UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ramon Hugo de Souza

Criptoanálise χ^2 do Cifrador MRC6

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Daniel Santana de Freitas, Dr.

Co-Orientador: Jorge Nakahara Jr., Dr.

Florianópolis, março de 2008

Criptoanálise χ^2 do Cifrador MRC6

Ramon Hugo de Souza

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Segurança em Sistemas Computacionais e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Mário Antônio Ribeiro Dantas, Ph.D.
Coordenador do PPGCC

Banca Examinadora:

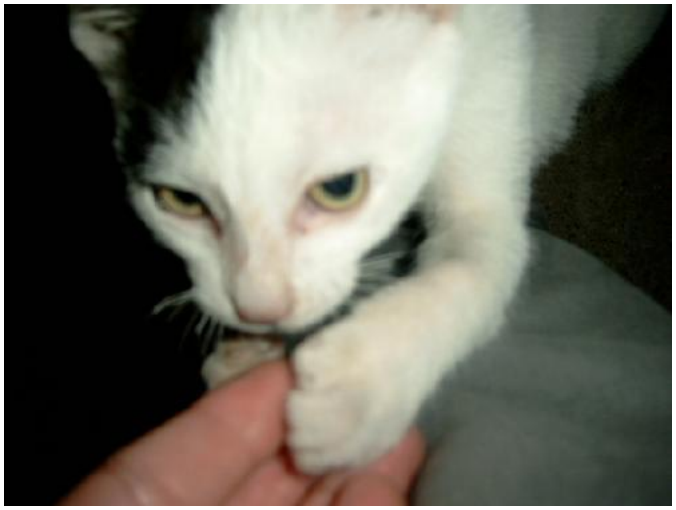
Daniel Santana de Freitas, Dr.

Joni da Silva Fraga, Ph.D.

Jorge Nakahara Jr., Dr.

Mário Antônio Ribeiro Dantas, Ph.D.

"Who's to know if your soul will fade at all
The one you sold to fool the world
You lost your self-esteem along the way"
-"Fake It" by Seether



In memorium of Mohamed

(Desaparecido em 11 de outubro de 2006)

Agradecimentos

Agradeço a todos que me ajudaram no desenvolvimento deste estudo e nos pré-requisitos fundamentais à criptanálise, em especial ao professor Daniel Santana, orientador desta dissertação e reponsável pela administração de algumas disciplinas, necessárias para o aprimoramento na área de criptanálise, que foram abertas como tópicos especiais no programa de pós-graduação em ciências da computação e ao professor Jorge Nakahara, por propôr ataques a alguns cifradores durante o período de estudo e pela sugestão de ataque ao MRC6.

Agradeço também a Vera Lúcia da secretaria por sempre prestar ajuda na resolução dos problemas administrativos necessários para apresentação da dissertação.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Nesta dissertação é feita uma primeira análise da segurança do cifrador em blocos MRC6, uma modificação do cifrador em blocos RC6 proposta por El-Fishawy *et al.* [1]. Em sua descrição, El-Fishawy *et al.* justificam o "novo" cifrador com base em diversos resultados experimentais por eles obtidos, os quais indicam que o MRC6 é *mais eficiente* do que a versão original do RC6. É um fato conhecido que todo cifrador moderno está sujeito a um delicado equilíbrio entre segurança e eficiência. No entanto, os autores do MRC6 não apresentam nenhuma análise de segurança do cifrador que estão propondo, o que sugere que eles adotam implicitamente a hipótese de que, uma vez que possui uma estrutura interna idêntica à do RC6, o MRC6 simplesmente "herda" as propriedades criptoanalíticas diferenciais e lineares do RC6. Neste trabalho, é demonstrado analiticamente que esta suposição é falsa: o novo entrelaçamento entre as palavras projetado por El-Fishawy *et al. não é suficiente* para compensar o enorme aumento no tamanho do bloco de cifragem que eles estão propondo (de 128 para 512 bits), o que acaba por produzir fraquezas *muito mais acentuadas* do que as existentes no RC6. Com base nesta análise, é desenvolvida uma variante do ataque χ^2 de Knudsen e Meier [4], de modo a levar em conta a peculiar lentidão na difusão do MRC6. São obtidos resultados da aplicação desta variante contra o MRC6, os quais demonstram claramente que, para um número de rodadas menor ou igual ao do RC6, ele pode ser quebrado com muito menos esforço do que força bruta e, como conseqüência, com muito menos esforço do que o RC6. De maneira geral, a análise dos resultados deste ataque contra o MRC6 sugere que *qualquer* versão do RC6 com mais do que 4 palavras processadas

por vez fica mais fraca do que a versão original. Observou-se ainda que, embora tendo sido especificamente concebida para atacar o MRC6, a variante do ataque χ^2 proposta no presente trabalho contribui também para melhorar o ataque χ^2 original de Knudsen e Meier contra o próprio RC6.

Abstract

This work presents the first security evaluation of the block cipher MRC6, a variant of the block cipher RC6 proposed by El-Fishawy *et al.* [1]. Several experimental results obtained by El-Fishawy *et al.* place their "new" design as the *most efficient* in the whole "family" of RC6 block ciphers, including MRC6, RC6 and another variant called ERC6. It is well known that any modern block cipher is the result of a delicate balance between security and efficiency. However, despite advertising an improvement in the throughput of RC6, the paper from El-Fishawy *et al.* did not include any cryptanalytic attack against their own proposal. This implies a strong assumption: since its internal structure is the same as RC6's, MRC6 would implicitly "inherit" RC6's differential and cryptanalytic resistance. This work shows that such an assumption cannot be valid. First, it is analytically demonstrated that the new layer of word rotations designed by El-Fishawy *et al. is not enough* to compensate for the huge increase that they are proposing in the size of the encrypting block (from 128 to 512 bits). Then, taking into account the very low diffusion in an iteration of the modifified RC6, a special variant of Knudsen and Meier's chi-square attack [4] is implemented against it. Experimental results clearly demonstrate that a chosen plaintext attack can break MRC6 with much less effort than brute force and, consequently, it is much easier to break MRC6 than RC6. Further, these results suggest that *any* variant of RC6 processing more than 4 words at a time is weaker than the original version. As an additional result, the proposed variant of the chi-square attack was used against RC6 itself, leading to an improvement over Knudsen and Meier's results.

Capítulo 1

Introdução

Esta dissertação trata da primeira análise de segurança do cifrador em blocos simétrico MRC6[1]. O MRC6 é uma modificação do cifrador em blocos RC6[5] apresentada por outros autores que não os da cifra original. O RC6, por sua vez, é uma evolução do RC5[6] e foi concebido para cumprir os requisitos do concurso iniciado no ano de 1997 para escolher o novo padrão de cifragem dos Estados Unidos da América para o início do século 21, o "Advanced Encryption Standard" (ou AES[7]). O AES deveria substituir o já comprovadamente frágil "Data Encryption Standard" (ou DES[8]), o qual vinha sendo utilizado como padrão para cifragem em blocos por mais de 20 anos. Embora não tenha vencido o concurso para o AES (quem venceu foi o cifrador Rijndael [9]), o RC6 acabou provando ser um dos 5 mais fortes candidatos a padrão norte americano para cifragem simétrica de blocos.

Conforme descrição apresentada nos capítulos 2 e 4, o cifrador RC6 possui uma estrutura muito simples, mas muito eficaz em evitar os principais ataques criptoanalíticos conhecidos até o momento da sua concepção. Em sua versão padrão, ele processa blocos de 128 bits de cada vez, produzindo 128 bits de texto cifrado. O algoritmo do RC6 envolve primitivas muito simples e eficientes, tais como ou-exclusivo, adição e multiplicação módulo 2^w e rotações de bits *dependentes dos dados*. Em uma estrutura semelhante à de Feistel, a cada rodada, todos os 128 bits de cada bloco são usados na modificação de duas das quatro palavras de 32 bits que

o compõem - os detalhes do RC6 são apresentados no capítulo 2. A estrutura do RC6 ajuda a frustrar os ataques de maior impacto até o ano 2000: a criptoanálise diferencial [10] e a criptoanálise linear [11]. Ocorre que as rotações *dependentes dos dados* fazem com que os bits sejam espalhados para posições "aleatórias" a cada rodada, tornando muito difícil acompanhar a propagação de "trilhas" lineares ou diferenciais ao longo da cifragem com o RC6. É como se uma parte do algoritmo de cifragem fosse definida somente no momento da sua utilização, dificultando o planejamento de um ataque criptoanalítico.

Mesmo assim, já em 1998, um relatório de autoria de alguns dos próprios autores do RC6 [2] mostrava que diversas relações lineares com alta probabilidade poderiam ser montadas, envolvendo os bits finais da primeira e da terceira palavras de cada bloco de entrada e os seus correspondentes após uma cifragem. Com base nestas relações, é mostrado naquele relatório que a criptoanálise linear quebraria o RC6 com esforço menor do que força bruta se ele fosse utilizado com apenas 12 das 20 rodadas recomendadas para o seu uso na cifragem de blocos de 128 bits.

A partir daí, ficou evidente que, justamente o fato de uma parte do RC6 depender dos dados, *poderia servir como recurso para atacá-lo.* O raciocínio é, basicamente, o seguinte: se uma parte do cifrador depende dos dados, um criptoanalista consegue ter *acesso indireto ao interior do algoritmo* a partir de uma cuidadosa manipulação daquilo que vai ser submetido para cifragem. Em outras palavras, em um ataque do tipo "texto em claro escolhido" (o oponente tem acesso ao resultado da cifragem de textos por ele especificados), pode-se levar o cifrador a funcionar em um estado que exponha fraquezas em sua estrutura interna. Na verdade, um ataque mais forte, com texto em claro apenas "conhecido" já seria viável diante desta característica do RC6 (apenas exigiria mais textos cifrados com a chave a ser recuperada).

Então, com base na *análise linear* feita em [2], Knudsen e Meier [4] e, de maneira indepedente, Baudron *et al.* [12], mostraram, durante o desenrolar do processo do AES (em 2000), que a submissão de muitos blocos de textos em claro com

uma certa característica especial em comum (ver capítulo 5) a versões reduzidas (com menos rodadas) do RC6 tendia a produzir textos cifrados *com correlação estatística*. Mais especificamente, eles mostraram que a submissão de um certo conjunto de blocos de texto em claro especiais a um RC6 reduzido produziria saídas cifradas cuja distribuição de freqüências seria identificada como *não-uniforme* por um teste estatístico adequado. Isto representa uma "falha" do cifrador RC6, pois todo (bom) cifrador moderno deve se comportar como um gerador pseudo-aleatório, ou seja, para qualquer que seja o conjunto de blocos na entrada, a distribuição das freqüências das saídas cifradas deve "passar como uniforme" em qualquer teste estatístico adequado.

O trabalho de Knudsen e Meier [4] é o mais completo dos dois e acabou se tornando a principal referência na área de criptoanálise estatística. Eles demonstraram experimentalmente a sua argumentação usando o teste estatístico chi-quadrado (χ^2). Essencialmente, o que eles fizeram foi usar medições experimentais de χ^2 para verificar a aderência da distribuição de freqüências de um certo subconjunto dos bits das saídas de um grande número de textos cifrados com RC6 a uma distribuição uniforme. Seus resultados acabaram se constituindo no mais forte ataque concebido contra o RC6 até o momento. Por exemplo, com base em simulações sobre versões do RC6 com até 5 rodadas, eles conseguem inferir que o teste χ^2 permitiria *distinguir*, com muito menos esforço do que força bruta, blocos de 128 bits cifrados por um RC6 com até 15 rodadas de textos produzidos por um gerador pseudo-aleatório. Mas eles foram mais além e mostraram ainda que era possível usar o teste χ^2 em um ataque mais forte, para recuperação de chaves. Após aplicar o teste χ^2 às saídas produzidas por uma certa quantidade de textos submetida a versões do RC6 de 128 bits com 2, 4 e 6 rodadas, eles inferiram que, com menos esforço do que força bruta, o teste χ^2 permitiria montar um ataque de recuperação de chaves que exigiria menos esforço do que força bruta contra um RC6 com *até 15 rodadas*. E mais ainda: para certas chaves fracas (uma em cada 2^{80}), este ataque teria sucesso até contra um RC6 com 17 rodadas.

Em resumo, além de denunciar imediatamente com base nos dados cifrados que o cifrador RC6 é o cifrador que está em uso, o que já é muito grave para

um cifrador moderno, o ataque de Knudsen e Meier permitia também recuperar a chave, com base em uma certa quantidade de textos em claro escolhidos, se a quantidade de rodadas fosse menor do que 16. Embora mesmo assim a versão nominal de um RC6 com 20 rodadas (proposta como padrão ao concurso AES) não ficasse ameaçada na prática, ficou evidente que a *combinação do uso de testes estatísticos para a detecção de fraquezas lineares com a manipulação dos dados de entrada* se constituía em uma séria ameaça, possivelmente não prevista pelos próprios projetistas do RC6.

Em 2004, El-Fishawy *et al.* [1] propuseram, na "International Conference on Electrical, Electronic and Computer Engineering - ICEEC" (um congresso patrocinado pela IEEE), uma modificação do algoritmo do RC6 e a denominaram de MRC6 (ou "Modified RC6"). Em sua versão, eles estendem o RC6 de modo a processar blocos de 512 bits, mantendo os mesmos parâmetros e primitivas da versão original do RC6. Na verdade, a única modificação que eles fazem é no *entrelaçamento* das palavras de 32 bits de texto que ocorre no final de cada rodada do RC6, como resultado de uma simples rotação de palavras. Enquanto que no RC6 esta rotação envolve 4 palavras (pois o RC6 processa 128 bits de cada vez), no MRC6 a mesma operação envolve as 16 palavras de 32 bits resultantes do processamento de 4 blocos de texto de 128 bits de cada vez. Ou seja, estruturalmente o MRC6 é *muito parecido com o RC6*, podendo ser definido (aproximadamente) como uma espécie de versão estendida do RC6 em que 4 blocos de texto de 128 bits (em um total de 512 bits) são processados em paralelo, mas só são "misturados" em conjunto ao final de cada rodada.

Os autores do MRC6 apresentam em [1] diversas evidências experimentais de que, apesar de muito simples, a modificação por eles proposta consegue aumentar a *eficiência de cifragem* do RC6. Mais do que isto, eles comparam experimentalmente a taxa de cifragem do MRC6 com as taxas do RC6, do RC5 e do ERC6 (uma outra versão modificada do RC6, descrita em [13]) e acabam concluindo que o MRC6 se constitui na versão "ótima" de todas elas, pois atinge as mais altas taxas e os menores tempos de cifragem e decifragem de blocos de texto. Eles

acabam concluindo que o MRC6 é "simples, compacto e flexível" e que tem condições de satisfazer "aos requisitos do concurso do AES" e também "às exigências dos desenvolvedores de sistemas computacionais seguros".

No entanto, apesar da modificação ao RC6 proposta por El-Fishawy *et al.* em [1] *parecer* consistente, seu trabalho não apresenta (e nem mesmo menciona) nenhuma análise criptográfica da segurança do novo algoritmo que eles estão propondo. Fica implícita no trabalho deles a idéia de que o MRC6 é um cifrador forte simplesmente porque a sua estrutura é, essencialmente, a mesma do RC6, a qual tem sido extensivamente atacada e analisada ao longo dos últimos anos.

De fato, *aparentemente* a modificação proposta por El-Fishawy *et al.* não compromete a segurança do algoritmo original do RC6, constituindo-se, inclusive, em um reforço significativo à sua resistência criptográfica. É natural supor que o acompanhamento de trilhas diferenciais e lineares fique ainda mais difícil quando é acrescentada ao algoritmo básico uma "troca de informações" envolvendo 16 palavras (de 32 bits) de texto em claro de cada vez, em vez das 4 palavras originais. Ou seja, trata-se de uma modificação que, aparentemente, tende a aumentar consideravelmente a *difusão* entre blocos do RC6, o que reforçaria, portanto, um dos dois mecanismos fundamentais da cifragem em blocos moderna. O outro mecanismo fundamental é denominado de *confusão* (ver detalhes em [3]). Conceitualmente, trata-se efetivamente de uma modificação do tipo usado para reforçar a difusão em um cifrador, mas o *modo* como esta idéia é implementada tem que ser cuidadosamente avaliado.

No projeto de cifradores em bloco, é freqüente que o mecanismo implementado para resolver uma fraqueza crie novas brechas que não constavam da versão anterior. Os bons cifradores modernos foram projetados com extremo rigor e os seus autores já costumam incluir no projeto original todas as modificações possíveis. Muitas versões são analisadas e é publicada apenas aquela que passou em exaustivos testes estatísticos e que resistiu a muitas tentativas de ataque pelos próprios autores (e pela comunidade acadêmica). No caso do MRC6, fica a dúvida: por que os próprios autores do RC6 não propuseram esta possibilidade de "extensão"

do seu algoritmo? O RC6 já é bastante parametrizado: pode-se variar, além do número de rodadas (r), os *tamanhos das palavras que compõem os blocos* (w bits) e da chave (b bytes), sendo a sua designação completa dada por RC6-r/w/b (A versão nominal é RC6-20/32/16). Não seria difícil incluir um quarto parâmetro, digamos "n", passando o RC6 a ser designado por "RC6-r/w/b/n". Este n seria a definição da *quantidade* de palavras (com w bits) que são processadas de cada vez. Neste caso, o MRC6 seria apenas um caso particular: ele seria um "RC6-r/w/b/16".

Mas o fato é que esta variação do RC6, incluindo a mistura de mais do que 4 palavras de cada vez, não é sugerida na proposta original do RC6 e também nunca houve nem sequer um comentário subseqüente dos autores a respeito desta possibilidade. Tudo isto, aliado ao fato de que os autores do MRC6 não cumpriram o mais elementar requisito de projeto de um cifrador novo, ou seja, atacar a própria criação, leva a uma natural desconfiança em relação à real força criptográfica do MRC6. Note-se que, apesar desta importante lacuna de projeto, os autores do MRC6 continuam utilizando-o como se ele fosse um cifrador de comprovada eficácia (ver [14]).

Usualmente, a pesquisa na área de criptografia de cifradores em bloco simétricos consiste em procurar falhas nos cifradores mais importantes. Os mais importantes são os cifradores que, após anos de escrutínio pela comunidade científica, não tiveram nenhuma falha significativa relatada. Ocasionalmente, a atenção se volta para variações destes cifradores, propostas com a intenção de se constituírem em "melhoramentos de projeto", como é o caso do MRC6.

A experiência têm mostrado que os melhores cifradores já foram projetados de uma maneira tão otimizada que qualquer modificação tende a produzir versões piores. Este foi o caso, por exemplo, do cifrador Akelarre [15], apresentado em 1996, o qual consistia em uma "fusão" dos cifradores IDEA e RC5. Com esta mistura, os autores pretendiam ter produzido o cifrador "ideal", aliando a altíssima taxa de cifragem do RC5 com as características matemáticas fortemente resistentes a criptoanálise que o IDEA possui. Mas o resultado foi desastroso: apesar de produzir uma saída essencialmente indistinguível de uma geração de números aleatórios, o

Akelarre continha uma falha de projeto grosseira. Com base nesta falha, em 1998, Knudsen e Rijmen [16] demonstraram que o Akelarre podia ser facilmente quebrado, mesmo considerando-se o ataque mais simples possível, ou seja, mesmo se o atacante só tivesse acesso a textos cifrados. Após esta bem sucedida criptoanálise do Akelarre, os seus autores ainda responderam com uma variante atualizada chamada de Ake98. Nesta nova versão, eles incluíram, entre outros aperfeiçoamentos, um novo bloco de difusão (muito mais consistente) e a adição de mais subchaves a cada rodada. Ainda assim, mesmo resultando de grandes reformas estruturais na cifra Akelarre, o Ake98 possuía grandes classes de chaves fracas. As chaves pertencentes a estas classes eram tão frágeis que permitiam uma criptoanálise mais rápida do que busca exaustiva com o uso de apenas 71 textos conhecidos, e isto para até 11.5 rodadas do Ake98 [17].

