_7

Como se pode observar na figura 3.3, os valores de TK_6 e TK_7 podem ser determinados exactamente da mesma maneira que os valores de TK_{10} e TK_{11}.

Parte 4 - Descobrir TK_0, TK_1, TK_{12} e TK_{13}

Considere-se a figura 3.4.

Como vimos anteriormente, só oito bits de TK_0 e TK_1 são desconhecidos. Para calcular o valor de $P1K_0$, podemos usar o facto do valor de $P1K_4$ ser conhecido, mas agora é necessário percorrer os valores de TK_{12} e TK_{13}, como também os 8 bits desconhecidos de TK_0 e TK_1, perfazendo um total de 24 bits. Novamente, para uma determinada tentativa nos valores de TK_0, TK_1, TK_{12} e TK_{13}, obtemos um valor para $P1K_0$.

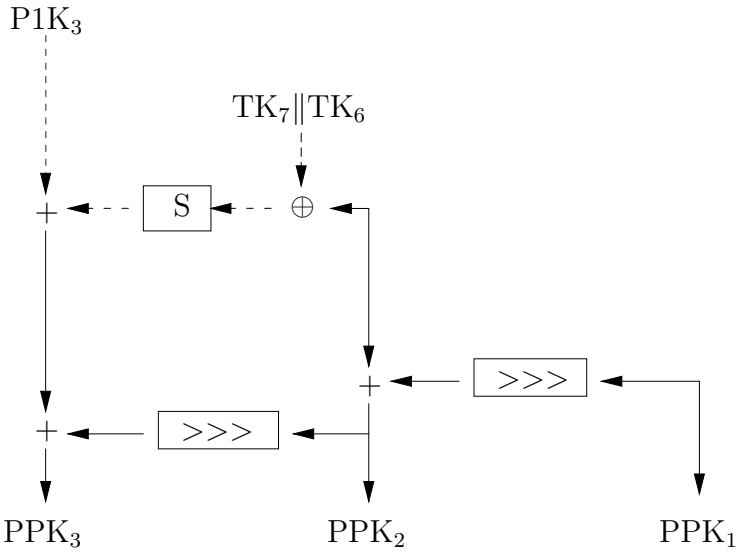


Figura 3.3: Parte da fase 2 necessária para calcular TK_6 e TK_7.

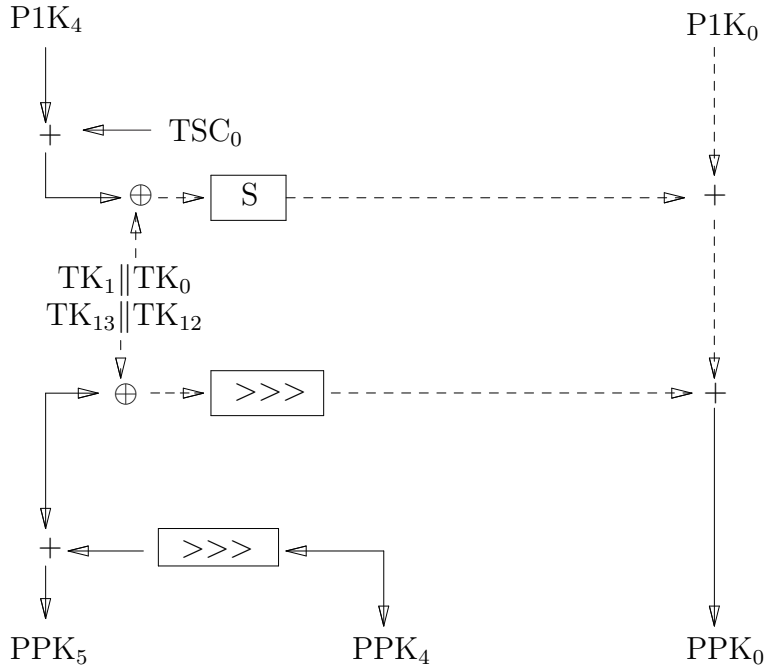


Figura 3.4: Parte da fase 2 necessária para calcular TK_0, TK_1, TK_{12} e TK_{13}.