Nesta dissertação é feita uma primeira análise de segurança do MRC6. É apresentado e implementado um ataque extremamente eficaz, o qual comprova claramente que o MRC6 pode ser quebrado com muito menos esforço do que o próprio RC6. A conclusão final é que o aparente aumento de difusão gerado pela modificação proposta por El-Fishawy *et al.* em [1] produz fraquezas lineares muito mais acentuadas do que as existentes no RC6. De maneira geral, o ataque descrito nesta dissertação contra o MRC6 mostra que *qualquer* versão do RC6 com mais do que 4 palavras processadas por vez fica mais fraca do que a versão original quando desconsiderando-se modificações nos dados responsáveis pela saída da função que define a mistura entre os sub-blocos. Em resumo, este é um estudo que comprova a regra geral, descrita acima, que estabelece que é muito difícil modificar o projeto dos melhores cifradores atuais sem introduzir fraquezas significativas.

1.1 Objetivos

O objetivo geral desta dissertação é analisar a segurança criptográfica do cifrador simétrico em blocos MRC6, comparando a sua resistência a criptoanálise com a do cifrador que lhe serve de base, o RC6.

Especificamente, este trabalho tem os seguintes objetivos:

- detectar fraquezas lineares no MRC6 que não estejam presentes no RC6;
- usar estas fraquezas para projetar um ataque estatístico do tipo χ^2 ao MRC6;
- implementar o ataque projetado no item anterior sobre versões reduzidas (com poucas rodadas) do MRC6;
- com base nos resultados obtidos, estimar o nível de segurança do MRC6 completo (com 20 rodadas);
- comparar as conclusões obtidas com aquelas que já são conhecidas para o RC6.

1.2 Descrição

Este trabalho está organizado como segue. Os capítulos 2 até 4 discutem os fundamentos teóricos do ataque ao MRC6 que está sendo proposto e cujos resultados, mostrados no capítulo 6, constituem o escopo desta dissertação.

Inicialmente, no capítulo 2, são apresentados os detalhes dos algoritmos dos cifradores RC5, RC6 e também são discutidos alguns dos princípios que orientaram os seus projetos. Todos estes detalhes são fundamentais para a compreensão da estrutura do MRC6 e também para a análise dos resultados obtidos nos capítulos finais. Logo a seguir, é descrita a modificação de projeto do RC6 que originou o MRC6. Os autores do MRC6 vêem apenas vantagens na sua modificação, as quais são relatadas neste capítulo. O capítulo 2 mostra ainda que a precária análise de segurança feita pelos autores do MRC6, se limitando apenas a mencionar que ele possui uma estrutura algébrica muito similar ao RC6, é insuficiente e *não permite* concluir que o MRC6 resiste aos ataques convencionais com a mesma robustez que o RC6.

O capítulo 3 trata das principais ameaças à segurança do RC6 que foram consideradas no seu projeto original: as criptoanálises diferencial e linear. São as fraquezas lineares listadas neste capítulo que dão base aos ataques estatísticos contra o RC6 discutidos no capítulo 4 (incluindo o ataque χ^2).

Em seguida vêm os dois capítulos que contêm os principais resultados e conclusões originais do presente trabalho.

O capítulo 5 consiste em um análise crítica do artigo de Knudsen e Meier [4]. No decorrer desta análise, foram obtidos alguns resultados e conclusões adicionais sobre a criptoanálise χ^2 do cifrador RC6, complementares àquilo que está relatado em [4]. Estes resultados adicionais incluem uma *variante* do ataque de recuperação de chaves proposto por Knudsen e Meier que é mais eficaz do que o ataque original, embora seja mais restritiva. É exatamente com base nesta variante do ataque de Knudsen e Meier ao RC6, descrita no capítulo 5, que será montado o ataque χ^2 ao MRC6 (descrito no capítulo 6).

Finalmente, o capítulo 6 mostra que os autores do MRC6 se equivocaram ao avaliar que a sua resistência à criptoanálise era equivalente a do próprio RC6. Os resultados do capítulo 6 comprovam, mais uma vez - um equívoco semelhante já havia sido cometido pelos autores do cifrador Akelarre (ver [15] e [16]) -, que similaridade de estrutura com cifradores fortes *não significa* mesma resistência criptográfica. Mais especificamente, estes resultados mostram que, devido à difusão ser, na verdade, *muito mais lenta* no MRC6 do que no RC6, a variante do ataque χ^2 ao RC6 que é descrita no capítulo 5 possui um efeito muito maior sobre o MRC6 do que sobre o próprio RC6. O capítulo 6 mostra justamente a adaptação desta variante do ataque de Knudsen e Meier para o caso do MRC6, juntamente com os detalhes de implementação e com a análise dos resultados experimentais correspondentes.

O capítulo 7 apresenta um resumo das conclusões obtidas nesta dissertação. De maneira geral, os resultados obtidos comprovam que, apesar de mais eficiente em termos de desempenho, o MRC6 é muito mais vulnerável a ataques estatísticos (e, portanto, também a ataques lineares) do que o próprio RC6.

Capítulo 2

Descrição dos cifradores RC5, RC6 e MRC6

São cifradores com operações e definição de rodadas simples que surgiram da linha dos cifradores RC2 (Cifrador desenvolvido por patrocínio da empresa norte americana Lotus para que após avaliação da National Security Agency (NSA) pudesse ser incorporado como parte do software Lotus Notes.) e RC4 (Cifrador desenvolvido para RSA Data Security - hoje conhecida como RSA Security - extremamente utilizado na definição de protocolos como o Secure Sockets Layer (SSL) - para proteção de trafego na internet -, o Wired Equivalent Privacy (WEP) e o Wi-Fi Protected Access (WPA) - utilizados para a segurança de redes sem fio.) - cifradores que tiveram seu código mantido em sigilo até 1996, ano em que foram postados anonimamente, o RC2 no forum Usenet em janeiro e o RC4 inicialmente na lista de emails "Cypherpunks" em setembro e posteriormente no grupo de notícias sci.crypth -, estes desenvolvidos em 1987 por Ronald Lorin Rivest, e que posteriormente deram origem a versões expandidas desenvolvidas por vários outros autores da área de criptografia - em parte com ênfase no aumento do número de bits criptografados quando se tratando do desenvolvimento de equipamentos criptográficos na pesquisa de Engenharia Elétrica.

O cifrador MRC6, objeto do presente estudo, consiste em uma va-

riação do cifrador RC6, um dos cinco finalistas do concurso AES e um dos mais fortes cifradores da atualidade. O RC6, por sua vez, já consistia em uma evolução do cifrador RC5, projetada para levar em conta as exigências do AES, tais como uma margem de segurança razoável, de modo a poder continuar em uso por bastante tempo em pleno século 21. A passagem do RC5 para o RC6 foi extremamente cuidadosa e o algoritmo que resultou acabou ficando bem mais forte do que o original em relação aos principais ataques criptoanalíticos conhecidos à época do seu projeto. Grande parte dos detalhes do RC6 descritos neste capítulo fazem parte do projeto do MRC6. Por outro lado, a transição do projeto do RC5 para o RC6 é aqui descrita como uma referência em relação à análise que é feita no capítulo 6 sobre a eficácia das modificações feitas pelos projetistas do MRC6 ao RC6.

O cifrador RC6 - cifrador desenvolvido pelos laboratórios da RSA Security e definido por padrão com blocos de 128 bits - é uma evolução do cifrador RC5 - cifrador desenvolvido pelos laboratórios da RSA Security e definido por padrão com blocos de 64 bits - com uma certa tendência a ser definido como uma duplicação do RC5 com algumas modificações - como se um fosse o outro com duas instâncias em paralelo - mantendo o mesmo padrão em relação à definição dos sub-blocos, e *quase que* apenas expandindo a camada de rotação interna dos blocos, já que o mesmo sofre também uma pequena adaptação quando da rotação de sub-blocos no fim da rodada devido ao aumento do número de sub-blocos de um cifrador em relação ao outro.

De um modo geral os cifradores da linha RC - cifradores RC2, RC4, RC5 e RC6 - são cifradores com definições que buscam altos rendimentos baseados na simplicidade das operações e tentam se manter dificeis de criptoanalisar pelo fato de possuírem rotações que dependem dos dados analisados, o que dificulta até mesmo a sua definição algébrica.

Cifradores em blocos que se baseiam nas definições iniciais de Ronald Lorin Rivest funcionam com sub-blocos específicos - menores que o bloco completo - definidos como redes de feistel como a mostrada na figura 2.1, para o caso do cifrador DES, e que possuem uma camada que faz com que os blocos interajam

de maneira a se misturarem com o passar das rodadas. Ou seja, são cifradores com definições de blocos como se fossem cifradores separados que em um certo momento interagem de modo a que com um certo número de rodadas eles possam ser encarados como um bom cifrador, porém como tem sido verificado com o passar do tempo, estes cifradores são um pouco perigosos se não definido corretamente o seu número de rodadas.

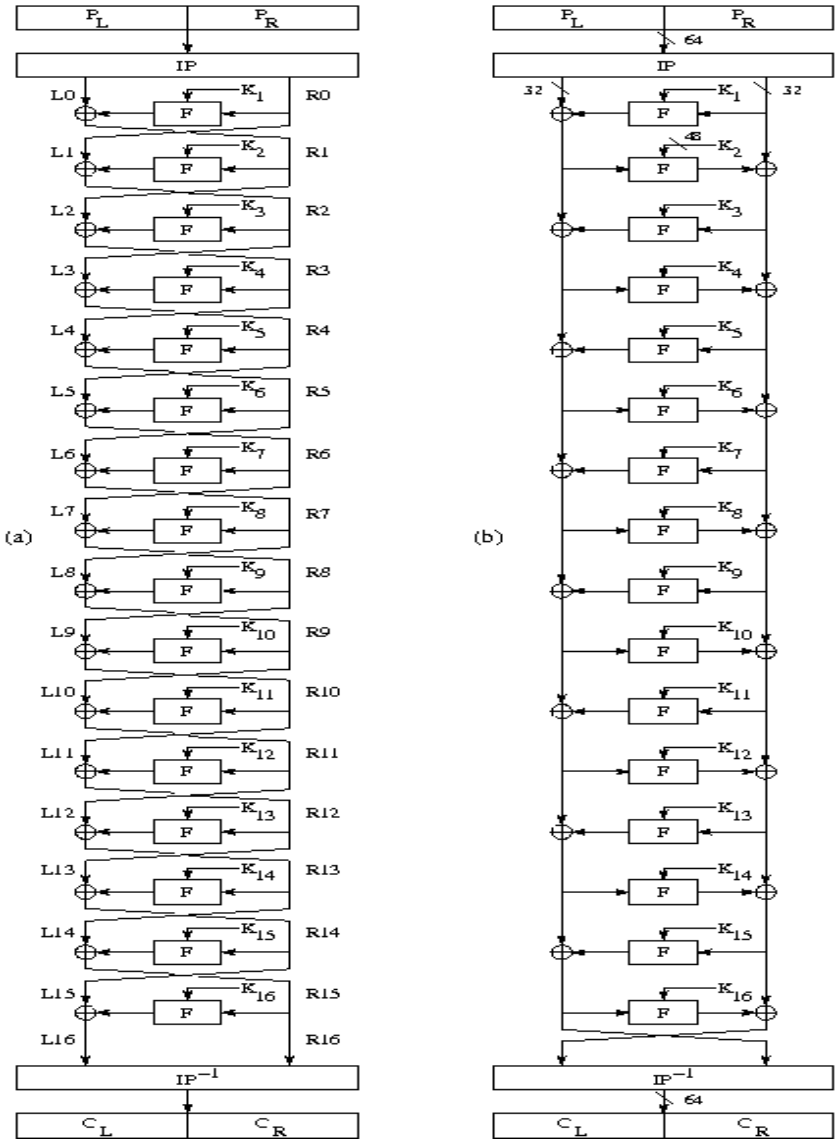


Figura 2.1: Rede de Feistel para o caso do cifrador DES.

De um modo geral, a função definida na modificação de sub-blocos

individualmente na definição de uma rede de feistel nestes cifradores está sempre relacionada com a rotação interna de bits com informação vinda de outros sub-blocos, porém de forma adaptada com a evolução de um algorítmo para outro - sendo a adaptação um ponto que aparentemente foi esquecido quando da definição do MRC6 e que exploraremos no capítulo 6.

2.1 Descrição e características do cifrador RC5

O RC5 é um cifrador de muita simplicidade definido em 1994 pelo criptógrafo Ronald Lorin Rivest [6]. Ele foi patenteado nos Estados Unidos para uso da empresa hoje conhecida como RSA Security (na época conhecida como RSA Data Security) em maio de 1997. Rivest é membro do Laboratório de Ciências da Computação e Inteligência Artificial do MIT - Massachusetts Institute of Technology - (CSAIL - Computer Science and Artificial Intelligence Laboratory), local onde também leciona. Ele é também o autor dos cifradores RC2 e RC4, além de co-autor do cifrador lançado como candidato ao AES, o RC6.

Este cifrador é definido com blocos de 32 bits (desenvolvido com propósito de avaliação e realização de experimentos), 64 bits (desenvolvido com a idéia principal de substituição de dados cifrados com o DES), ou 128 bits. Sua chave tem possibilidade de ir de 0 a 2040 bits e possibilita que seja definido entre 0 e 255 rodadas, sendo a sugestão inicial dada por seu criador a utilização de blocos de 64 bits, chave de 128 bits e 12 rodadas. O cifrador foi desenvolvido com muita flexibilidade nos campos de segurança e eficiência, e a sugestão dada quando de seu desenvolvimento está relacionada ao aspecto de segurança e velocidade, no contexto da substituição do então padrão de cifragem DES.

Seu autor o define como sendo um cifrador com três rotinas, sendo elas:

- a rotina de *expansão de chave*, em que a chave dada pelo usuário é expandida para uma tabela de tamanho dependente do número de rodadas que foi definido, a qual é utilizada no processo de cifragem e decifragem;

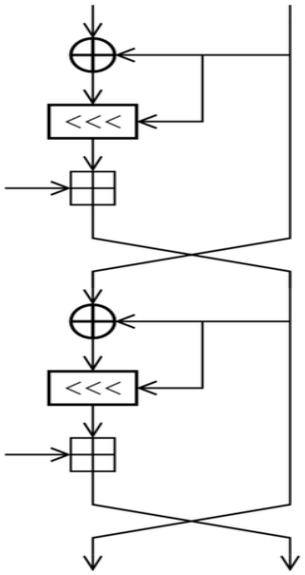


Figura 2.2: Uma rodada do cifrador em blocos RC5

- a rotina de *encriptação*, que consiste de três operações primitivas: adição de inteiros módulo 2^w (denotado por "+"), ou-exclusivo bit a bit (denotado por $\oplus$), e rotação variável (denotada por $x <<< y$);

- a rotina de *decriptação*, que desfaz a encriptação.

Note-se que o cifrador RC5 chama a atenção à criptoanálise desde a época de sua criação devido à suas rotações dependentes de dados, as quais continuaram por legado presentes em suas "próximas versões" (como o RC6), bem como em outros cifradores criados tomando este como ponto de partida.

Por outro lado o uso de rotações dependentes de dados e a combinação de diferentes operações parecem prover a segurança do RC5, sendo que esta rotação dependente de dados é o que realmente o mantém seguro quanto aos ataques linear e diferencial.

A base deste algorítmo são as estruturas de Feistel, como pode ser verificado na figura 2.2. É um cifrador muito simples que pode ser facilmente definido em poucas linhas de código (como será demonstrado na seqüência).

Quanto à segurança do cifrador, vários estudos citados na página

da RSA Laboratories [18], tais como [19], [20], [21] e [22], demonstraram como esta estrutura geral definida para o cifrador contribui para sua segurança, de modo que a estrutura se manteve no sucessor RC6.

Como descrito por seu autor, o RC5 é um algoritmo parametrizado como RC5-w/r/b, com seus parametros definidos do seguinte modo:

- w - O tamanho da palavra definido em bits. Pode possuir os valores de 16, 32 ou 64 bits, tendo como padrão o valor de 32 bits. E como o RC5 trabalha com duas palavras, os blocos de texto em claro e texto cifrado possuem tamanho de 2w bits.
- r - Número de rodadas que pode ser definido entre 0 e 255.
- b - Número de bytes da chave K, com possibilidade de valores entre 0 e 255, ou seja até 2040 bits.

2.1.1 Geração de chaves por expansão

O algoritmo de geração de chaves por expansão, assim como definido em [6], gera um array S de tamanho t = 2(r+1), composto por palavras binárias derivadas da chave K definida pelo usuário. O núcleo desta geração se baseia em duas constantes e no uso de três passos algorítmicos simples. As constantes são definidas para um w arbitrario como sendo:

$$\begin{aligned} P_w &= Impar((e-2) \times 2^w) \\ Q_w &= Impar((\phi-1) \times 2^w) \end{aligned}$$

aonde:

$$\begin{aligned} e &= 2.718281828459045235... \text{ (base dos logaritmos naturais)} \\ \phi &= 1.618033988749894848... \text{ (razão de ouro)} \end{aligned}$$

e onde $Impar(x)$ é o inteiro ímpar mais próximo de x, arredondado pra cima caso x seja um inteiro par.

O primeiro passo algorítmico é copiar a chave secreta $K[0, ..., b-1]$ em um array $L[0, ..., c-1]$ em que $c = [b/u]$ palavras, e que $u = w/8$ é o número de bytes por palavra. Sendo carregados os valores de K em L como sendo u bytes. Os valores não preenchidos em L são definidos neste ponto como sendo zero. No caso em que $b = c = 0$, c é resetado para 1 e $L[0]$ para zero.

O segundo passo algorítmico é o de inicilizar o array S como sendo um padrão de bits particular pseudo-aleatório e independente de chave usando uma progressão aritmética modulo 2^w determinada pelas constantes P_w e Q_w. De modo que já que Q_w é ímpar, então a progressão aritmética tem período igual a 2^w.

$S[0] = P_w$
de i=1 até t-1 faça:

$$S[i] = S[i-1] + Q_w$$

E por fim, o terceiro passo algorítmico é a mistura da chave com três passagens pelos arrays S e L. De um modo mais específico, devido à potencial diferença de tamanhos de S e L, o array maior vai ser processsado três vezes, e o outro array pode ser manuseado mais vezes.