Atente-se agora no seguinte: suponhamos que temos os valores correctos de TK_0, TK_1, TK_{12} e TK_{13} (designemos este conjunto de valores por TC), e consideremos os mesmos valores, trocando o bit menos significativo de TK_{12} (denotado por TE). Vamos comparar os valores obtidos na figura 3.4 para $P1K_0$ em ambos os casos.

Os valores de TK_0 e TK_1 são os correctos, portanto os valores da seta horizontal na parte de cima vão ser iguais em ambos os casos. Fazendo a rotação (que aparece no meio da figura), os valores obtidos diferem apenas no bit mais significativo. Como as restantes operações para calcular $P1K_0$ consistem de subtracções mod 2^{16}, os valores de $P1K_0$ obtidos através de TE vão diferir apenas no bit mais significativo dos valores de $P1K_0$ obtidos por TC, para cada chave RC4 disponível. Em particular, os valores de $P1K_0$ calculados através de TE vão ser todos iguais. Isto significa que o bit menos significativo de TK_{12} não pode ser determinado nesta altura, no entanto, facilmente se determinará mais à frente. Tem-se também que só ficamos a conhecer os 15 bits menos significativos de $P1K_0$.

Note-se que nesta parte temos funções com domínio e conjunto de chegada $\{0,1\}^{23}$ e $\{0,1\}^{16}$, respectivamente.

Parte 5 - Descobrir TK_2, TK_3, TK_{14} e TK_{15}

A figura 3.5 mostra a parte mais dispendiosa do ataque.

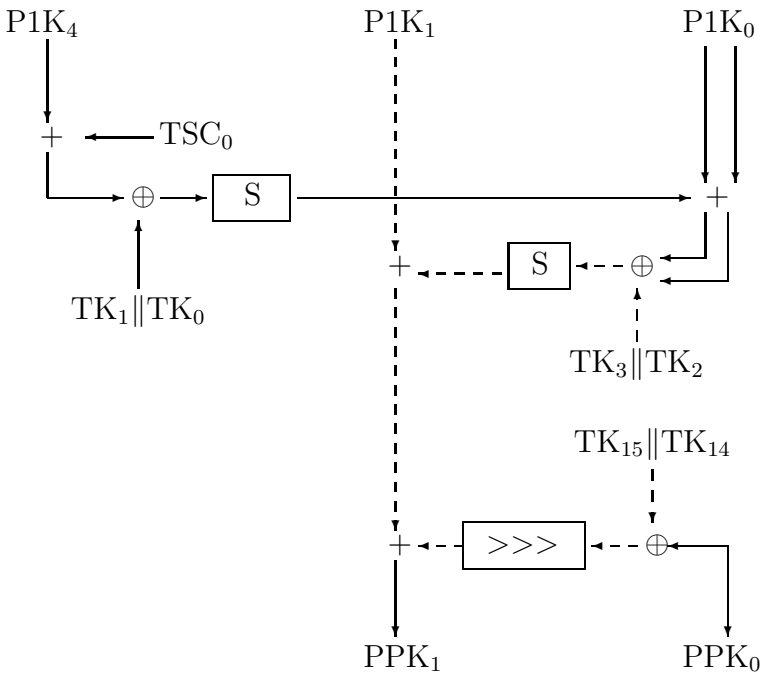


Figura 3.5: Parte da fase 2 necessária para calcular TK_2, TK_3, TK_{14} e TK_{15}.

Aqui pretende-se obter os valores correctos de TK_2, TK_3, TK_{14} e TK_{15}. Para cada tentativa (nestes valores), calculamos os valores de $P1K_1$ (para cada chave RC4). Como vimos anteriormente, $P1K_0$ pode tomar um de dois valores, e portanto, teremos de fazer

a verificação através do valor de $P1K_1$. Para o valor correcto de TK_{14} e de TK_{15} vão existir dois valores sugeridos para $TK_3\|TK_2$, um para cada $P1K_0$.