$i = j = 0$
$A = B = 0$
faça 3*max(t,c) vezes:

$$A = S[i] = (S[i] + A + B) <<< 3$$
$$B = L[j] = (L[j] + A + B) <<< (A + B)$$
$$i = (i + 1) mod(t)$$
$$j = (j + 1) mod(c)$$

Note que, trabalhando algebricamente com definições em GF(2), não teremos como definir exatamente qual bit vai para qual posição, de modo que não teremos na definição de cada bit seu "número de posição" com operações baseadas em valores de bits anteriores. Porém, no fim os bits de saída estarão baseados somente nos valores dos bits originais definidos pelo usuário, ou seja, serão somas com até 64

variáveis no modo sugerido por seus autores. Estas somas não são de fácil obtenção, devido às adições modulo 2^w, de modo que ficaremos presos a bem mais variáveis quando do tratamento algébrico.

2.1.2 Cifragem

A cifragem opera sobre dois registradores de w bits A e B, fornecendo a saída também em A e B. O algoritmo é dado pelo pseudo-código a seguir:

Calcular S[2r+2] subchaves
$A = A + S[0]$
$B = B + S[1]$
de i=1 até r faça:

$$A = ((A \oplus B) <<< B) + S[2i]$$
$$B = ((B \oplus A) <<< A) + S[2i + 1]$$

2.1.3 Decifragem

Quando da decifragem, assim como na cifragem, os dados são fornecidos em dois registradores de tamanho de w bits A e B, e a saída é dada também em A e B. O algoritmo é dado pelo pseudo-código a seguir:

Calcular S[2r+2] subchaves
de i=r até1 faça:

$$B = ((B - S[2i + 1]) >>> A) \oplus A$$
$$A = ((A - S[2i]) >>> B) \oplus B$$

$A = A - S[0]$
$B = B - S[1]$

2.2 Descrição e características do cifrador RC6

Assim como o RC5, o RC6 é um cifrador de chaves simétricas de muita simplicidade. Ele foi desenvolvido por Ronald Lorin Rivest em conjunto com

Matt Robshaw, Ray Sidney, e Yiqun Lisa Yin para participar da competição ao AES do NIST - National Institute of Standards and Technology, agência de tecnologia do governo norte americano -, tendo sido publicado em agosto de 1998 como candidato ao AES em [5]. Ele foi desenvolvido com base no RC5 já com o propósito de ser um cifrador proprietário da RSA Security - a não ser que viesse a se tornar o AES.

É um cifrador definido para trabalhar com blocos de 128 bits e com a possibilidade de chaves de 128, 192 e 256 bits - isso no modelo candidato ao AES, já que assim como o RC5 ele pode possuir chave de até 2040 bits, ou seja, como seu antecessor também possibilita que o tamanho da palavra seja utilizado com outros valores. É também definido como um cifrador com 20 rodadas.

Em estrutura, o RC6 é um cifrador muito parecido com seu predecessor RC5, utilizando também rotações baseadas em dados. Ele quase pode ser encarado como a utilização de dois RC5 em paralelo. É claro que esta modificação leva em conta que sub-blocos de 32 bits seriam mais eficazes na época da sua concepção, já que o RC5 possuia 64 bits e o RC6 precisaria de 128 bits para ser submetido ao AES - ou seja, foram mantidos os sub-blocos de tamanho 32 bits.

Em relação ao seu predecessor, o RC6 possui duas características novas: a inclusão da *multiplicação de inteiros* e o uso de quatro b/4-bit registradores em vez de dois b/2-bit registradores como no RC5 (b é o tamanho do bloco). Esta última característica confirma que o RC6 tende realmente a ser dois RC5 "em paralelo". A multiplicação inteira é utilizada para acelerar a difusão, de modo a permitir a utilização de menos rodadas e então poder-se aumentar a "velocidade" do cifrador.

É claro que uma simples definição do RC6 como "dois RC5 em paralelo", apenas com um ajuste na rotação final da rodada, seria insuficiente e poderia gerar uma necessidade de aumento de rodadas muito grande. Por esta razão, o RC6 já leva em conta a utilização de novas operações, de modo a evitar o aumento excessivo de rodadas e melhorar sua performance de difusão. Como pode ser visto na figura 2.3, ele possui uma maior interação envolvendo os 4 sub-blocos de formação do cifrador, de modo a garantir que os mesmos se misturem mais rapidamente, devido à dependência gerada com o passar das rodadas.

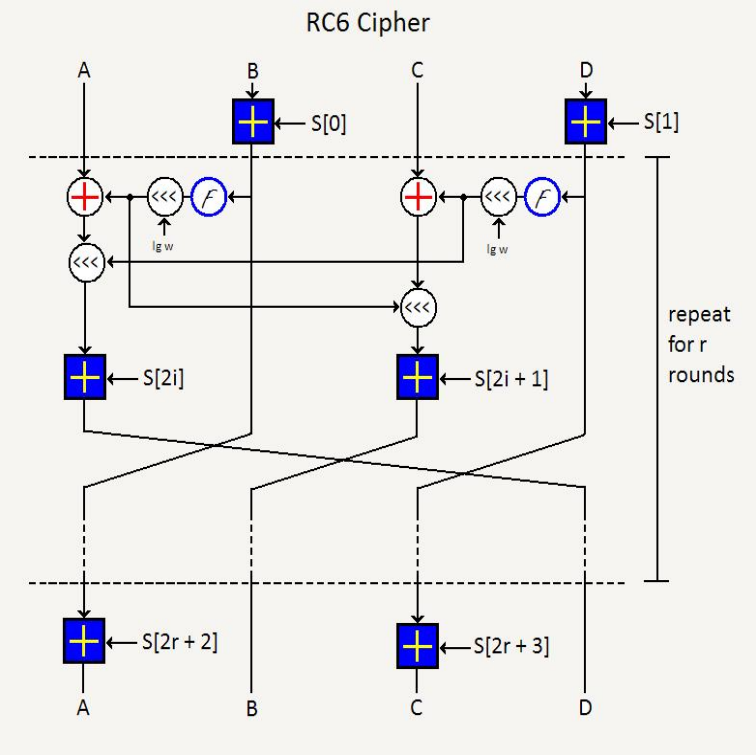


Figura 2.3: Uma rodada do cifrador em blocos RC6.

Note que a utilização de rotações no topo da figura 2.3 que fazem com que o bloco A rode devido ao D e que o bloco C rode devido ao B aumenta muito mais a interação do que dois RC5 em paralelo com uma rotação de sub-blocos modificada no final.

Como descrito por seus autores, o RC6 é um algoritmo parametrizado como RC6-w/r/b, com seus parametros definidos do seguinte modo:

- w - O tamanho da palavra definido em bits. Pode possuir os valores de 16, 32 ou 64 bits, tendo como padrão o valor de 32 bits. E como o RC6 trabalha com quatro palavras, os blocos de texto em claro e texto cifrado possuem tamanho de 4w bits.

- r - Número de rodadas que pode ser definido entre 0 e 255.

- b - Número de bytes da chave K, com possibilidade de valores entre 0 e 255,

ou seja até 2040 bits.

2.2.1 Geração de chaves por expansão

A geração de chaves do cifrador RC6-w/r/b é praticamente a mesma que a do RC5-w/r/b, de modo que a única diferença entre eles é que no caso do RC6-w/r/b são derivadas mais palavras a partir da chave original para uso na encriptação e decriptação.

O usuário fornece uma chava K de b bytes, que pode ter tamanho de 0 a 255 bytes, então são derivadas t=2(r+2) palavras de w bits cada e armazenadas então num array S[t], que é utilizado tanto na encriptação quanto na decriptação. O valor de t muda em relação ao RC5 pelo fato do RC6 utilizar o "key-withening" de forma mais explícita.

Note que as rodadas do RC6 podem ser encaradas como semi-rodadas vistas no RC5, e então pode-se manter o número de bits de chave de acordo com o RC5, já que a chave é adicionada somente nos blocos A e C pelas rodadas como pode ser observado na figura 2.3.

$S[0] = P_w$
De i=1 até t-1 faça
$S[i] = S[i-1] + Q_w$
i=j=0
A=B=0
faça 3*max(t,c)
$A = S[i] = (S[i] + A + B) <<< 3$
$B = L[j] = (L[j] + A + B) <<< (A + B)$
$i = (i + 1) mod(t)$
$j = (j + 1) mod(c)$

Sendo o algorítmo de geração de chaves o mesmo que o utilizado pelo RC5, apenas mudando o valor de t.

2.2.2 Cifragem

Quando da cifragem assumimos que são dados quatro registradores de tamanho de w bits A, B, C e D, e que a saída é dada também em A, B, C e D. O algoritmo é dado pelo pseudo-código a seguir:

Calcular S[2r+4] subchaves
$B = B + S[0]$
$D = D + S[1]$
de i=1 atér faça:

$t = (B \times (2B + 1) <<< lg(w)$
$u = (D \times (2D + 1) <<< lg(w)$
$A = ((A \oplus t) <<< u) + S[2i]$
$C = ((C \oplus u) <<< t) + S[2i + 1]$
$(A, B, C, D) = (B, C, D, A)$

$A = A + S[2r + 2]$
$C = C + S[2r + 3]$

2.2.3 Decifragem

Quando da decifragem, assim como na cifragem assumimos que são dados quatro registradores de tamanho de w bits A, B, C e D, e que a saída é dada também em A, B, C e D. O algoritmo é dado pelo pseudo-código a seguir:

Calcular S[2r+4] subchaves
$C = C - S[2r + 3]$
$A = A - S[2r + 2]$
de i=r até1 faça:

$(A, B, C, D) = (D, A, B, C)$
$u = (D \times (2D + 1) <<< lg(w)$
$t = (B \times (2B + 1) <<< lg(w)$
$C = ((C - S[2i + 1]) >>> t) \oplus u$

$$A = ((A - S[2i]) >>> u) \oplus t$$

$$D = D - S[1]$$

$$B = B - S[0]$$

2.3 Princípios de projeto dos cifradores RC5 e RC6

De um modo geral são cifradores baseados em sub-blocos de um tamanho de bits específico, e que seja pelo menos metade do valor total previamente definido ao cifrador e normalmente do tamanho padrão lidado por processadores da época em que foram desenvolvidos, que inicialmente eram muito parecidos com uma rede de feistel. Em seguida, passaram a considerar uma adaptação para estas redes com mais do que apenas dois blocos como na sua definição geral. Devido à análise de outros cifradores que tomam o RC6 como base, e mesmo olhando para a evolução do RC5 para o RC6, podemos dizer que o aumento do número destes sub-blocos sempre precisa ser compensado com o aumento de um número mínimo de rodadas, já que a adaptação de uma simples rotação no final desta rede de Feistel um pouco modificada, e a função relativa a rotação dependente de outros sub-blocos de dados aparentemente não interagem com força rapidamente envolvendo todos os blocos, em especial se não houver uma boa consideração sobre a origem dos dados que definem a função de rotação dos sub-blocos.

Baseando-se nestas observações podemos perceber o porque de o número de rodadas definidas para o RC6 ser 20 enquanto o número de rodadas do RC5 foi inicialmente proposto como sendo 12, levando-se em consideração as modificações acrescentadas ao RC6, já que também sabemos que quando encarado como o RC5 o RC6 funciona como se tivesse na verdade 10 rodadas. E desta maneira também sabemos que um grande aumento de sub-blocos teria que ser compensado com o número de rodadas e uma redefinição do que define a rotação interna dos sub-blocos, o que dificulta o desenvolvimento de cifradores com esta base que possam

ser criados de maneira a serem mais eficientes e seguros de mesmo modo como o RC6, por exemplo - devendo também se levar em consideração que seria necessário trabalhar com novas operações, que poderiam prejudicar a velocidade de cifragem.

Estes cifradores foram projetados com a intuição de que deveriam ter altas taxas de cifragem baseados em operações simples e em blocos com número de bits de acordo com o número processado pelo processador de tamanho padrão em que era utilizado na epoca de sua publicação inical, ou seja 32 bits. O RC5 parte da extrema simplicidade envolvendo apenas rotação de bits dentro dos sub-blocos, somas baseadas em ou-exclusivo envolvendo mais de um um sub-bloco, e somas com chave dentro de um corpo de tamanho 2^w. O RC6 distribui as 24 sub-rodadas do RC5 em rodadas e passa a utilizar apenas 20 rodadas, acrescentando apenas operações de multiplicação dentro de um corpo de tamanho 2^w quando da definição da modificação a blocos que serão somados e quando da rotação do resultado após esta soma. Ou seja, é apenas uma evolução para garantir que o algoritmo novo com padrão de bits 128 e não mais apenas 64 mantivesse a alta taxa de cifragem do seu predecessor em um tempo ótimo.

2.3.1 Design e motivação da evolução

Assim como definido em [5] a evolução do RC5 para o RC6 de modo a suplantar as necessidades para possivel designação como AES toma como prioridades em primeiro a segurança, em segundo a simplicidade e em terceiro uma boa performance.

Quanto a segurança e simplicidade partimos em especial da simplicidade do cifrador RC5 como um objeto de pesquisa atrativo, sendo importante para a facilidade de sua análise na comunidade cientifica, o que permite a verificação de sua segurança mais facilmente. De modo que o RC6 sendo desenvolvido a partir do RC5 além de prover uma maior segurança em um cifrador que já chamava a atenção pela sua simplicidade, mantém em parte também esta mesma simplicidade.

Os passos do design do RC6 a partir do RC5 como apresentados

em [5] são dados por:

1. Iniciando com um loop de meia rodada basica do RC5:

   de i = 1 até r faça
   $A = ((A \oplus B) <<< B) + S[i]$
   (A,B)=(B,A)

2. Rodando duas copias do RC5 em paralelo: uma nos registradores A,B e outra nos registradores C,D:

   de i = 1 até r faça
   $A = ((A \oplus B) <<< B) + S[2i]$
   $C = ((C \oplus D) <<< D) + S[2i + 1]$
   (A,B)=(B,A)
   (C,D)=(D,C)

3. No estagio de troca, em vez de trocar A por B e C por D, permutar os registradores como (A,B,C,D)=(B,C,D,A), e então AB é misturado com CD. Neste estágio o looping interno fica assim:

   de i = 1 até r faça
   $A = ((A \oplus B) <<< B) + S[2i]$
   $C = ((C \oplus D) <<< D) + S[2i + 1]$
   (A,B,C,D)=(B,C,D,A)

4. Misturar as computações de AB com CD pela troca onde é definida a rotação nas computações:

   de i = 1 até r faça
   $A = ((A \oplus B) <<< D) + S[2i]$
   $C = ((C \oplus D) <<< B) + S[2i + 1]$
   (A,B,C,D)=(B,C,D,A)

5. Em vez de utilizar B e D diretamente quando da definição da rotação e da soma, utiliza-se uma versão transformada destes registradores. Os ganhos de segurança são que as rotações dependentes de dados que serão derivadas da saída desta transformação deve depender de todos os bits da palavra de entrada e esta transformação deve prover uma boa mistura dentro da palavra. A escolha da transformação para o RC6 é dada pela função $f(x) = x(2x + 1)(mod(2^w))$ acompanhada pela rotação por cinco posições. Esta transformação aparenta cumprir com a segurança esperada enquanto tirando vantagem de primitivas simples que são eficientemente implementadas nos processadores modernos. Note que f(x) é um a um módulo 2^w e que os bits de maior ordem de f(x), que determinam a rotação, dependem fortemente de todos os bits de x. Isso nos dá:

   de i = 1 até r faça
   $t = (B \times (2B + 1)) <<< 5$
   $u = (D \times (2D + 1)) <<< 5$
   $A = ((A \oplus t) <<< u) + S[2i]$
   $C = ((C \oplus u) <<< t) + S[2i + 1]$
   (A,B,C,D)=(B,C,D,A)

6. No início e no fim das r rodadas são adicionados os passos de "pre-whitening" e "post-whitening". Sem estes passos o texto em claro revela parte da entrada para a primeira rodada de encriptação e o texto cifrado revela parte da entrada para a última rodada da encriptação. Estas adições ajudam nestes quesitos e deixam o RC6 como:

   B=B+S[0]
   D=D+S[1]
   de i = 1 até r faça
   $t = (B \times (2B + 1)) <<< 5$
   $u = (D \times (2D + 1)) <<< 5$

$A = ((A \oplus t) <<< u) + S[2i]$

$C = ((C \oplus u) <<< t) + S[2i + 1]$

(A,B,C,D)=(B,C,D,A)

A=A+S[2r+2]

C=C+S[2r+3]

Apesar desta parecer uma evolução lógica os autores do RC6 dizem ter verificado varias outras alternativas e então julgado esta como sendo a que melhor mantinha os quesitos de segurança, simplicidade e performance. Eles ainda apontam que se fossem manter o RC6 com apenas dois registradores poderiam manter o seguinte código:

B=B+S[0]

de i = 1 até r faça

$t = (B \times (2B + 1)) <<< 5$

$A = ((A \oplus t) <<< t) + S[2i]$

(A,B)=(B,A)

A=A+S[2r+1]

Quanto a boa performance para um dado nível de segurança tomando o RC5 como base de partida existe um grande conhecimento quanto à resistência da cifra quanto aos ataques mais conhecidos até o momento de sua concepção. Enquanto as tecnicas mais recentes demonstram que o RC5-32/12/b pode não ser um cifrador que mantenha-se seguro em termos de longo prazo, estes ataques não provêem uma forma prática real de se atacar ele com uma versão de 16 rodadas.

A maioria dos resultados criptoanalíticos encontrados sobre o RC5 aparentam estar ligados a avalanche relativamente lenta de mudança entre as rodadas. A adição inteira ajuda a prover uma razoável mudança devido ao efeito do "vai um", mas as mais dramáticas mudanças se dão quando duas rotações diferentes são utilizadas num ponto similir de encriptação de dois textos em claro relacionados. Tipicamente um atacante pode ter algum controle sobre a evolução das diferenças

rodada a rodada, em versões do RC5 com menos rodadas, o que pode permitir a montagem de um ataque.

As mudanças incrementais do RC6 em relação ao RC5 já estão então esboçadas. Duas significantes mudanças são a introdução da função quadrática $B \times (2B + 1)$, e respectiva $D \times (2D + 1)$, e a rotação fixa de cinco bits.

A função quadrática provê uma taxa de difusão mais rápida de modo a aumentar as chances de que simples diferenciais irão se misturar mais cedo do que ocorre com o RC5. Os valores quadráticos transformados de B e D são usados no lugar de B e de D para modificar os registradores A e C, aumentando a não linearidade sem perder entropia (já que a transformação é uma permutação). A rotação fixada em cinco bits tem um simples e importante papel complicando tanto a análise linear quanto a diferencial.

2.4 Descrição e características do cifrador MRC6

Talvez a característica mais importante a ser considerada em um cifrador em blocos moderno, depois da sua segurança criptográfica, seja o seu desempenho. E é justamente este o ponto que parece ter levado os autores do MRC6 a terem desenvolvido uma modificação de um cifrador com a robustez do RC6.

O cifrador MRC6 [1] pretende ser uma melhoria do cifrador RC6 [5], um dos cinco melhores candidatos a padrão norte americano de cifragem em bloco, segundo o concurso AES [7]. Na verdade, o cifrador MRC6 consiste em uma modificação estrutural bastante simples no projeto do RC6, nas rotações de final de rodada, de maneira a permitir o processamento de blocos de 512 bits de cada vez. Blocos assim tão grandes não são usuais nos cifradores modernos e o máximo que é considerado em aplicações comuns é que a cifragem ocorra sobre blocos de 256 bits. No entanto, os autores do MRC6 afirmam que esta modificação proporciona um ganho significativo em desempenho, "sem perda em termos de segurança", o que justificaria a sua classificação como o cifrador de desempemnho "ótimo" na "família" de cifradores do tipo do RC6. Uma outra possível vantagem do uso de um cifrador

que opere sobre blocos tão grandes poderia ser a sua utilidade na construção de funções de compressão para funções hash grandes o suficiente para evitar os ataques mais recentes, tais como a função SHA-512 [23].