Analogamente ao que vimos na parte 4, também não conseguimos aqui determinar o bit menos significativo de TK_{14}. Isto significa que apenas iremos fazer tentativas em 31 bits (16 de $TK_3\|TK_2$ e 15 de $TK_{15}\|TK_{14}$). Para cada tentativa, teremos de fazer a verificação em $P1K_1$ duas vezes (pois temos dois valores possíveis para $P1K_0$). Assim, a complexidade desta parte é de $2 * 2^{31} = 2^{32}$ verificações. Esta parte domina a complexidade total do ataque. No final teremos quatro conjuntos de valores possíveis de TK_2, TK_3, TK_{12} e TK_{14}, pois não sabemos o bit menos significativo de TK_{12} nem de TK_{14}.

Parte 6 - Descobrir TK_4 e TK_5

Finalmente, a figura seguinte mostra como os valores de TK_4 e TK_5 podem ser obtidos.

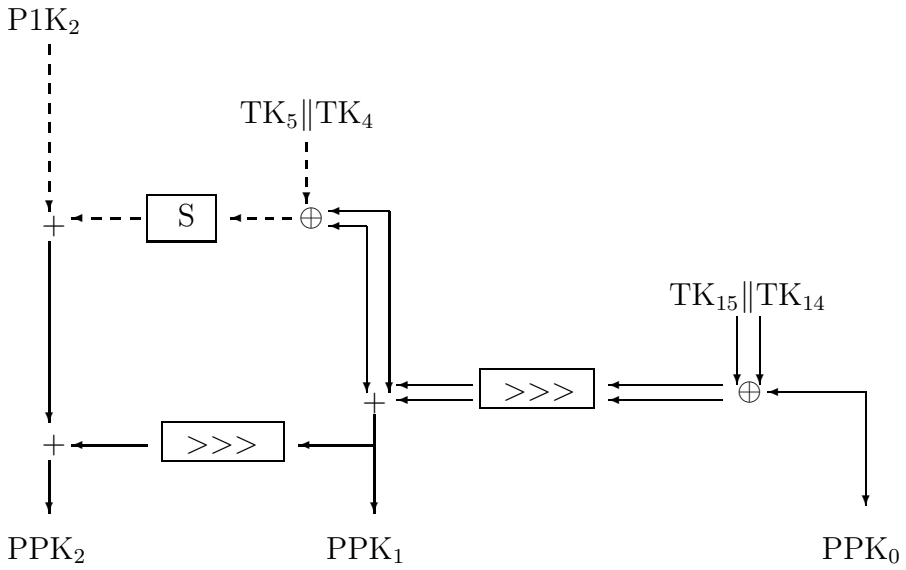


Figura 3.6: Parte da fase 2 necessária para calcular TK_4 e TK_5.

Aqui, cada tentativa em TK_4 e TK_5 pressupõe, tal como anteriormente, a análise nas duas possibilidades do bit menos significativo de TK_{14}. Para cada uma destas, vai existir um valor de TK_4 e TK_5 sugerido como correcto.

Descobrir o bit menos significativo de TK_{12} e de TK_{14}

Depois de efectuar todos os seis passos descritos anteriormente, ficamos com quatro valores possíveis para a chave temporal TK (completa). A cada possível valor de TK corresponde um valor de P1K. O valor correcto de TK pode ser obtido determinando a saída da fase 1 do algoritmo para cada um dos candidatos a TK, e verificar qual deles tem como saída o seu P1K correspondente. A probabilidade de que uma TK errada resulte no seu P1K correspondente é $4 * 2^{-80} = 2^{-78}$, pois P1K tem 80 bits.

Discussão do ataque

Os autores fazem a análise da complexidade de um ataque utilizando apenas duas chaves RC4 (ver secção III.H. de [11]), apresentando o resultado de aproximadamente $\mathcal{O}\ (2^{38})$. Escrevem também que (na ausência de chaves RC4) é possível montar um ataque à chave temporal TK com complexidade em tempo $\mathcal{O}\ (2^{105})$, que é uma redução significativa comparativamente a um ataque de força bruta, que tem complexidade em tempo $\mathcal{O}\ (2^{128})$, embora não seja possível de o concretizar na prática.