O MRC6 segue a mesma idéia de expansão do RC5 para o RC6, só que agora sendo uma expansão sobre o RC6. Ele pode ser descrito quase que como 4 RC6 em paralelo. Porém, devido ao grande aumento do número de sub-blocos de 32 bits - diferente do RC6 que é quase apenas como dois RC5 com mudanças significantes na rotação interna de bits - aparentemente a simples mudança na rotação de sub-blocos ao final da rodada não parece uma boa ideia - possibilitando os ataques apresentados no capítulo 6 - se não houver um certo aumento no número de rodadas - assim como foi feita na expansão do RC5 para o RC6, só que neste caso houve também a introdução de operações e mudança na rotação interna dos sub-blocos de 32 bits além do "aumento" de rodadas. No caso do MRC6, para se manter a simplicidade da mudança somente na rotação dos blocos ao final da rodada, devido ao grande aumento do número de sub-blocos de 32 bits, os quais são interrelacionados de maneira forte apenas 4 a 4, o número de rodadas deveria ser maior do que as 20 definidas para o RC6, ou assim como na expansão do RC5 para o RC6 devia-se repensar a entrada de dados responsável pela rotação de cada sub-bloco interno, de modo a não funcionar realmente apenas quase que como 4 RC6 em paralelo.

O que de certa forma é um ponto bem forte a se perceber quando se passa a uma análise do MRC6, já que ele somente "estica" o ponto de rotação não modificando a relação entre os sub-blocos, de modo a funcionar quase como 4 RC6 em paralelo com uma leve iteração por rodada, o que pode obrigar que o mesmo possua um número um tanto elevado de rodadas para parecer um cifrador tão seguro quanto seus predecessores.

2.4.1 Geração de chaves por expansão

A geração de subchaves é praticamente a mesmo do RC6 só que para este algorítmo são necessárias mais chaves sendo t=(8r+16) armazenadas então em

S[0,...,t-1]. O algorítmo como descrito em [1] é apresentado abaixo, considerando-se a chave pré carregada de b bytes na palavra c, o array L[0,...,c-1] e número r de rodadas, e a saída como sendo chaves de rodadas de w bits em S[0,...,8r+15].

Procedure:

$S[0] = P_w$

De i=1 até $\boldsymbol{t-1}$ faça

$S[i] = S[i-1] + Q_w$

i=j=0

A=B=0

faça 3*max(t,c)

$A = S[i] = (S[i] + A + B) <<< 3$

$B = L[j] = (L[j] + A + B) <<< (A + B)$

$i = (i + 1) mod(t)$

$j = (j + 1) mod(c)$

Onde observamos um aparente erro de digitação no artigo [1] em que define "De i=1 até 2r+3 faça" e consideramos (t-1) no lugar de (2r+3), afinal 2r+3 é na verdade o t-1 do RC6 e não o esperado para o MRC6.

2.4.2 Cifragem

Este cifrador funciona exatamente como 4 RC6 em paralelo com a modificação quando da rotação para a esquerda dos blocos no fim da rodada quando ele então considera o cifrador com 16 sub-palavras e faz a rotação das mesmas como podemos verificar na figura 2.4. O algorítmo como descrito em [1] é apresentado abaixo, considerando-se a entrada como sendo 16 palavras de w-bits definidas como Ri com i de 1 a 16, número r de rodadas, e chaves de rodadas armazenadas em S[0,...,8r+15], e a saída como sendo os registradores Ri com i de 1 a 16.

Procedure:

R2=R2+S[0], R4=R4+S[1]

R6=R6+S[2], R8=R8+S[3]

R10=R10+S[4], R12=R12+S[5]

R14=R14+S[6], R16=R16+S[7]

De i=1 até r faça

$k = (R2 * 2R2 + 1) <<< lg(w)$

$l = (R4 * (2R4 + 1)) <<< lg(w)$

$m = (R6 * (2R6 + 1)) <<< lg(w)$

$n = (R8 * (2R8 + 1)) <<< lg(w)$

$t = (R10 * (2R10 + 1)) <<< lg(w)$

$u = (R12 * (2R12 + 1)) <<< lg(w)$

$v = (R14 * (2R14 + 1)) <<< lg(w)$

$z = (R16 * (2R16 + 1)) <<< lg(w)$

$R1 = ((R1 \oplus k) <<< l) + S[8i]$

$R3 = ((R3 \oplus l) <<< k) + S[8i + 1]$

$R5 = ((R5 \oplus m) <<< n) + S[8i + 2]$

$R7 = ((R7 \oplus n) <<< m) + S[8i + 3]$

$R9 = ((R9 \oplus t) <<< u) + S[8i + 4]$

$R11 = ((R11 \oplus u) <<< t) + S[8i + 5]$

$R13 = ((R13 \oplus v) <<< z) + S[8i + 6]$

$R15 = ((R15 \oplus z) <<< v) + S[8i + 7]$

$(R1, R2, ..., R15, R16) = (R2, R3, ..., R16, R1)$

R1=R1+S[8r+8], R3=R3+S[8r+9]

R5=R5+S[8r+10], R7=R7+S[8r+11]

R9=R9+S[8r+12], R11=R11+S[8r+13]

R13=R13+S[8r+14], R15=R15+S[8r+15]

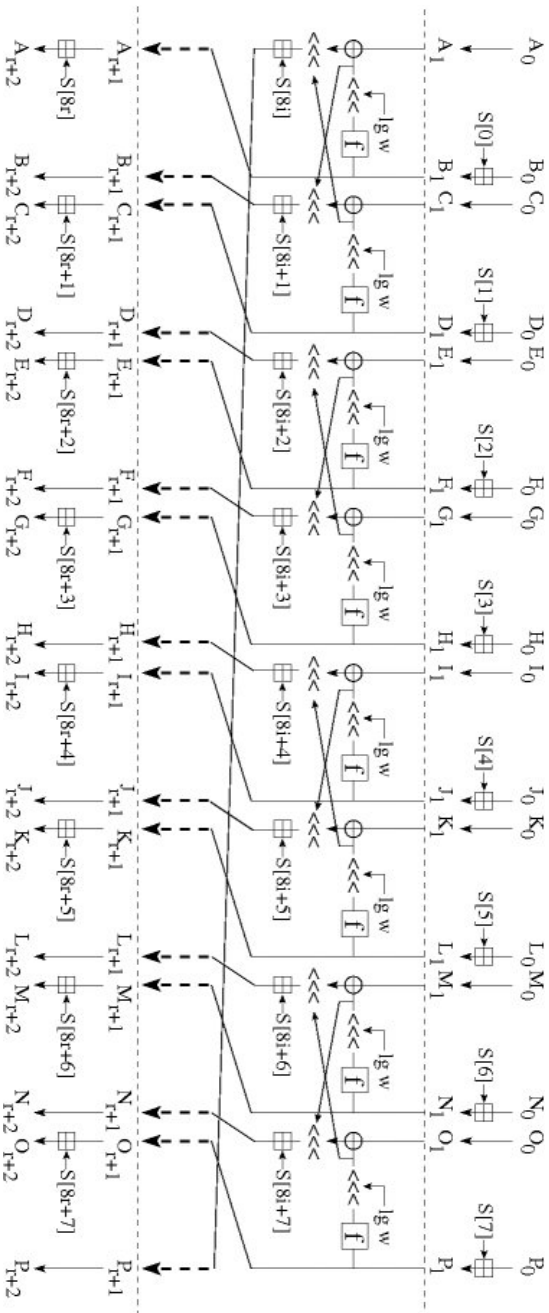


Figura 2.4: Uma rodada do cifrador em blocos MRC6.

2.4.3 Decifragem

Para decifrar ele trabalha com a adaptação de 4 RC6 em paralelo com a rotação envolvendo as 16 sub-palavras, sendo também muito similar a decifragem de 4 RC6. O algorítmo como descrito em [1] é apresentado abaixo, considerando-se a entrada como sendo 16 palavras de w-bits definidas como Ri com i de 1 a 16, número r de rodadas, e chaves de rodadas armazenadas em S[0,...,8r+15], e a saída como sendo os registradores Ri com i de 1 a 16.

Procedure:

R15=R15-S[8r+15], R13=R13-S[8r+14]

R11=R11-S[8r+13], R9=R9-S[8r+12]

R7=R7-S[8r+11], R5=R5-S[8r+10]

R3=R3-S[8r+9], R1=R1+S[8r+8]

De i=r até 1 faça

$(R1, R2, ..., R15, R16) = (R16, R1, ..., R14, R15)$

$z = (R16 * 2R16 + 1) <<< lg(w)$

$v = (R14 * (2R14 + 1)) <<< lg(w)$

$u = (R12 * (2R12 + 1)) <<< lg(w)$

$t = (R10 * (2R10 + 1)) <<< lg(w)$

$n = (R8 * (2R8 + 1)) <<< lg(w)$

$m = (R6 * (2R6 + 1)) <<< lg(w)$

$l = (R4 * (2R4 + 1)) <<< lg(w)$

$k = (R2 * (2R2 + 1)) <<< lg(w)$

$R15 = ((R15 - S[8i + 7]) >>> v) \oplus z$

$R13 = ((R13 - S[8i + 6]) >>> z) \oplus v$

$R11 = ((R11 - S[8i + 5]) >>> t) \oplus u$

$R9 = ((R9 - S[8i + 4]) >>> u) \oplus t$

$R7 = ((R7 - S[8i + 3]) >>> m) \oplus n$

$R5 = ((R5 - S[8i + 2]) >>> n) \oplus m$

$R3 = ((R3 - S[8i + 1]) >>> k) \oplus l$

$$R1 = ((R1 - S[8i]) >>> l) \oplus k$$

R16=R16-S[7], R14=R14-S[6]

R12=R12-S[5], R10=R10-S[4]

R8=R8-S[3], R6=R6-S[2]

R4=R4-S[1], R2=R2-S[0]

2.4.4 Desempenho e eficiência

Para as verificações de desempenho e eficiência do MRC6 é apresentado além dos dados para o RC5 e o RC6 também os dados para o ERC6 para uma análise comparativa com um cifrador de maior semelhança com este. Estes dados de [1] tomam como base testes realizados em programas compilados no Borland C++ em um Pentium III com clock de 800 Mhz e com 128 MB de Ram sobre o sistema Windows 98.

	4M	8M	12M	16M	20M	24M
RC5	0.384	0.775	1.326	1.569	1.935	2.320
RC6	0.281	0.665	0.852	1.119	1.485	1.644
ERC6	0.231	0.515	0.695	0.970	1.165	1.465
MRC6	0.201	0.417	0.618	0.865	1.030	1.240

Tabela 2.1: Dados de tempo de encriptação em segundos do RC5, RC6, ERC6 e MRC6[1]

Assim como apresentado em [1] na tabela 2.1 apresentamos os dados de tempo de encriptação para o RC5, o RC6, ERC6 e para o MRC6 com os parâmetros w=32, r=20 e b=16 sobre diversos tamanhos de blocos de dados.

Considerando os mesmo valores de w,r e b definidos anteriormente temos na tabela 2.2 a vazão de encriptação e decriptação para o RC5, RC6, ERC6 e MRC6 como apresentada em [1] e a diferença nos tempos de encriptação e decriptação nos leva a crer que os autores em sua publicação inicial contaram o tempo de

encriptação com a geração de chaves e de decriptação sem a geração de chaves já previamente gerada quando da encriptação.

	RC5	RC6	ERC6	MRC6
Encriptação	10.299	14.556	17.279	19.473
Decriptação	9.612	13.461	14.570	15.790

Tabela 2.2: Vazão de encriptação e decriptação em MB/s do RC5, RC6 e MRC6[1]

Tipo de algoritmo	r=8	r=12	r=16	r=20	r=24
RC5 enc.	24.212	16.596	12.155	10.299	8.649
RC5 dec.	22.868	16.021	11.942	9.612	8.229
RC6 enc.	33.500	23.651	17.939	14.556	12.237
RC6 dec.	32.355	22.203	16.909	13.461	11.167
ERC6 enc.	40.893	28.123	21.385	17.279	14.394
ERC6 dec.	35.052	23.971	17.973	14.570	12.174
MRC6 enc.	44.811	31.584	24.342	19.473	16.219
MRC6 dec.	37.392	25.595	19.509	15.790	13.130

Tabela 2.3: Vazão pelo número de rodadas em MB/s do RC5, RC6, ERC6 e MRC6[1]

Quanto ao efeito do número de rodadas sobre a vazão do cifrador considerando w=32 e b=16 temos os dados como apresentados em [1] apresentados na tabela 2.3, notando que novamente parece que a nossa consideração sobre a geração de chaves pode ser verdadeira quando verificamos a decriptação sempre mais rápida.

O efeito do tamanho dos blocos sobre a vazão para r=32 e b=16 apresentado em [1] é demonstrado na tabela 2.4.

Tipo de algoritmo	w=16	w=32
RC5 enc.	6.343	12.155
RC5 dec.	5.972	11.942
RC6 enc.	8.923	17.939
RC6 dec.	8.357	16.909
ERC6 enc.	10.450	21.385
ERC6 dec.	9.209	17.973
MRC6 enc.	12.130	24.342
MRC6 dec.	9.742	19.509

Tabela 2.4: Vazão pelo tamanho dos blocos em MB/s do RC5, RC6, ERC6 e MRC6[1]

Tipo de algoritmo	b=16	b=24	b=32	b=64	b=128
RC5 enc.	12.569	12.684	12.751	12.788	13.000
RC5 dec.	12.120	11.920	11.934	11.885	12.043
RC6 enc.	17.774	17.768	17.942	17.607	18.079
RC6 dec.	16.937	16.937	17.009	17.303	16.968
ERC6 enc.	21.229	21.317	21.596	21.780	21.995
ERC6 dec.	18.520	18.417	18.525	18.661	18.665
MRC6 enc.	24.221	24.217	24.221	24.263	23.943
MRC6 dec.	19.584	19.505	19.538	19.528	19.636

Tabela 2.5: Vazão pelo tamanho da chave em MB/s do RC5, RC6, ERC6 e MRC6[1]

E por fim o efeito do tamanho da chave sobre a vazão para r=16 e w=32 é apresentado na tabela 2.5.

2.4.5 Segurança do MRC6 avaliada pelos seus próprios autores

Na publicação original do cifrador não é apresentada nenhuma informação adicional sobre sua segurança até mesmo quando se tratando dos ataques linear e diferencial, talvez pelo fato de que sua concepção leva em consideração o fato de ser análogo ao RC6. Porém quando lidando com mais blocos sem interliga-los como da expansão do RC5 para o RC6 pode-se criar uma necessidade de um au-

mento do número de rodadas necessárias para que o cifrador possa ter seus dados de saída comparados com um gerador de dados pseudo aleatório. A não apresentação destas informações foi o que nos levou a realizar a verificação do cifrador levando em conta um dos melhores ataques conhecidos ao RC6 com a verificação de sua saída em relação a um gerador pseudo aleatório com o uso da análise χ^2.

Capítulo 3

Segurança do cifrador RC6

Conforme descrito no capítulo 2, o MRC6 pode ser visto como o resultado da aplicação de quatro cifradores RC6 em paralelo, seguida de um "entrelaçamento" de palavras extremamente simples - na verdade, este entrelaçamento é a única "novidade" do MRC6 - para misturar o efeito dos quatro RC6. É natural, portanto, que a primeira preocupação em uma análise de segurança do MRC6 seja verificar o impacto que as modificações propostas têm sobre as fraquezas já conhecidas para o RC6 - os autores do MRC6 parecem supor, ao contrário, que as suas modificações *reforçam* a resistência do RC6 à criptoanálise. Com efeito, como comprovam os resultados do capítulo 6, as fraquezas lineares que já haviam sido constatadas para o RC6 ficam de tal forma acentuadas no MRC6, que o ataque χ^2, o qual só conseguia sucesso parcial contra o RC6, permite a obtenção da chave em um ataque do tipo texto em claro escolhido, com esforço muito menor do que força bruta, para um MRC6 com até mais do que 20 rodadas.

Neste capítulo é apresentado um resumo de uma análise de segurança feita com a participação de alguns dos próprios autores do RC6, publicada como um relatório em 1998 [2]. Após analisar os melhores ataques disponíveis contra o RC6 (até 1998), o relatório conclui que os ataques mais bem sucedidos contra o RC6 (naquela época) eram os clássicos criptoanálise diferencial e criptoanálise linear. A tabela 3.1 apresenta um resumo do volume de textos em claro exigido nas

situações mais favoráveis a um ataque encontradas no relatório de Contini *et al.* [2]. Observa-se que, com as suas 20 rodadas nominais, o RC6 está bastante seguro contra ataques diferenciais e lineares. Um outro aspecto importante a ser observado na tabela 3.1 é que o RC6 é *mais vulnerável a criptoanálise linear do que a criptoanálise diferencial*, sendo que, com um ataque linear pode-se quebrar um RC6 com blocos de 128 bits com esforço menor do que força bruta *para até 16 rodadas*.

	Número de rodadas:				
Ataque:	8	12	16	20	24
Criptoanálise Diferencial	2^{56}	2^{117}	$\mathbf{2^{190}}$	$\mathbf{2^{238}}$	$\mathbf{2^{299}}$
Criptoanálise Linear	2^{47}	2^{83}	2^{119}	$\mathbf{2^{155}}$	$\mathbf{2^{191}}$

Tabela 3.1: Quantidades de textos em claro estimadas para os melhores ataques diferencial e linear contra o RC6 descritos em [2]

Atualmente, um dos ataques mais importantes (servindo de referência) contra o RC6 consiste em "reunir" diversas fraquezas descobertas na criptoanálise linear descrita em [2], de maneira a criar uma fraqueza única e mais acentuada, com o auxílio de uma ferramenta estatística bem conhecida: o teste χ^2. O ataque χ^2 é o tema dos capítulos 4, 5 e 6 e por isto não será discutido aqui.

Além de servir como base para a melhor técnica atual de criptoanálise do RC6, os ataques descritos por Contini *et al.* são relevantes para o presente trabalho porque tratam de um pré-requisito essencial para que se possa chegar a um ataque como o descrito no capítulo 6 contra um RC6 modificado: a habilidade de identificar fraquezas diferênciais e lineares em um cifrador cuja função de rodada *depende dos dados*. Por esta razão, eles são aqui resumidos.

3.1 Resistência do RC6 à criptoanálise diferencial

Nesta seção, após uma breve descrição dos conceitos essenciais da criptoanálise diferencial de Biham e Shamir [10], é discutida rapidamente a criptoa-

nálise diferencial do RC5 e são apresentados os detalhes do ataque diferencial ao RC6 descrito no relatório sobre a segurança do RC6 de Contini *et al.*. Existem algumas variantes do ataque básico que tiveram relativo sucesso contra outros cifradores, tais como a criptoanálise com diferenciais truncadas de Knudsen [24] ou a criptoanálise diferencial de ordem mais alta de Lai [25]. Como não se tem nenhuma notícia de sucesso destas técnicas contra o RC5, o qual está disponível para a comunidade acadêmica desde 1994, pode-se concluir que usá-las para atacar o RC5 é um problema bastante difícil. Então, dada a semelhança de projeto entre RC6 e RC5, é razoável supor que aplicá-las ao RC6 seja um problema, no mínimo, tão difícil quanto o seu uso contra o RC5 [2].

3.1.1 Visão geral de criptoanálise diferencial

Uma descrição mais aprofundada pode ser encontrada no trabalho pioneiro de Biham e Shamir [10] e também em alguns livros-texto sobre criptografia, tais como o livro de Stinson [3]. Para os propósitos desta seção, o resumo simples a seguir, adaptado de [2], é suficiente.

A criptoanálise diferencial é um ataque de texto em claro escolhido sobre cifradores em bloco iterativos. Por meio de uma cuidadosa escolha de pares de texto a serem cifrados, a *diferença* entre os respectivos estados na entrada da última rodada pode ser prevista com uma probabilidade conhecida. A definição de *diferença* depende das operações que são usadas no algoritmo do cifrador que está sendo atacado - definição de diferença mais utilizada: ou-exclusivo entre dois blocos de texto. A essência do metodo consiste em observar que a diferença prevista para o par de entradas da última rodada, juntamente com a diferença do par de textos cifrados correspondente, podem ser usadas para deduzir informações sobre a sub-chave da última rodada.

Note-se que a probabilidade de um par de textos em claro escolhido efetivamente produzir a diferença prevista para a entrada da rodada final usualmente decresce exponencialmente com o número de rodadas. Logo, ataques diferenciais em

versões reduzidas (com poucas rodadas) de cifradores iterativos podem ser devastadores, enquanto que versões completas (com número de rodadas adequado) dos mesmos cifradores podem ser imunes ao mesmo ataque.

A melhor escolha a impor para a diferença nos pares de texto em claro é descoberta investigando-se **características**, as quais especificam, rodada a rodada, diferenças esperadas para pares de texto que tenham sido submetidos ao cifrador. Há uma probabilidade associada à característica, a qual representa a chance da diferença da última rodada ocorrer, dado que a diferença especificada para a primeira rodada seja usada.

Lai *et al.* [26] apresentam uma importante generalização do conceito de características. Eles criaram o conceito de **diferenciais** para designar uma versão das características em que só importam as diferenças na entrada e na saída. Com relação às diferenças intermediárias, todas as possibilidades são consideradas, de tal forma que as diferenciais representam uma espécie de "feixe de características". Com isto, a probabilidade de ocorrência de um par (diferença na entrada, diferença na saída) é muito maior segundo o conceito de diferenciais do que segundo o conceito de características. As principais conclusões da criptoanálise diferencial do RC6 que constam no relatório de Contini *et al.* [2] presumem a utilização do conceito de diferenciais de Lai *et al.*.

3.1.2 Criptoanálise diferencial do RC5

Devido às semelhanças estruturais entre os dois cifradores, fica claro que um estudo aprofundado sobre os ataques que já foram descobertos contra o RC5 ajuda na concepção de ataques mais eficazes contra o RC6. Segundo o relatório de Contini *et al.* [2], a análise do RC5 desde a sua divulgação (em 1994) tem mostrado que ele é muito forte contra ataques diferenciais. Mesmo assim, a análise de diversos trabalhos mostra que a estratégia mais vantajosa é utilizar diferenças *que não afetem o valor das rotações*. Ocorre que qualquer diferença no valor de uma rotação rapidamente causa um efeito avalanche que dificulta muito a construção de

características úteis. Em conseqüência desta "lição" resultante de anos de ataques contra o RC5, todas as tentativas de criptoanálise diferencial contra o RC6 que aparecem no relatório usam diferenças que não produzem diferença nos valores de rotação para um dado par.

Uma outra conclusão importante da revisão bibliográfica sobre ataques diferenciais relatada em [2] é que uma maneira eficaz de controlar a evolução das diferenças ao longo de uma característica é manter baixo o peso de Hamming do maior número possível de diferenças intermediárias. Em função desta importante observação, os principais ataques diferenciais contra o RC6 descritos no relatório (e comentados a seguir) utilizam diferenças com apenas um bit não nulo.

3.1.3 Criptoanálise diferencial do RC6

Para os propósitos do presente trabalho, não é necessário discutir todos os detalhes e implicações do ataque diferencial ao RC6 descrito em [2]. Em vista disto, neste item é discutida apenas a origem da característica que foi utilizada. Isto é o suficiente para compreender o tipo de fraqueza que está sendo explorada, o que ajuda na identificação de novas fraquezas no RC6 e, por conseguinte, ajudará também na criptoanálise do MRC6 (ver capítulo 6).

Antes de apresentar a característica utilizada, é preciso listar algumas definições relevantes para o ataque diferencial de [2].

Definição 3.1 *[2] A expressão e_i será usada para denotar uma palavra de 32 bits que é nula, exceto por um único bit na posição i. Em termos de inteiros, temos que $e_i = 2^i$.*

Definição 3.2 *[2] Uma diferença genérica é representada como Δ.*

A descrição de [2] assume também algumas hipóteses, sendo que as mais importantes são as seguintes:

- em geral, um ataque diferencial a um cifrador com r rodadas vai exigir uma característica com $(r - 2)$ rodadas;

- no caso do RC6, existem duas medidas naturais de diferença:
  - xor bit-a-bit
  - subtração de inteiros módulo 2^{32}: termina sendo uma medida melhor de diferença para a análise do RC6, em conseqüência da função quadrática.

O relatório de Contini *et al.* é escrito de maneira a conduzir o leitor através de um conjunto de variantes simplificadas do RC6, em uma ordem progressiva de complexidade, até atingir o RC6 propriamente dito. As variantes são as seguintes:

- RC6-I-NFR:
  - o sufixo "I" se refere à substituição da função quadrática $f(x) = x.(2.x+1)$ pela função Identidade, ou seja, $f(x) = x$
  - o sufixo "NFR" significa que foi excluída do algoritmo original do RC6 a rotação fixa de $log\ w$ bits (NRF = No Fixed Rotations)
- RC6-I: apenas o efeito da função quadrática é descartado
- RC6-NFR: apenas o efeito da rotação fixa é descartado
- RC6: algoritmo original completo

Esta é uma abordagem muito interessante, pois isola os efeitos das duas maiores inovações na passagem do RC5 para o RC6. Além disto, as características e diferenciais obtidas para as versões mais simples servem de base para o que é mostrado para a versão completa. Para os propósitos desta dissertação, é suficiente comentar os resultados do relatório para as versões RC6-I-NFR, RC6-I e RC6, o que é feito a seguir.

3.1.3.1 Criptoanálise diferencial do RC6-I-NFR

Inicialmente, o relatório mostra que todas as características para um ataque ao RC6-I-NFR podem ser resumidas em três ciclos básicos. Apenas um

dos três ciclos já é suficiente para demonstrar os detalhes do ataque. Este ciclo está mostrado na tabela 3.2, aonde, conforme definição 3.2, Δ representa uma diferença genérica de 32 bits e aonde cada linha, a partir da segunda, representa o estado de um bloco após a ação de uma rodada de cifragem. No caso do RC6-I-NFR, a diferença é definida, por conveniência, como sendo o ou-exclusivo (xor) dos textos.

(a)				(b)				(c)			
Δ	Δ	0	0	e_{31}	e_{31}	0	0	e_t	e_t	0	0
Δ	0	0	0	e_{31}	0	0	0	e_t	0	0	0
0	0	0	Δ	0	0	0	e_{31}	0	0	0	e_s
0	Δ	Δ	0	0	e_{31}	e_{31}	0	0	e_u	e_s	0
Δ	Δ	0	Δ	e_{31}	e_{31}	0	e_{31}	e_u	e_u	0	e_v
Δ	Δ	Δ	0	e_{31}	e_{31}	e_{31}	0	e_u	e_u	e_v	0
Δ	Δ	0	0	e_{31}	e_{31}	0	0	e_u	e_u	0	0

Tabela 3.2: Característica iterativa básica para atacar o RC6-I-NFR

A coluna (b) da tabela 3.2 mostra uma escolha específica (a mais natural) para os valores de Δ. Conforme observado na literatura sobre o RC5, cada diferença tem somente um bit não-nulo. Note-se que é possível construir características mais gerais, permitindo que Δ assuma valores diversos, mas ainda mantendo o padrão do ciclo básico. Estas características estão mostradas na coluna (c) da mesma figura. É exatamente a reunião de todas as diversas possibilidades para os valores de s, u e v que não destroem o padrão (ou seja, com valores situados entre 5 e 31) que dá origem ao conceito de "diferencial" de Lai *et al.* [26]. No relatório de Contini *et al.* são discutidas as probabilidades para as diversas características e também para a diferencial que as engloba. No caso específico da diferencial, a probabilidade vale $\approx 2^{-22}$. É importante notar que todas as características, assim como a diferencial, que aparecem na tabela 3.2 são *iterativas*, ou seja, podem ser "encadeadas" para atacar mais rodadas do que as mostradas.

3.1.3.2 Criptoanálise diferencial do RC6-I

Em seguida, Contini *et al.* removem a restrição relativa à rotação fixa de $log\ w$ bits, passando a se concentrar na versão simplificada RC6-I. A única simplificação agora é que a função quadrática não atua, ficando no seu lugar a função identidade. Como a única inclusão é a de uma rotação fixa, fica fácil adaptar as características da tabela 3.2 para este caso. O resultado desta adaptação é mostrado na tabela 3.3, aonde a coluna (a) mostra uma característica de 6 rodadas já generalizada, para permitir a visualização da diferencial associada, e aonde a coluna (b) mostra uma das possíveis características que podem ser montadas com base nesta generalização. Neste caso, a probabilidade estimada da diferential (iterativa) de 6 rodadas mostrada na tabela 3.3 é de $\approx 2^{-23}$. Aqui também o conceito de "diferença" envolve o ou-exclusivo (xor) dos blocos de texto.

(a)				(b)			
e_{t+5}	e_t	0	0	e_{16}	e_{11}	0	0
e_t	0	0	0	e_{11}	0	0	0
0	0	0	e_s	0	0	0	e_{26}
0	e_u	e_s	0	0	e_{26}	e_{26}	0
e_u	e_{u-5}	0	e_v	e_{26}	e_{21}	0	e_{26}
e_{u-5}	e_{u-10}	e_v	0	e_{21}	e_{16}	e_{26}	0
e_{u-10}	e_{u-15}	0	0	e_{16}	e_{11}	0	0

Tabela 3.3: Característica iterativa generalizada de 6 rodadas para atacar o RC6-I e caso particular correspondente.

3.1.3.3 A função quadrática

O próximo passo, a caminho de obter características e diferenciais para o RC6 completo, é incluir a função quadrática. Aqui a noção de "diferença" é modificada, de modo a representar adequadamente o RC6 completo. A diferença

passa agora a ser definida como subtração de inteiros (módulo 2^{32}). Esta definição é bastante conveniente neste caso, pois permite "passar direto" pela operação de adição das sub-chaves, No entanto, a interação entre esta definição de diferença e a operação de xor que faz parte do cifrador teve que ser tratada com cuidado. Contini *et al.* apresentam um resultado que ajuda bastante nesta interação. Trata-se do "Lema 8" do relatório, o qual permite computar as probabilidades de uma característica ao passar por uma função do tipo $w = z \oplus f(x)$, aonde $f(x)$ é a função quadrática da definição do RC6. Este lema se constitui, possivelmente, no resultado mais importante daquele relatório e, por esta razão, é aqui reproduzido.

Para dois conjuntos de entradas x_1, z_1 e x_2, z_2, sejam $w_1 = z_1 \oplus f(x_1)$ e $w_2 = z_2 \oplus f(x_2)$, conforme representação esquemática na figura 3.1.

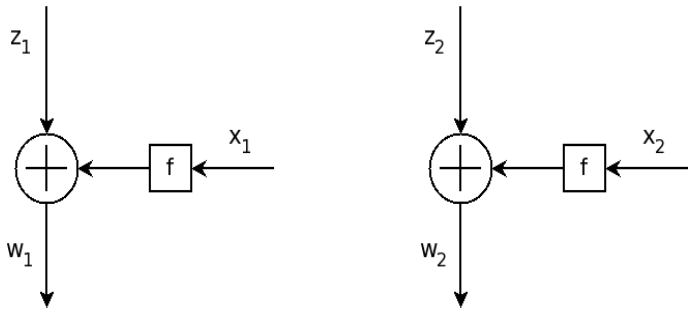


Figura 3.1: Representação esquemática da composição da função quadrática com a operação $\oplus$.

Agora, sejam $\delta_x = x_2 - x_1$, $\delta_z = z_2 - z_1$ e $\delta_w = w_2 - w_1$.

Lema 3.1 *(Lema 8 de [2]) Seja p_i a probabilidade da característica $(\delta_x, \delta_z) \rightarrow \delta_w$, aonde $(\delta_x, \delta_z) = (2^i, 0)$ e $\delta_w = 2^i$. Similarmente, seja q_i a probabilidade da característica $(\delta_x, \delta_z) \rightarrow \delta_w$, aonde $(\delta_x, \delta_z) = (2^i, 2^i)$ e $\delta_w = 0$. Então sempre ocorre $p_i = q_i$ e além disto:*

$$p_i = q_i = \begin{cases} 2^{i-31} & for\ 15 \leq i \leq 31, \\ x \in [2^{i-35}, 2^{i-30}] & for\ 0 \leq i \leq 14. \end{cases}$$

Prova: ver [2]. □

O caso $0 \leq i \leq 14$ tem um tratamento complexo e o resultado mostrado foi obtido

experimentalmente, A tabela 3.4 mostra a diferença entre as probabilidades reais e 2^{i-31}, para $0 \leq i \leq 14$. Note-se que a probabilidade p_i decresce monotonicamente com i, mas a uma taxa não constante.

i	2^{31-i}	$1/p_i$	i	2^{31-i}	$1/p_i$
0	2^{31}	$2^{30.10}$	1	2^{30}	$2^{29.15}$
2	2^{29}	$2^{28.57}$	3	2^{28}	$2^{28.17}$
4	2^{27}	$2^{27.71}$	5	2^{26}	$2^{27.21}$
6	2^{25}	$2^{26.89}$	7	2^{24}	$2^{26.42}$
8	2^{23}	$2^{25.92}$	9	2^{22}	$2^{25.44}$
10	2^{21}	$2^{24.98}$	11	2^{20}	$2^{23.09}$
12	2^{19}	$2^{19.58}$	13	2^{18}	$2^{18.10}$
14	2^{17}	$2^{17.01}$			

Tabela 3.4: Probabilidade p_i do Lema 3.1 para $0 \leq i \leq 14$.

3.1.3.4 Criptoanálise diferencial do RC6 completo

Com base no lema 3.1, é possível construir as duas características de uma rodada para o RC6 que estão mostradas na tabela 3.5. Basta combinar estas duas características com aquelas que foram obtidas para o RC6-I, mostradas na tabela 3.3, para se chegar à característica iterativa de 6 rodadas para o RC6 mostrada na tabela 3.6. Levando em conta todas as possibilidades para esta "família" de características, Contini *et al.* concluem que a probabilidade da diferencial respectiva fica 4 vezes maior do que a da melhor característica, chegando a 2^{-91}. Isto corresponde justamente à melhor complexidade por eles encontrada para um ataque diferencial a um RC6 com 8 rodadas usando a diferencial de forma iterativa (ver tabela 3.7).

Diferenciais e características iterativas são muito interessantes porque podem ser "encadeadas" para montar ataques a mais rodadas de um cifrador. Entretanto, a exigência de que sejam iterativas impede que sejam aquelas com maior

0	0	e_{t+5}	e_t		0	0	0	e_t
		$\downarrow$					$\downarrow$	
0	0	e_t	0		0	e_s	e_t	0

Tabela 3.5: Características de 1 bit para uma rodada do RC6.

(a)				(b)			
e_{t+5}	e_t	0	0	e_{16}	e_{11}	0	0
e_t	0	0	0	e_{11}	0	0	0
0	0	0	e_s	0	0	0	e_{26}
0	e_u	e_s	0	0	e_{26}	e_{26}	0
e_u	e_{u-5}	0	e_v	e_{26}	e_{21}	0	e_{26}
e_{u-5}	e_{u-10}	e_v	0	e_{21}	e_{16}	e_{26}	0
e_{u-10}	e_{u-15}	0	0	e_{16}	e_{11}	0	0

Tabela 3.6: Característica **iterativa** generalizada de 6 rodadas para atacar o RC6 e um caso particular correspondente (I_6).

probabilidade, para um dado número de rodadas. Ou seja, se removermos a restrição de que uma característica de 6 rodadas deva ser iterativa, é possível encontrar outras características de 6 rodadas com maior probabilidade. Com isto em mente, Contini *et al.* encontraram diversas outras opções de características (diferenciais) que não eram iterativas mas que podiam ser encadeadas *em uma das extremidades* da característica iterativa. Com isto, eles montaram características "sob medida" para cada número de rodadas que tinham maior probabilidade do que simplesmente encadear a característica iterativa. Eles também anteciparam um certo "truque" típico dos criptoanalistas para reduzir a quantidade de textos exigidos no ataque. Levando tudo isto em consideração, eles obtiveram, por exemplo, uma diferencial não-iterativa que podia ser encadeada ao final da diferencial I_6 e que tinha uma probabilidade associada de 2^{-71}, à qual eles denominaram de E_6. Eles também montaram mais uma

diferencial não-iterativa de 6 rodadas que não podia ser encadeada nem no ínicio ("*B*eginning") e nem no final ("*E*nd") da diferencial I_6 e chamaram de E'_6. Esta E'_6 tinha probabilidade de 2^{-56} e servia somente para atacar o RC6 com 8 rodadas. Da mesma forma, eles montaram uma B_6, uma E_2 e uma E_4 e estimaram as probabilidades associadas. Depois, buscando as combinações ótimas entre estas diferenciais, eles chegaram aos valores de complexidade mostrados na tabela 3.7.

	Número de rodadas:				
Variante:	8	12	16	20	24
característica iterativa	2^{93}	2^{151}	2^{214}	2^{279}	2^{337}
diferencial iterativa	2^{91}	2^{147}	2^{210}	2^{273}	2^{329}
diferenciais "sob-medida"	2^{56} E'_6	$\mathbf{2^{117}}$ $B_6 - E_4$	2^{190} $B_6 - I_6$ $-E_2$	2^{238} $B_6 - I_6$ $-E_6$	2^{299} $B_6 - I_6$ $I_6 - E_4$

Tabela 3.7: Estimtivas para as quantidades de textos em claro necessárias para montar um ataque diferencial contra o RC6 [2].