Por fim, os autores referem que implementaram o ataque descrito nas seis partes anteriores num computador para verificar a sua exactidão. Utilizaram um processador Intel Pentium 4 2.53 GHz, tendo demorado 6-7 minutos a recuperar TK, tendo quatro ou mais chaves RC4. Com duas chaves RC4, o ataque demorou aproximadamente 15 horas.

Este capítulo descreve a possibilidade de recuperar a chave temporal TK utilizada pelo TKIP para proteger mensagens em redes sem fios. A recuperação de TK pressupõe que um atacante esteja de posse de pelo menos duas chaves RC4 e, neste sentido, a análise feita não implica que a segurança do WPA (TKIP) tenha sido violada, mas assinala a importância de manter todas as chaves RC4 em segredo.

Conclusão

A criptografia tem uma longa e fascinante história. Inicialmente, os principais praticantes desta "arte" eram pessoas envolvidas em questões militares ou governamentais em geral. A criptografia foi usada como uma ferramenta para proteger segredos e estratégias nacionais.

A proliferação dos computadores e dos sistemas de comunicação nos anos 60 do século XX trouxe com ela a demanda pelo sector privado de obter meios de proteger informação em formato digital e de fornecer serviços de segurança.

Com este trabalho, pretendemos descrever os protocolos de segurança das redes sem fios Wi-Fi, como também expor as suas falhas, permitindo ao leitor ter a percepção do nível de segurança que tem nos seus produtos.

Até 2001, o método de segurança existente para proteger as redes Wi-Fi era o WEP, que se revelou ser bastante ineficiente tendo sido descobertas bastantes falhas. Como resultado, em 2002, houve um esforço enorme feito pela indústria no sentido de desenvolver um substituto para o WEP, de maneira a que os sistemas existentes pudessem ser actualizados.

Em finais de 2002, a Wi-Fi *Alliance* anunciou o WPA, que foi especificamente concebido para permitir actualizações à maioria dos sistemas existentes, através de *software* próprio. Com o WPA, todas as falhas existentes no WEP foram reparadas. Actualmente, o método usual de segurança é o AES-CCMP.

Após a leitura deste manuscrito, podemos certamente concluir que, é extremamente difícil conceber protocolos de segurança para sistemas de comunicação em geral, mas muito particularmente para sistemas de redes sem fios onde existe a agravante de qualquer pessoa poder ter acesso à informação muito facilmente, em virtude do meio em que se propaga.

Esperamos, também, com este trabalho, dar a entender que esta é uma área com um enorme potencial a nível de investigação. Como vimos na parte final do trabalho, novas medidas terão de ser tomadas para eliminar as fraquezas existentes nas novas soluções de segurança. Esta parece ser de facto uma área infindável, uma vez que sempre que um problema é resolvido, logo aparecem novos problemas...

Apêndice A

O algoritmo que a seguir se apresenta foi executado utilizando o programa *Maple* e com ele pretende-se "demonstrar" que 60 chaves x-boas são suficientes para obter correctamente o byte x da sequência de bytes da chave secreta usada pelo WEP (ver secção 1.2.3).

```
n := 0: k := 60
for m from 1 to 10000 do
  a := array[0..255]:
   for i from 0 to 255 do
   a[i] := 0:
   od:

   for i from 1 to k do
    roll := rand(1..25500):
    s := roll ():
      if s <= 1275 then a[0] := a[0]+1:
      else
      x := ceil ((s−1275)/95):
      a[x] := a[x]+1:
      fi:
   od:

   r := 0:
    for i from 1 to 255 do
      if a[0]>a[i] then r := r+1:
```

```
    fi:
  od:
 if r = 255 then n := n+1:
 fi:
od:
s := evalf(n/m);
```

Tendo uma lista de números entre 0 e 255, o que fizemos foi criar uma função que dá o número 0 com probabilidade $0,05$ ($= \frac{1275}{25500}$) e i ($1 \leq i \leq 255$) com probabilidade $\frac{0,95}{255}$ ($= \frac{95}{25500}$). Geramos k valores aleatoriamente[1] e registamos o número de ocorrências de cada um dos valores i ($i = 0, \ldots, 255$). O nosso teste consiste em verificar se o número 0 ocorreu mais vezes (estritamente) do que qualquer um dos outros números entre 1 e 255. Esta experiência é repetida 10000 vezes.

Para $k = 55$, obtivemos o valor $s \approx 46\%$.

Para $k = 60$, obtivemos o valor $s \approx 51\%$.

Para $k = 65$, obtivemos o valor $s \approx 54\%$.

[1]No algoritmo que aqui apresentamos, optamos por explicitar para k o valor de 60.

Apêndice B

Definição B.0.1 *Seja* $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ *uma função bijectiva (a que chamamos permutação) que representamos da seguinte seguinte maneira:*

$$f = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix}.$$

Diz-se que f *é um desarranjo*[1] *se* $f(x) = x$ *não tiver solução.*

Por exemplo,

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

é um desarranjo, enquanto que

$$g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

não, pois $g(1) = 1$.

Suponhamos que queremos determinar o número de desarranjos do conjunto $\{1, 2, \ldots, n\}$. Seja D_n esse valor (a letra D vem da palavra desarranjo). Para calcular D_n vamos utilizar um teorema muito útil em Análise Combinatória (uma demonstração deste teorema pode ser encontrada em [15], páginas 93-94):

Teorema B.0.1 (Princípio da inclusão-exclusão) *Sejam* $n \in \mathbb{N}$ *e* $A_1, A_2, \ldots, A_n$

[1] Por vezes também se denomina de *permutação caótica*.

conjuntos finitos. Então

$$\begin{aligned}\#(A_1 \cup A_2 \cup \ldots \cup A_n) = &\sum_{i=1}^{n} \#(A_i) - \sum_{1\le i<j}^{n} \#(A_i \cap A_j) + \sum_{1\le i<j<k}^{n} \#(A_i \cap A_j \cap A_k) \\ &- \sum_{1\le i<j<k<p}^{n} \#(A_i \cap A_j \cap A_k \cap A_p) + \ldots \\ &+ (-1)^{n-1} \#(A_1 \cap A_2 \cap \ldots \cap A_n).\end{aligned} \tag{B.1}$$

onde $\#(B)$ denota o número de elementos de um conjunto B (finito).

Se definirmos por A_i o conjunto das permutações de $\{1, 2, \ldots, n\}$ tal que $f(i) = i,\ i \in \{1, 2, \ldots, n\}$, queremos determinar o número de elementos que não pertencem a nenhum dos A_i's, isto é, o número de elementos no complementar da união dos A_i's. Logo

$$\begin{aligned}D_n = n! - &\sum_{i=1}^{n} \#(A_i) + \sum_{1\le i<j}^{n} \#(A_i \cap A_j) - \sum_{1\le i<j<k}^{n} \#(A_i \cap A_j \cap A_k) + \ldots \\ &+ (-1)^n \#(A_1 \cap A_2 \cap \ldots \cap A_n).\end{aligned}$$

Como existem n termos na primeira soma, C_2^n termos na segunda, C_3^n na terceira, ..., $C_n^n = 1$ na última (em que $C_k^n = \frac{n!}{k!(n-k)!}$), e

$$\begin{aligned}\#(A_i) &= (n-1)!, \\ \#(A_i \cap A_j) &= (n-2)!, \\ \#(A_i \cap A_j \cap A_k) &= (n-3)!, \\ &\vdots \\ \#(A_1 \cap A_2 \cap \ldots \cap A_n) &= 1,\end{aligned}$$

temos

$$\begin{aligned}D_n &= n! - n(n-1)! + \frac{n!}{2!(n-2)!}(n-2)! - \frac{n!}{3!(n-3)!}(n-3) + \ldots + (-1)^n 1 \\ &= n! - \frac{n!}{1!} + \frac{n!}{2!} - \frac{n!}{3!} + \ldots + (-1)^n \frac{n!}{n!}.\end{aligned}$$

Colocando $n!$ em evidência obtemos:

$$D_n = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^n \frac{1}{n!}\right). \tag{B.2}$$

Se, por exemplo, quisermos determinar o número de desarranjos de três objectos: 1, 2 e 3, temos

$$D_3 = 3!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!}\right) = 2.$$

De facto, escrevendo as seis permutações dos elementos do conjunto $\{1, 2, 3\}$,

$$f_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, f_2 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, f_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

$$f_4 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, f_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}, f_6 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix},$$

verifica-se que só f_4 e f_5 são desarranjos.