Em resumo, a tabela 3.7 contém as situações mais favoráveis a uma criptoanálise diferencial do RC6, encontradas em um excelente estudo sobre a segurança deste cifrador publicado por alguns dos próprios autores do RC6, ainda durante o desenrolar do processo de escolha do AES. Nota-se que, na melhor das hipóteses, a criptoanálise diferencial consegue quebrar um RC6 de 12 rodadas com esforço menor do que força bruta. Mas isto exigiria quase a totalidade do codebook (2^{117} textos em claro!), além de exigir a aplicação do conceito de diferenciais, uma sofisticação teórica do clássico conceito de características de Biham e Shamir. De um ponto de vista prático, tal criptoanálise exigiria também a busca de diferenciais ótimas para um dado número de rodadas e até mesmo a antecipação dos "truques" práticos empregados pelos criptoanalistas no momento de um ataque real. Tudo

isto mostra, é claro, que o RC6 foi projetado de modo a frustrar qualquer tentativa de quebra baseada em criptoanálise diferencial, a qual era, reconhecidamente, uma das duas melhores técnicas de criptoanálise no momento do projeto do RC6. Um outro importante requisito do projeto cumprido pelo RC6 foi uma alta resistência à criptoanálise linear, detalhada na próxima seção.

Além de ajudar a conhecer melhor o projeto do RC6 (e do MRC6), o estudo da criptoanálise diferencial, resumido nesta seção, é importante para a compreensão dos ataques lineares descritos a seguir, os quais se constituem no principal fundamento dos ataques estatísticos desenvolvidos e implementados no âmbito desta dissertação.

3.2 Criptoanálise linear do RC6

Os ataques χ^2 a que se referem os capítulos 5 e 6 exploram fraquezas cuja origem pode ser explicada por uma análise da *linearidade* dos respectivos cifradores. Um cifrador tem que parecer um gerador pseudo-aleatório para quem tem acesso somente a textos por ele cifrados. Entre outras coisas, isto significa que *qualquer combinação linear* de bits específicos de saída e entrada tem que ter a mesma chance de ocorrer ou não ocorrer, ou seja, tem que ter probabilidade muito próxima de 50%. Por exemplo, o ataque χ^2 ao RC6 (ver capítulo 5) se baseia no fato de que combinações lineares envolvendo os cinco últimos bits da primeira e da terceira palavras de cada bloco de entrada e os respectivos bits na saída ocorrem com probabilidades nitidamente afastadas de 50% (ver [2] e também discussão em [4]).

Isto mostra que a aplicação da criptoanálise estatística como é feita nesta dissertação depende de uma razoável compreensão de conceitos relacionados à criptoanálise linear, tais como linearidade e distância de funções booleanas. Adicionalmente, é preciso ter clareza sobre o modo como uma criptoanálise linear é realizada, de modo a compreender a origem das fraquezas lineares já localizadas no RC6 e também a maneira como esta informação auxiliou na identificação das

fraquezas exploradas no ataque ao MRC6 descrito no capítulo 6.

Em vista disto, esta seção fornece inicialmente uma visão geral sobre criptoanálise linear, incluindo alguns conceitos fundamentais sobre a teoria de aproximações lineares de funções booleanas, com base nos textos de Pieprzyk [27] e de Stinson [3]. Em seguida, são detalhados os ataques de criptoanálise linear específicos contra o RC6, que constam no relatório de Contini *et al.* [2].

3.2.1 Visão geral de criptoanálise linear

A criptoanálise linear, desenvolvida por Matsui [11], é uma técnica criptoanalítica que pode ser aplicada a qualquer cifrador iterativo. Baseia-se em relações lineares probabilísticas entre alguns bits do texto em claro, alguns bits do estado antes das substituições na última rodada e alguns bits da chave. Ou seja, assume-se que é provável que exista um subconjunto do total de bits envolvidos na cifragem cujo $\oplus$ se comporta de uma maneira não aleatória, assumindo o valor 0, por exemplo, com probabilidade afastada de 1/2.

O procedimento de ataque está baseado na suposição de que um oponente tem acesso a um grande número de pares de texto claro-cifrado, cifrados com a mesma chave K:

1. para cada par, ele inicia a decifragem do texto cifrado, usando todos os possíveis candidatos para a chave da última rodada;

2. para cada candidato, ele computa os bits de estado envolvidos na relação linear e determina se a referida relação linear vale;

3. sempre que valer, ele incrementa o contador correspondente;

4. valores corretos para estes bits de chave vêm do candidato a chave que tiver a contagem mais afastada de $1/2 \times \#$ pares.

Uma estimativa da complexidade deste ataque depende do "lema do Empilhamento", um resultado da teoria das probabilidades discutido a seguir, com base no texto de Stinson [3].

3.2.1.1 O Lema do Empilhamento

Sejam $\mathbf{X}_1, \mathbf{X}_2, \ldots$ variáveis aleatórias independentes assumindo valores do conjunto $\{0,1\}$ e sejam $p_1, p_2, \ldots$ números reais tais que $0 \leq p_i \leq 1$. Suponha ainda que $\mathbf{Pr}[\mathbf{X}_i = 0] = p_i$ (o que significa que $\mathbf{Pr}[\mathbf{X}_i = 1] = 1 - p_i$).

Agora, considerando-se a variável aleatória discreta $\mathbf{X}_i \oplus \mathbf{X}_j$, observa-se que ela tem a seguinte distribuição de probabilidades:

$$\mathbf{Pr}[\mathbf{X}_i \oplus \mathbf{X}_j = 0] = p_i p_j + (1 - p_i)(1 - p_j) \tag{3.1}$$

$$\mathbf{Pr}[\mathbf{X}_i \oplus \mathbf{X}_j = 1] = p_i(1 - p_j) + (1 - p_i)p_j \tag{3.2}$$

Na análise linear, é conveniente reescrever as equações acima em termos do conceito de *tendência*, definido a seguir.

Definição 3.3 *A **tendência** da variável aleatória $\boldsymbol{X}_i$ é definida como:*

$$\epsilon_i = p_i - \frac{1}{2} \tag{3.3}$$

Definição 3.4 *Para $i_1 < i_2 < \cdots < i_k$, $\boldsymbol{\epsilon_{i_1,i_2,\ldots,i_k}}$ é definido como a tendência da variável aleatória $\boldsymbol{X_{i_1} \oplus \cdots \oplus X_{i_k}}$*

Fácil constatar que $\epsilon_{i_1,i_2} = 2\epsilon_{i_1}\epsilon_{i_2}$. A generalização desta idéia consiste no lema a seguir. Note-se que este lema só vale quando as variáveis são independentes.

Lema 3.2 *(Lema do Empilhamento) Seja $\epsilon_{i_1,i_2,\ldots,i_k}$ a tendência da variável aleatória $\boldsymbol{X}_{i_1} \oplus \cdots \oplus \boldsymbol{X}_{i_k}$. Então: $\epsilon_{i_1,i_2,\ldots,i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$*

Prova: por indução. □

NOTA: este lema pressupõe independência entre as rodadas de cifragem, o que não é rigorosamente verdadeiro no caso de um cifrador real. Porém, ele permite estimativas suficientemente precisas das ordens de grandeza do ataque.

Este lema é útil porque permite obter a tendência do "ou-exclusivo de algumas variáveis aleatórias discretas" a partir das tendências das variáveis individuais. Uma vez que uma aproximação linear consiste justamente no "ou-exclusivo

de variáveis especialmente escolhidas", o lema do empilhamento se torna fundamental para que se possa estimar a probabilidade de sucesso do ataque e daí chegar à complexidade envolvida. Ocorre que quanto maior a tendência de uma aproximação linear, menor é o número de textos que precisam ser tomados para que um bit correto de chave possa ser deduzido. Mais precisamente, suponha que a tendência de uma aproximação A_0 seja dada por ϵ_0. Então Matsui demonstra em [11] que a quantidade de texto em claro requerida para explorar esta tendência com uma alta taxa de sucesso é dada por:

$$c \times \epsilon_0^{-2} \tag{3.4}$$

aonde c é alguma constante que depende do estilo do ataque que foi preparado.

Note-se que este resultado contrasta com a criptoanálise diferencial, aonde a quantidade de dados exigidos é simplesmente proporcional ao inverso da probabilidade da diferencial (ver seção 3.1).

3.2.1.2 Aproximações lineares de funções binárias

Este item consiste em uma adaptação do item 3.5.1, do cap. 3 ("Private-Key Cryptosystems"), do livro de Pieprzyk [27]. São fornecidos alguns elementos importantes para a compreensão do conceito de linearidade, o qual é central para os ataques estatísticos implementados no presente trabalho.

Definição 3.5 *Uma função booleana $h : \Sigma^n \rightarrow \Sigma$ em n variáveis $s_1, \ldots, s_n$ é dita* ***linear*** *se pode ser representada como:*

$$h(s) = a_1 s_1 \oplus \ldots \oplus a_n s_n \tag{3.5}$$

onde $a_i \in \Sigma = \{0, 1\}, \;\; i = 1, \ldots, n.$

Ou seja, uma função booleana é linear se a sua saída pode ser obtida a partir de uma simples representação linear das suas entradas.

Adicionalmente, o conjunto de todas as funções booleanas lineares em n variáveis é denotado por:

$$\mathcal{L}_n = \{h : \Sigma^n \rightarrow \Sigma \;\; | \;\; h = a_1 s_1 \oplus \ldots \oplus a_n s_n\} \tag{3.6}$$

Este conjunto é importante porque inclui *todas* as possibilidades de funções lineares que poderão ser utilizadas como aproximação para alguma função não-linear de interesse. Uma análise linear da caixa-S de um cifrador, por exemplo, busca identificar qual função deste conjunto está mais próxima de cada função não-linear que faz parte da transformação que esta caixa-S representa.

Definição 3.6 *Uma função booleana* $f : \Sigma^n \to \Sigma$ *é dita* ***afim*** *se, para algum* $h(s) \in \mathcal{L}_n$*:*

$$f(s) = h(s) \quad ou \quad f(s) = h(s) \oplus 1 \tag{3.7}$$

Ou seja, f será chamada de "afim" se puder ser representada exatamente por pelo menos uma das funções lineares possíveis (ou a sua negação) nas mesmas variáveis que as dela.

O conjunto de todas as funções booleanas afins em n variáveis é:

$$\mathcal{A}_n = \mathcal{L}_n \cup \{h \oplus 1 \mid h \in \mathcal{L}_n\} = \mathcal{L}_n \cup \overline{\mathcal{L}_n} \tag{3.8}$$

o que corresponde a todas as funções lineares e suas negações.

O principal instrumento de análise de linearidade de uma função booleana é a sua tabela verdade. Uma função booleana é representada de maneira única por esta tabela.

Definição 3.7 *Assuma que o argumento* $\alpha_i \in \Sigma^n$ *percorre todos os seus possíveis valores* $0, 1, \ldots, 2^n - 1$*:*

$$\begin{aligned} \alpha_0 &= (00\ldots0) \\ \alpha_1 &= (00\ldots1) \\ &\ldots \\ \alpha_{2^n-1} &= (11\ldots1) \end{aligned}$$

A ***tabela verdade*** *de* f *é equivalente ao vetor:*

$$f = (f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{2^n-1})) \tag{3.9}$$

Um conceito fundamental, definido a seguir, é aquele que permite avaliar quão próxima uma dada função não-linear (que descreve um bit de uma caixa-S, por exemplo) está de alguma função linear: a distância de Hamming.

Definição 3.8 *A **distância de Hamming** $d(f,g)$ entre duas funções booleanas $f,g:\Sigma^n\rightarrow\Sigma$ é o número de 1s no vetor:*

$$(f(\alpha_0)\oplus g(\alpha_0), f(\alpha_1)\oplus g(\alpha_1),\ldots,f(\alpha_{2^n-1})\oplus g(\alpha_{2^n-1})) \tag{3.10}$$

Ou seja, a distância de Hamming entre duas funções f e g é o número de *discordâncias* entre f e g.

Finalmente, de posse de todos os "instrumentos" anteriores, é possível compor um conceito central na análise linear de cifradores: a não-linearidade. A não-linearidade de uma função f é a *distância* entre f e a sua *melhor aproximação linear*, ou seja:

Definição 3.9 *A **não-linearidade** $N(f)$ de uma função booleana $f:\Sigma^n\rightarrow\Sigma$ é:*

$$N(f) = min\ d(h,f)\ \ \forall h\in\mathcal{A}_n \tag{3.11}$$

Note-se que, conhecendo a distância $d(f,h)=d_{f,h}$, onde $h\in\mathcal{L}_n$, é fácil de obter $d(f,h\oplus 1)=2^n-d_{f,h}$. Desta forma, para determinar a não-linearidade de uma função $f:\Sigma^n\rightarrow\Sigma$, é suficiente encontrar todas as distâncias entre a função e o conjunto de funções lineares, já que as distâncias para as funções afins do conjunto $\overline{\mathcal{L}_n}$ podem ser computadas a partir das distâncias das funções lineares.

Exemplo 3.1 *Para a função $f:\Sigma^2\rightarrow\Sigma$ dada por $f(s)=s_1s_2$, a tabela-verdade e as funções lineares de $\mathcal{L}_2=\{0,s_1,s_2,s_1\oplus s_2\}$ são mostradas na tabela 3.8. Logo, as distâncias de f até as funções que fazem parte de $\mathcal{L}_2$ são:*

- $d(f,0)=d(f,s_1)=d(f,s_2)=1$
- $d(f,s_1\oplus s_2)=3$ □

(s_2, s_1)	f	0	s_1	s_2	$s_1 \oplus s_2$	$f \oplus s_1$	$f \oplus s_2$	$f \oplus s_1 \oplus s_2$
00	0	0	0	0	0	0	0	0
01	0	0	1	0	1	1	0	1
10	0	0	0	1	1	0	1	1
11	1	0	1	1	0	0	0	1

Tabela 3.8: Tabela verdade de $f = s_1 s_2$ e funções lineares de $\mathcal{L}_2 = \{0, s_1, s_2, s_1 \oplus s_2\}$

3.2.1.3 Aproximações Lineares de Caixas-S

A força criptográfica de um algoritmo de cifragem moderno se apóia em uma escolha adequada da estrutura criptográfica da rodada de cifragem. Tais estruturas podem ser vistas como uma *coleção de funções booleanas*, ou "Caixas-S", ligando bits de entrada e de saída em cada rodada de cifragem. Uma definição mais precisa deste conceito é dada a seguir.

Definição 3.10 *[27] Uma* ***caixa-S*** $S : \Sigma^n \rightarrow \Sigma^m$ *é uma coleção de* m *funções* $f_i : \Sigma^n \rightarrow \Sigma$ *em* n *variáveis booleanas* $s = (s_1, \ldots, s_n)$ *para as quais:*

$$S(s) = (f_1(s), \ldots, f_m(s)) \tag{3.12}$$

A noção de não-linearidade foi definida para apenas uma função booleana, mas ela pode ser naturalmente estendida para caixas-S.

Definição 3.11 *[27] A* ***não-linearidade*** *de uma caixa-S* $(n \times m)$ *dada por* $S = (f_1, \ldots, f_m)$ *é definida como:*

$$N(S) = min\ N(w_1 f_1 \oplus \ldots \oplus w_m f_m \oplus v) \tag{3.13}$$

aonde $w = (w_1, \ldots, w_m) \in \Sigma^m \quad ; \quad v \in \Sigma$

Não-linearidade de uma caixa-S é, portanto, a menor distância entre *qualquer combinação linear dos seus bits de saída* e cada uma das funções afins nas mesmas variáveis. Logo, quanto menor a não-linearidade, maior a "brecha linear" na definição (não-linear) de uma caixa-S.

Mesmo a caixa-S sendo completamente não-linear, ou seja, mesmo se nenhuma das funções que definem os seus bits de saída pode ser considerada "afim", é útil *quantificar a proximidade* destas funções não-lineares em relação a todas as possibilidades de funções lineares nas mesmas variáveis. Se alguma das funções não-lineares for muito próxima de uma linear, um ataque linear eficiente pode ser montado assumindo-se que, em vez da (incômoda) função não-linear, a caixa-S utiliza a linear que lhe é semelhante. Para uma certa fração fixa de todos os casos possíveis, está hipótese estará correta. É justamente com base nestes casos que é montado o ataque.

O exemplo a seguir, adaptado de Stinson [3], ilustra claramente o processo de busca de "brechas lineares" nas funções não-lineares que definem uma caixa-S. De modo a quantificar as probabilidades de ocorrência de determinados casos, Stinson define os bits de entrada e saída de uma caixa-S ($m \times n$) como variáveis aleatórias:

Definição 3.12 *$X = (x_1, \ldots, x_n)$ define os **bits de entrada** e $Y = (y_1, \ldots, y_n)$ define os **bits de saída**, aonde Y corresponde a uma permutação de X dada por S, ou seja:*

$$(y_1, \ldots, y_n) = \pi_S(x_1, x_2, \ldots, x_n) \tag{3.14}$$

Nesta definição, cada coordenada representa uma variável aleatória uniforme:

- x_i define $\mathbf{X}_i$, com tendência $\epsilon_i = 0$ (ou seja, com $p_i = 1/2$)
- y_i define $\mathbf{Y}_i$, com tendência $\epsilon_i = 0$

Computa-se, então, a tendência (afastamento de $p = 1/2$) de variáveis aleatórias do tipo:

$$\mathbf{X}_{i_1} \oplus \cdots \mathbf{X}_{i_n} \oplus \mathbf{Y}_{j_1} \oplus \cdots \mathbf{Y}_{j_\ell} \tag{3.15}$$

A idéia chave é que variáveis aleatórias deste tipo com tendência afastada de 0 (ou seja, com probabilidade significativamente diferente de 1/2) representam "brechas" em potencial para ataques criptoanalíticos lineares.

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Tabela 3.9: Caixa-S exemplo utilizada por Stinson [3]

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{Y}_1$	$\mathbf{Y}_2$	$\mathbf{Y}_3$	$\mathbf{Y}_4$
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

Tabela 3.10: $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4 \; ; \; \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \mathbf{Y}_4)$ para o exemplo 3.2

Exemplo 3.2 *Seja a caixa-S* $\pi_S : \{0,1\}^4 \rightarrow \{0,1\}^4$ *dada pela tabela 3.9.*

- *Nas linhas da tabela 3.10 estão todas as combinações (possíveis) de valores das oito variáveis aleatórias* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_4, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_4$
- *Nota-se que* $\boldsymbol{Pr}[\boldsymbol{X}_1 \oplus \boldsymbol{X}_4 \oplus \boldsymbol{Y}_2 = 0] = \frac{1}{2}$ *e* $\boldsymbol{Pr}[\boldsymbol{X}_1 \oplus \boldsymbol{X}_4 \oplus \boldsymbol{Y}_2 = 1] = \frac{1}{2}$
  - *logo. a tendência desta variável aleatória é 0*
- *Se fosse a variável* $\boldsymbol{X}_3 \oplus \boldsymbol{X}_4 \oplus \boldsymbol{Y}_1 \oplus \boldsymbol{Y}_4$*, o valor da tendência seria* $-3/8$*.*
- *Não é difícil computar as tendências de todas as 256 variáveis aleatórias que possuem esta forma.* □

As 256 tendências do exemplo anterior podem ser colocadas na forma de uma tabela 16×16, em que as 16 linhas representam todas as possíveis combinações de $\mathbf{X}_1$ a $\mathbf{X}_4$ e as colunas representam as possíveis combinações para $\mathbf{Y}_1$ a $\mathbf{Y}_4$. Tabelas deste tipo, chamadas de **Tabelas de Aproximações Lineares** de uma caixa-S, constituem-se no principal elemento na montagem de um ataque de criptoanálise linear a um cifrador.

3.2.2 Aproximações lineares para o RC6

Diversos conceitos e definições fundamentais para a parte do relatório de Contini *et al.* relativa a criptoanálise linear são discutidos na seção anterior (ver seção 3.1).

Contini *et al.* definem aproximações de uma maneira ligeiramente diferente da utilizada por Stinson [3] e discutida na seção anterior. Eles consideram uma palavra x de 32 bits como um vetor em $\mathbb{Z}_2^{32}$ e utilizam uma quantidade de 32 bits (τ), para representar os bits de x que entram em uma dada aproximação linear. Assim, $x \cdot \tau$ pode ser visto como o produto escalar entre x e τ, ou então, mais simplesmente, como a aplicação da *máscara* τ sobre a string x. Um exemplo de aproximação linear escrita desta forma é a seguinte:

$$(A \cdot e_0) \oplus (C' \cdot e_0) \oplus (D' \cdot e_0) \quad = \quad k$$

Aproximações lineares podem ser combinadas e a tendência (ver definição 3.3) de uma aproximação composta pode ser estimada com o auxílio do lema do empilhamento (lema 3.2). Em vista da vasta gama de ataques lineares que podem ser montados sobre o RC6, Contini *et al.* [2] definem dois tipos diferentes de aproximação, as quais eles definem como sendo "Tipo I" e "Tipo II". Cada tipo (descrito a seguir) consiste em um formato diferente para aproximar as rotações dependentes dos dados. Aproximações do tipo I envolvem *o valor das rotações* na expressão linear utilizada, enquanto que expressões do tipo II envolvem apenas o dado que vai ser rotacionado e aquele que resulta após a rotação. Cada uma leva a um tipo diferente de ataque linear.

As aproximações do tipo II são utilizadas de maneira muito semelhante àquilo que é feito em um ataque diferencial (ver seção 3.1). Elas são apropriadas para um acompanhamento detalhado do que acontece em cada operação de uma rodada do RC6. Aparentemente, Contini *et al.* acreditavam que este nível de detalhe permitiria vislumbrar mais facilmente as aproximações que levam aos ataques lineares mais eficazes contra o RC6. No entanto, as aproximações do tipo I permitem levar em conta a estrutura de uma rodada do RC6 de maneira muito mais eficaz, evitando convenientemente a necessidade de levar em consideração aproximações para as funções quadráticas. Em conseqüência disto, os melhores ataques lineares descritos no relatório acabam sendo aqueles que seguem as aproximações do tipo I. Os ataques do tipo II acabam tendo apenas valor teórico, como auxiliares na compreensão da estrutura interna do RC6.

São os ataques do tipo I que levaram os autores do RC6 a definirem 20 rodadas como adequados para oferecer o nivel de segurança exigido pelo AES. Ataques do tipo I são também os que servem de base para o ataque chi-quadrado ao RC6 de Knudsen e Meier [4]. Por esta razão, no texto a seguir são descritos em detalhe somente os ataques associados a aproximações do tipo I.

Definição 3.13 *Para* $\Gamma_a, \Gamma_b, \Gamma_c \in \{0,1\}^{32}$ *o Tipo I de aproximação linear para o RC6 envolve aproximações para as rotações dependentes de dados,* $A = B <<< C$*, com a seguinte forma:*

$$A \cdot \Gamma_a \;\; = \;\; B \cdot \Gamma_b \oplus C \cdot \Gamma_c$$

Se assumirmos que Γ_a pode ser rotacionado em Γ_b em diferentes t maneiras então a tendência desta aproximação - $\frac{t}{32}$ corresponde à possibilidade de uma rotação cíclica qualquer levar Γ_a para Γ_b - é dada por $\rho = \frac{1}{2} \times \frac{t}{32}$. Especificamente no contexto do RC6, os bits de Γ_a são forçados a consistirem daqueles das posições menos significativas ($lg(w)$), o que garante que Γ_a só pode ser mapeado em Γ_b de uma maneira (t=1) e então, neste caso, a tendência da aproximação vale sempre 2^{-6}.

Como as rotações dependentes de dados não provocam queda na tendência, independente do peso de Hamming de Γ_i, e já que sempre vai existir uma queda na tendência através de uma soma de inteiros com o aumento do peso de Hamming das variáveis envolvidas, a análise de Contini *et al.* se concentrou exclusivamente no uso de aproximações "single-bit" do tipo I.

Definição 3.14 *Para* $\Gamma_a, \Gamma_b, \Gamma_c \in \{0,1\}^{32}$ *o Tipo II de aproximação linear para o RC6 envolve aproximações para as rotações dependentes de dados,* $A = B <<< C$*, com a seguinte forma:*

$$A \cdot \Gamma_a \quad = \quad B \cdot \Gamma_b$$

Como a probabilidade de acerto desta relação linear depende apenas dos bits de A e B envolvidos, a tendência desta proximação é dada pela mesma expressão da aproximação do tipo I. Os bits de C acabam tendo apenas efeito linear sobre o resultado da rotação.

3.2.2.1 Utilizando aproximações lineares do tipo I

Esta análise se baseia na seguinte aproximação linear de duas rodadas do RC6:

$$(A \cdot e_t) \oplus (C \cdot e_s) = (A'' \cdot e_u) \oplus (C'' \cdot e_v) \tag{3.16}$$

aonde A e C são a primeira e terceira palavras de algum dado intermediario, e A'' e C'' são a primeira e a terceira palavras depois de duas rodadas de encriptação usando o RC6.

Esta relação pode ser representada graficamente como mostra a tabela 3.11, com uma aproximação genérica na coluna (a) e uma escolha específica na coluna (b).

A origem desta expressão, a qual serve de base para os melhores ataques lineares descritos no relatório de Contini *et al.*, está em uma cuidadosa observação da estrutura do RC6. Com o uso de aproximações do tipo I, é possível aproveitar o "entrelaçamento" das palavras durante uma rodada do RC6 para evitar

(a)				(b)			
e_t	-	e_s	-	e_0	-	e_0	-
-	e_u	-	e_v	-	e_0	-	e_0
e_u	-	e_v	-	e_0	-	e_0	-

Tabela 3.11: Representação gráfica da relação linear que utiliza aproximações do tipo I [2].

ter que considerar aproximações para a função quadrática. Ocorre que o bit que resulta após a rotação fixa na metade esquerda (por exemplo) de uma rodada do RC6 (ver figura 2.3), dado por um bit de u ou t (ver algoritmo no item 2.2.2), fica anulado se o mesmo bit entrar na aproximação da rotação dependente dos dados que lhe corresponde.

Considerando-se que aproximações da rotação dependente dos dados de 1 bit têm um bias fixo em 2^{-6} (ver definição 3.13), e considerando-se que a aproximação linear através da operação de adição de inteiros no bit menos significativo sempre vale com probabilidade 1 (e, portanto, com tendência 1/2), e levando em conta o lema do empilhamento (lema 3.2), conclui-se que a aproximação para $t = s = u = v = 0$ da equação 3.16 vale com uma tendência dada por:

$$2^{-1} \times 2^{-6} \times 2^{-1} \times 2^{-6} \times 2^{3} = 2^{-11} \tag{3.17}$$

Uma nova aplicação do lema do empilhamento permite concluir, por exemplo, que é possível montar uma aproximação linear para 6 rodadas do RC6 com tendência $2^{-33} \times 2^{2} = 2^{-31}$. Pode-se generalizar esta idéia, construindo aproximações para qualquer número par de rodadas do RC6.

Note-se que o valor 2^{-1} para a tendência da soma de inteiros sobre o bit menos significativo é o caso particular de um lema importante, o qual servirá de base para a principal conjectura do ataque de Knudsen e Meier [4]. Este teorema é reproduzido a seguir, para futura referência.

Lema 3.3 *Dado $y = x + a$ para alguma palavra de 32 bits fixa a e valor de entrada*

variável x, seja α_i a tendência da aproximação linear $y \cdot e_i = x \cdot e_i$, para $0 \leq i \leq 31$, calculado como uma média sobre todos os possíveis valores de a. Então:

$$\alpha_i = \begin{cases} 1/2 & \text{se } i = 0, \\ 1/4 & \text{caso contrário.} \end{cases}$$

A equação 3.17 corresponde ao valor da tendência no caso particular mostrado no lado direito da tabela 3.11. Levando em consideração o teorema acima, nota-se que a mesma equação corresponderia a uma tendência de 2^{-12} se um dos valores de u ou v (lado esquerdo da tabela 3.11) fosse igual a zero e o outro fosse **não-nulo mas menor do que 5** e corresponderia a uma tendência de 2^{-13} quando tivéssemos $1 \leq u, v \leq 4$.

Como um último refinamento a ataques do tipo I, Contini *et al.* consideraram generalizações do conceito de aproximações lineares, como já haviam feito no ataque diferencial, com o conceito de "diferenciais" generalizando o conceito de "características" (ver seção 3.1). No caso linear, esta generalização consiste em considerar todas as possíveis "trilhas" que ligam o início e o fim de uma aproximação. Isto pode ser feito permitindo que os bits iniciais e finais variem, levando ao conceito de **aproximações múltiplas**, ou mantendo os bits iniciais e finais fixos e apenas buscando todas as possibilidade de ligá-los com trilhas, levando ao conceito de **fecho linear** - note-se que o conceito de "fecho linear" é um perfeito análogo ao conceito de "diferenciais". Assim como as diferenciais intensificam o efeito das características de diferenças, existem estudos que mostram que os fechos lineares representam uma melhora significativa sobre aproximações lineares simples.

A tabela 3.12 mostra os resultados de Contini *et al.* para todos estes ataques sobre o RC6. Todos estão fundamentados na aproximação linear simples descrita acima. A principal conclusão que se pode tirar desta tabela é que, em sua versão mais sofisticada, considerando o fecho de uma aproximação linear - em termos práticos, é muito difícil implementar a utilização do fecho linear em um ataque -, a criptoanálise linear exige menos esforço do que força bruta para quebrar um RC6 com até 16 rodadas. É justamente esta observação que levou os autores do RC6 a

fixar o seu número de rodadas nominal em 20 rodadas.

	Número de rodadas				
Variante	8	12	16	20	24
RC6 *Ataque Linear Basico*	2^{62}	2^{102}	$\mathbf{2^{142}}$	$\mathbf{2^{182}}$	$\mathbf{2^{222}}$
RC6 + *Aproximações Múltiplas*	2^{51}	2^{91}	$\mathbf{2^{131}}$	$\mathbf{2^{171}}$	$\mathbf{2^{211}}$
RC6 + *Aproximações Múltiplas + Linear Hulls*	2^{47}	2^{83}	2^{119}	$\mathbf{2^{155}}$	$\mathbf{2^{191}}$

Tabela 3.12: Estimativas para as quantidades de textos necessárias para montar um ataque criptoanalitico linear sobre o RC6.

Capítulo 4

Ataques estatísticos a cifradores em bloco

Os ataques estatísticos se configuram em um modo conveniente de detectar fraquezas na não-linearidade dos cifradores em bloco: cifra-se uma certa quantidade de textos e deixa-se a estatística decidir se os dados cifrados apresentam ou não alguma regularidade que possa ser utilizada para recuperar a chave. A principal ferramenta estatística usada como instrumento de criptoanálise é o teste de aderência χ^2. O método χ^2 consiste em *manipular textos em claro* de tal modo que o valor do χ^2 associado a uma parte dos respectivos textos cifrados com determinado cifrador assuma um valor significativamente alto. É claro que, se for possível concluir que um conjunto de dados cifrados não é uniforme, uma fraqueza importante terá sido detectada.

O ataque χ^2 foi originalmente proposto por S. Vaudenay como um ataque ao DES [28], e Handschuh e Gilbert aplicaram a idéia ao cifrador SEAL [29]. Ele também já foi aplicado ao RC5 ([30] e [31]) e, principalmente, ao RC6 (ver seção 4.2 abaixo). O ataque χ^2 pode ser usado tanto para distinguir textos cifrados de números gerados aleatoriamente quanto para implementar ataques de recuperação de chave.

A primeira parte deste capítulo trata, inicialmente, da utilização

da estatística χ^2 para distinguir uma distribuição desconhecida de uma distribuição uniforme. Em seguida, é brevemente discutido o conceito de intervalo de confiança, o qual será decisivo para justificar as inferências para grande número de rodadas, com base em resultados para pequeno número de rodadas, que serão feitas nos caps. 5 e 6. O capítulo segue com a descrição de importantes ataques criptoanalíticos contra o RC6: o ataque estatístico de Gilbert *et al.* e o ataque "com correlações" de Knudsen e Meier. Os resultados do presente trabalho constituem-se, essencialmente, em extensões do trabalho de Knudsen e Meier, mas o trabalho de Gilbert *et al.* contém importantes elementos teóricos sobre os ataques estatísticos ao RC6.

4.1 Fundamentos Estatísticos

4.1.1 O teste de aderência χ^2

O teste estatístico χ^2 é comumente usado para comparar a distribuição de freqüências real, em dados observados experimentalmente, com aquela que seria esperada. Neste trabalho, ele será utilizado para tentar detectar uma distinção entre dados cifrados com o cifrador RC6 (ou com o MRC6) e dados produzidos por um gerador pseudo-aleatório. O χ^2 servirá, portanto, para verificar se a distribuição de freqüências de um certo volume de dados cifrados com o RC6 (ou com o MRC6) pode ser distingüida de uma distribuição de freqüências *uniforme*. Neste caso, é claro que o fato do χ^2 acusar não-uniformidade em dados gerados por algum cifrador significa que alguma "fraqueza" foi identificada.

Especificamente, o teste χ^2 vai servir para detectar alguma correlação entre *sub-blocos* dos dados de entrada ao RC6 com um número de rodadas reduzido e sub-blocos correspondentes nas saídas respectivas. O teste χ^2 é rapidamente descrito a seguir, com base no resumo de [4] e também no livro eletrônico de Lane *et al.* [32].

Sejam $X_0, X_1, \ldots, X_{n-1}$ variáveis aleatórias independentes e identicamente distribuídas, assumindo valores no conjunto de categorias $\{a_0, a_1, \ldots, a_{m-1}\}$,

de acordo com uma distribuição de freqüências desconhecida. Então o teste χ^2 serve para decidir se uma dada observação $X_0, X_1, \ldots, X_{n-1}$ é consistente com a seguinte hipótese (a chamada "hipótese nula"):

$$Pr\{X = a_j\} = p(j) \text{ para } 0 \leq j < m \tag{4.1}$$

Ou seja, para decidir se os valores das probabilidades associadas às variáveis $X_0, X_1, \ldots, X_{n-1}$ se distribuem sobre os m valores $\{a_0, a_1, \ldots, a_{m-1}\}$ de acordo com uma certa distribuição dada, $p(j)$.

Em um caso geral, a estatistica χ^2 é definida como segue.

Definição 4.1 *Seja* $N_{a_j}(X_0, X_1, \ldots, X_{n-1})$ *o número de vezes que a observação* $X_0, X_1, \ldots, X_{n-1}$ *assume o valor* a_j*. Então a* ***estatística*** χ^2 *é definida como:*

$$\chi^2 = \sum_{j=1}^{m} \frac{[N_{a_j}(X_0, X_1, \ldots, X_{n-1}) - n.p(j)]^2}{n.p(j)} \tag{4.2}$$

Note-se que $\sum_j N_{a_j}(X_0, X_1, \ldots, X_{n-1}) = n$.

A distribuição Qui-Quadrado é a distribuição da soma de *desvios normais padrão* - um desvio normal padrão especifica quantos desvios padrão um valor amostrado está distante da média da sua população - elevados ao quadrado. O número de graus de liberdade de uma distribuição é igual ao número desvios normais padrão sendo somados. Logo, uma distribuição Qui-Quadrado com um grau de liberdade, escrita como $\chi^2(1)$, corresponde simplesmente à distribuição de um único desvio normal elevado ao quadrado. Desta forma, a área de uma distribuição χ^2 que corresponde a valores menores do que 4, por exemplo, é o mesmo que a área de uma distribuição normal padrão para valores menores do que 2.

A distribuição Qui-quadrado pode ser associada ao seguinte problema: suponha que sejam amostrados dois valores a partir de uma distribuição normal padrão e que os seus quadrados sejam somados. Então, qual é a probabilidade de que a soma destes dois quadrados seja, por exemplo, igual ou maior do que 6? A resposta pode ser encontrada com o uso da distribuição Qui-Quadrado com

dois graus de liberdade. Ela nos informa que a probabilidade de um valor de χ^2 (com 2 GL) ser maior ou igual a 6 é 0.05.

Em um teste χ^2, a estatística χ^2 medida sobre os dados é comparada a $\chi^2_{a,m-1}$, o qual é denominado de *limiar* para o teste χ^2 com $m-1$ graus de liberdade e com nível de significância a.

Quando se trata especificamente de testar a aderência a uma distribuição *uniforme*, a expressão acima pode ser simplificada para:

$$\chi^2 = \frac{m}{n}\sum_{j=1}^{m}[N_{a_j}(X_0, X_1, \ldots, X_{n-1}) - \frac{n}{m}]^2 \tag{4.3}$$

Nas investigações realizadas nos caps. 5 e 6, vamos precisar especificamente dos limiares mostrados na tabela 4.1.

Tabela 4.1: Limiares χ^2 para alguns valores de nível de confiança e de graus de liberdade.

χ^2		Graus de liberdade (GL)		
		2^8-1	$2^{10}-1$	$2^{16}-1$
Nível de Confiança (C)	0.50	254.33	1022.30	65534.33
	0.75	269.85	1053.10	65779.19
	0.90	284.34	1081.40	65999.39
	0.95	293.25	1098.50	66131.63

Estatisticamente, diz-se que, à medida que n se torna suficientemente grande, a distribuição da quantidade dada pela equação 4.3 se aproxima da distribuição de uma variável aleatória χ^2, com $m-1$ graus de liberdade. Na prática, o valor 269.85, para 255 graus de liberdade, significa que, para n grande, a quantidade medida pela equação 4.3 vai ultrapassar 269.85 apenas 25% do tempo, desde que a distribuição associada às observações $(X_0, X_1, \ldots, X_{n-1})$ seja efetivamente uniforme.

4.1.2 Intervalos de Confiança para a Média

Quando se computa um intervalo de confiança, computa-se a média de *uma amostra* com o objetivo de estimar a média da população. Esta estimativa vai

depender da média da amostra e do erro padrão desta média. Este item apresenta, a partir do livro-texto eletrônico de Lane *et al.* [32], o modo como estes valores são obtidos e como são utilizados para construir intervalos de confiança.

4.1.2.1 Média e desvio padrão populacionais conhecidos

Inicialmente, vamos assumir que são conhecidos a média e o desvio padrão da população. É claro que, se a média populacional já fosse conhecida, não haveria necessidade de um intervalo de confiança. No entanto, assumir que são conhecidas as características da população ajuda a entender como os intervalos de confiança são construídos e com que objetivo.

Antes de tratar de uma amostra, é preciso imaginar muitas amostras de mesmo tamanho sendo selecionadas de uma população. Uma distribuição construída com as freqüências relativas das médias destas amostras é denominada de **distribuição amostral da média**. As características desta distribuição são extremamente importantes para a construção de um intervalo de confiança. A **média da distribuição amostral da média**, $\mu_{\overline{X}}$, é dada pela média da população de onde os valores foram amostrados (μ), ou seja:

$$\mu_{\overline{X}} = \mu \tag{4.4}$$

Já a **variância da distribuição amostral da média** é computada como:

$$\sigma^2_{\overline{X}} = \frac{\sigma^2}{N} \tag{4.5}$$

ou seja, ela corresponde à variância populacional dividida pelo tamanho da amostra, N (o qual também é a quantidade de valores usados para computar cada média). Logo, quanto maior o tamanho da amostra, menor a variância da distribuição amostral da média correspondente.