Apêndice C

O algoritmo que a seguir se apresenta foi executado utilizando o programa *Maple* e tem por objectivo obter uma estimativa do valor da probabilidade de, escolhidas três permutações aleatóriamente, elas não conterem pontos em comum (ver secção 3.1).

```
with (combinat, randperm):
n := 2^5: m := 0: r := 10000:
for i from 1 to r do
    p1 := randperm(n):
    p2 := randperm(n):
    p3 := randperm(n):
     for j from 1 to n do
      if p1[j]=p2[j] and p1[j]=p3[j] and p2[j]=p3[j] then j := n+1:
      else
       if j = n then m := m+1: fi:
      fi:
     od:
od:
s := evalf(m/r);
```

Este algoritmo percorre as seguintes etapas: primeiro são geradas três permutações aleatoriamente. Seguidamente é feita a verificação da existência ou não de pontos em comum entre elas. A experiência é repetida 10000 vezes e o seu resultado é obtido na variável s.

Para $n = 2^5$, obtivemos $s \approx 97\%$.

Para $n = 2^7$, obtivemos $s \approx 99\%$.

Bibliografia

[1] Barker, E., Barker, W., Burr, W., Polk, W., Smid, M., *Recommendation for Key Management - Part 1: General (Revised)*. NIST Special Publication 800-57, May 2006.

[2] Borisov, N., Goldberg, I. and Wagner, D., *Intercepting mobile communications: The insecurity of 802.11*. In MOBICOM 2001, Rome, Italy, July 2001.

[3] Dawson, E. and Nielsen, L., *Automated cryptanalysis of XOR plaintext strings*. Cryptologia, (2): 165-181, Apr.1996.

[4] Edney, J., Arbaugh, W. A., *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Boston, Addison-Wesley, 2005.

[5] Fluhrer, S. R., Mantin, I. and Shamir, A., *Weaknesses in the key scheduling algorithm of RC4*, SAC: Annual International Workshop on Selected Areas in Cryptography, LNCS, 2001.

[6] Jonsson, J. 2002. On the security of CTR+CBC-MAC. In *SAC 2002 - Ninth Annual Workshop on Selected Areas of Cryptography*.

[7] LAN MAN Standards Committe of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11, 1999 Edition*, 1999.

[8] LAN MAN Standards Committe of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *Amendment 6: Medium Access Control (MAC) Security Enhancements*. IEEE Std 802.11i, 2004.

[9] Matthew, S. Gast, *802.11 Wireless Networks: The Definitive Guide*. Second Edition, O'Reilly, 2005.

[10] Menezes, A. J., Van Oorschot, P. C. and Vanstone, S. A., eds. 1996. *Handbook of Applied Cryptography*. New York: CRC Press.

[11] Moen, V., Raddum, H. and Hole, K. J., *Weakness in the Temporal Key Hash of WPA*. ACM SIGMOBILE Computing and Communications Review, Vol. 8, Issue 2, ACM Press, pp. 76-83, 2004.

[12] Morgado, A., Carvalho, J., Carvalho, P., Fernandez, P., *Análise Combinatória e Probabilidade*. Gráfica Wagner Ltda. Rio de Janeiro, 1991.

[13] Petrick, A., O'Hara, B., *IEEE 802.11 Handbook: A designer's Companion*. Published by IEEE Press.

[14] Rivest, R., *RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4*. Tech note, RSA Data Security, Inc, October 2001.

[15] Santos, J. P. O., Mello, M. P., Murari, I. T.C., *Introdução à análise combinatória*. 2. ed. Campinas, SP: Editora da Unicamp, 1998.

[16] Stubblefield, A., Ioannidis, J. and Rubin, A. D., *Using the Fluhrer, Mantin and Shamir attack to break WEP*. Technical Report TD-4ZCPZZ, AT&T Labs, August 2001.

Índice