NOTA: É fácil mostrar que esta expressão decorre da lei da soma de variâncias, que estabelece que a soma (e a diferença) das variâncias de duas variáveis independentes é igual à soma das variâncias destas variáveis. Para isto, tome-se a distribuição amostral da soma de N números amostrados de uma população com

variância σ^2. Pela lei da soma das variâncias, a variância desta distribuição seria dada por $N \times \sigma^2$. Daí, uma vez que a média vale $1/N$ vezes uma soma de variáveis, a variância da distribuição amostral da média deve valer $1/N^2$ vezes a variância desta soma - propriedade da variância -, o que leva a σ^2/N.

O **erro padrão da média** é o desvio padrão da distribuição amostral da média, ou seja, ele é dado por:

$$\sigma_{\overline{X}} = \frac{\sigma}{\sqrt{N}} \tag{4.6}$$

O erro padrão é representado por um σ porque ele é, antes de mais nada, um desvio padrão. O subscrito $\overline{X}$ indica que o erro padrão em questão é o erro padrão da média.

O resultado teórico que fornece o fundamento para os intervalos de confiança é o teorema a seguir.

Teorema 4.1 *Teorema do Limite Central: Dada uma população com uma média finita μ e uma variância finita não-nula σ^2, a distribuição amostral da média se aproxima de uma distribuição normal com média μ e variância σ^2/N à medida que N, o tamanho das amostras desta distribuição, aumenta.*

NOTA: as expressões para a média (equação 4.5) e para a variância (equação 4.6) da distribuição amostral da média não são novas e nem são surpreendentes. O que é realmente notável é que, *independente da forma da distribuição da população "pai"*, a distribuição amostral da média se aproxima de uma distribuição normal à medida que N aumenta.

Em vista disto, conclui-se que, quando a média μ e o desvio-padrão σ de uma população são conhecidos, computa-se um intervalo com $(C * 100)\%$ de confiança para μ a partir de uma amostra de tamanho N desta população da seguinte forma:

$$\text{limite inferior} = \overline{X} - z_C \times \sigma_{\overline{X}} \tag{4.7}$$

$$\text{limite superior} = \overline{X} + z_C \times \sigma_{\overline{X}} \tag{4.8}$$

aonde z_C é o número de desvios padrão requeridos em torno da média de uma distribuição normal para conter $(C * 100)\%$ da área e $\sigma_{\overline{X}}$ é o erro padrão da média.

4.1.2.2 Média e desvio padrão populacionais desconhecidos

A fórmula anterior para o cálculo de um intervalo de confiança exige o conhecimento da média (μ) e do desvio padrão da população (σ), o que, normalmente, não é realístico, apesar de ajudar a compreender a origem dos intervalos de confiança. Quando nem a média e nem a variância são conhecidos, mas precisam ser estimados dos dados, a fórmula para o intervalo de confiança passa a depender do *estimador* para o erro padrão ($s_{\overline{X}}$), da seguinte forma:

$$\text{limite inferior} = \overline{X} - z_C \times s_{\overline{X}} \tag{4.9}$$

$$\text{limite superior} = \overline{X} + z_C \times s_{\overline{X}} \tag{4.10}$$

aonde o estimador do erro padrão é computado, a partir do estimador para o desvio padrão da população, s, como:

$$s_{\overline{X}} = \frac{s}{\sqrt{N}} \tag{4.11}$$

e aonde o estimador do desvio padrão é calculado diretamente sobre a amostra coletada, a partir de:

$$s = \sqrt{\frac{\sum_i (X_i - \overline{X})^2}{N - 1}} \tag{4.12}$$

NOTA: o denominador desta fórmula corresponde ao número de graus de liberdade, o qual é dado por "tamanho da amostra menos um" para levar em conta o fato de que as estimativas de média e de desvio padrão não são independentes, já que os mesmos dados servem para estimar as duas características. Normalmente, o desvio padrão é calculado sobre N porque se considera que o número de graus de liberdade de uma amostra corresponde ao seu tamanho. Porém, quando o cálculo de s é feito sobre uma *estimativa da média*, deve-se considerar um grau de liberdade a menos no cálculo de s, o qual deve ser feito sobre $N - 1$.

4.1.2.3 Intervalos construídos a partir de amostras pequenas

Ficou faltando um último ajuste na fórmula do intervalo de confiança para que ela possa ser usada nos casos que serão apresentados nos capítulos 5 e 6: é preciso considerar que *as amostras serão pequenas.*

Quando o tamanho da amostra é pequeno (menor do que 30), deve-se utilizar a distribuição t de Student, em vez da distribuição normal padrão. Quando o tamanho da amostra é grande (maior do que 30), a distribuição t é muito similar à normal e não é preciso se preocupar com este detalhe. Porém, com tamanhos de amostra menores, a distribuição t fica mais "achatada" (ou "leptocúrtica"). Por exemplo, com uma distribuição normal, 95% da distribuição está dentro dos limites de 1.96 desvios para mais e para menos a partir da média. No caso da distribuição t, se tivermos um tamanho de amostra de apenas 5, teremos que 95% da área fica nos limites de 2.78 desvios a partir da média. Portanto, neste caso, o erro padrão da média teria que ser multiplicado por 2.78 em vez de 1.96. Os valores de t a serem usados em um intervalo de confiança podem ser obtidos de uma tabela da distribuição t, para um número de graus de liberdade dado por $N - 1$.

Assim, a fórmula do intervalo de confiança, considerando uma amostra pequena, e média e desvio padrão populacionais estimados somente com base nesta amostra, é dada por:

$$\text{limite inferior} = \overline{X} - (t_C) \times \frac{s}{\sqrt{N}} \tag{4.13}$$

$$\text{limite superior} = \overline{X} + (t_C) \times \frac{s}{\sqrt{N}} \tag{4.14}$$

aonde $\overline{X}$ é a média de *uma* amostra, N é o tamanho desta amostra, t é valor da distribuição de Student com $N-1$ graus de liberdade para o grau de confiabilidade (C) desejado e s é o desvio padrão da amostra (= estimador do desvio da população), dado pela equação 4.12.

4.2 Principais ataques estatísticos ao RC6

Entre os diversos trabalhos relacionados com ataques estatísticos ao RC6 que foram publicados, existem dois que resumem a essência do método: o "ataque estatístico" de Gilbert *et al.* [33] e as "correlações no RC6" de Knudsen e Meier [4]. Grande parte dos demais trabalhos sobre este tema consistem em complementos ou variações de [4].

Knudsen e Meier [4] adotam um ponto de vista mais prático, procurando inferir o que ocorre com um grande número de rodadas de um RC6 com blocos de 128 bits (e chaves de 128, 192 e 256 bits), com base em resultados experimentais para poucas rodadas e também para blocos menores (32 e 64 bits). Já Gilbert *et al.* [33] apontam um evento probabilístico capaz de permitir vazamento de informação durante a cifragem do RC6 (muito similar à "brecha linear" aproveitada em [4]) e a partir disto desenvolvem, com base em algumas poucas "hipóteses heurísticas", uma análise teórica sobre o impacto de um ataque que explore este evento.

De maneira geral, o ataque de Gilbert *et al.* é bem mais amplo do que o ataque de Knudsen e Meier. Mas o ataque descrito em [4] acabou sendo a referência em relação ao tema, possivelmente pelo fato da ferramenta χ^2, usada em [4] para classificar certos sub-blocos de textos cifrados como associados ou não a uma distribuição uniforme, permitir uma análise mais simples e direta dos resultados, expondo de maneira muito clara a relação existente entre fraquezas lineares e a eficácia dos testes estatísticos. Já a analise feita em [33] produz estimativas de complexidade sem usar o χ^2, mas apenas trabalhando com as distribuições prováveis para os bits de certos sub-blocos da diferença (módulo 2^{32}) entre saída e entrada. O resultado de Gilbert *et al.* é mais amplo e mais eficiente do que o de Knudsen e Meier, sendo que este último pode ser considerado um caso particular do primeiro, porém, para os propósitos do presente trabalho, é melhor partir do ataque de Knudsen e Meier, utilizando [33] como complemento para as justificativas teóricas.

Esta seção trata justamente de discutir os dois ataques mencionados acima.

4.2.1 Criptoanálise χ^2 de Knudsen e Meier

No ano de 2000, Knudsen e Meier [4] apresentaram um artigo no "7th Fast Software Encryption" que descreve o uso do teste de aderência χ^2 como uma ferramenta de ataque ao cifrador RC6. Naquele trabalho, intitulado de "Correlations in RC6", os autores mostram que é possível distinguir versões reduzidas do RC6 com até 15 rodadas de uma permutação aleatória. Misturando resultados experimentais, algumas discussões teóricas e muitas justificativas práticas, eles mostraram exatamente como o χ^2 poderia ser usado para discernir entre um RC6 com número reduzido de rodadas e um gerador pseudo-aleatório. Eles não foram os primeiros a imaginar que a estatística χ^2 pudesse servir para atacar o RC6, dado que Baudron *et al.* [12] já haviam mencionado esta possibilidade em um relatório preliminar. Mas os ataques de distinção por eles descritos são mais eficientes do que os de Baudron *et al.* e eles também apresentam todos os detalhes de um ataque de *recuperação de chaves* contra o RC6.

No trabalho de Knudsen e Meier, o papel do χ^2 consiste em testar a distribuição dos valores de um certo subconjunto de bits das saídas de muitos textos cifrados com RC6 contra uma distribuição uniforme. Nos testes que eles realizaram, para RC6 com $\{2, 4, 6\}$ rodadas, o χ^2 detectava a não-uniformidade com esforço muito menor do que força bruta, sempre que uma quantidade suficiente de textos cifrados estava disponível.

O subconjunto do bloco de bits que serve de base para estes ataques foi identificado como sendo:

- para ataques de recuperação de chaves: os últimos 5 bits das palavras A e C do bloco de entrada (respectivamente, 1a e 3a palavras entre as 4 que compõem o bloco)

- para ataques de distinção: os últimos 5 bits das palavras B e D do bloco de entrada (respectivamente, 2a e 4a palavras)

A origem desta escolha vem, segundo os autores, de observações retiradas da parte

de criptoanálise linear do relatório sobre a segurança do RC6 elaborado por Contini *et al.* (ver item 3.2.2.1). Basicamente, Contini *et al.* mostram que a aproximação linear descrita por:

$$(A \cdot e_t) \oplus (C \cdot e_s) = (A'' \cdot e_u) \oplus (C'' \cdot e_v) \tag{4.15}$$

aonde A'' e C'' são a primeira e a terceira palavras depois de duas rodadas de encriptação usando o RC6, apresenta uma tendência significativa ($= 2^{-11}$) para $t = s = u = v = 0$, no caso de ambas as rotações envolvidas serem nulas (o que ocorre com probabilidade 2^{-10}). Além disto, se t, s, u e v forem não-nulas, mas menores do que 5, o valor da tendência, na mesma situação (rotações nulas), diminui, mas ainda continua significativo.

Com esta observação em mente, Knudsen e Meier procedem a investigação do que ocorre com a *concatenação dos últimos 5 bits de A e C* após duas rodadas de cifragem com o RC6. Eles denominam este sub-bloco de 10 bits de X. Portanto, X é formado pelos bits que estão definidos como "y", ao final das duas rodadas mostradas na figura 4.1. Eles mostram, então, como esta observação explica o fato dos bits de X ficarem distribuídos de maneira não-uniforme ao final das mesmas duas rodadas.

4.2.1.1 Explicação para o surgimento de não-linearidade

É necessário comentar apenas o caso de rotação nula, pois os outros casos (rotações de 1, 2, 3 e 4 bits) são semelhantes. Então, impondo-se que os últimos 5 bits de A e C mostrados na figura 4.1 sejam nulos - note-se que o que importa é que eles são **constantes** - e que (por acaso) os últimos 5 bits de t e u (ver figura 4.1) tenham resultado nulos, os valores de ambas as rotações dependentes dos dados na primeira rodada (simbolizados pelas seqüências "*bbbbb*" na figura) também serão nulos. Em conseqüência disto, os últimos bits de A e de C não serão alterados nem pelo ou-exclusivo e nem pelas rotações dependentes dos dados, às quais eles serão submetidos logo ao entrar no cifrador. Assumindo-se ainda (sem perda de generalidade) que os últimos 5 bits de $S[2]$ e $S[3]$ também são nulos, a conseqüência

imediata é que os últimos 5 bits de A e de C *não serão alterados pelo algoritmo*. Ou seja, seguindo este raciocínio, se submetermos um grande volume de dados a esta simplificação do RC6 (com duas rodadas), será possível notar uma nítida tendência a não-uniformidade se focarmos a atenção sobre os últimos 5 bits de A e C.

O que importa nesta análise é que a chance disto acontecer corresponde exatamente à chance dos bits que definem a rotação na primeira rodada (bits "*bbbbb*" na figura) serem nulos mais a probabilidade dos bits de X (bits "*yyyyy*" na figura) serem nulos por acaso quando as rotações não forem nulas. Ora, assumindo-se que a função quadrática f se comporta como um gerador pseudo-aleatório, a chance dela produzir uma palavra de 32 bits com os 5 bits mais significativos iguais a zero é exatamente de 1 em 32, ou seja, 2^{-5}. Logo, a chance das duas seqüências *bbbbb* mostradas na figura 4.1 serem nulas simultaneamente é de 2^{-10}. Por outro lado, em função da análise de Contini *et al.*, é razoável assumir que, se as rotações deduzidas de t e de u forem maiores ou iguais a 5, as saídas A'' e C'' possam ser consideradas (aproximadamente) *aleatórias*. Com saídas aleatórias, a chance dos últimos 5 bits de A'' (por exemplo) serem (por acaso) exatamente iguais a zero é de 1/32. A chance dos últimos bits de t (por exemplo) coresponderem a um valor maior do que 5 é de 27/32. Juntando tudo isto, conclui-se que a chance dos últimos 5 bits das palavras A e C passarem por duas rodadas do RC6 inalterados é de, no mínimo (Ainda existem muitas outras maneiras destes bits se anularem, para valores de rotação entre 1 e 4):

$$2^{-10} + (\frac{27}{32} \times \frac{1}{32})^2 \simeq 2^{-10} + 2^{-10.5} \tag{4.16}$$

Isto mostra claramente que a distribuição de X é tendenciosa (não-uniforme), pois, no caso uniforme, o valor desta probabiliade deveria ser exatamente de 2^{-10}. A diferença de $2^{-10.5}$ aparece na forma de valores de χ^2 muito acima do limiar correspondente à uniformidade em uma distribuição qualquer.

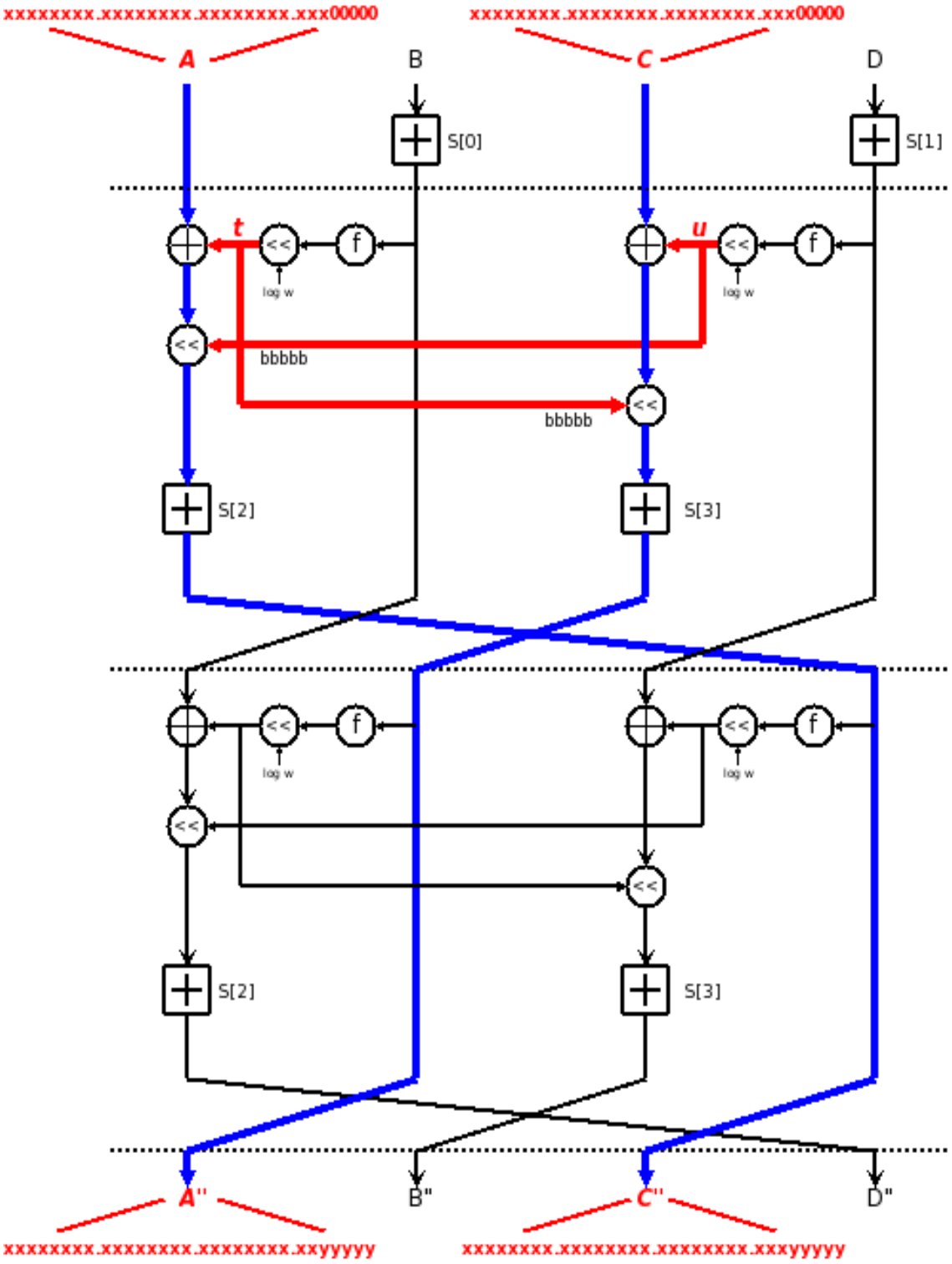


Figura 4.1: Representação esquemática da fraqueza linear explorada por Knudsen e Meier em seu ataque χ^2 a um RC6 reduzido para 2 rodadas.

4.2.1.2 Valores experimentais de χ^2

Knudsen e Meier passam então a uma demonstração experimental desta tendência. Para isto submetem grandes quantidades de textos (blocos e chaves de 128 bits) a versões reduzidas (com 2, 4 e, aonde foi possível, 6 rodadas) do RC6. Todos os blocos de 128 bits submetidos tinham em comum a característica de que os últimos 5 bits das palavras A e C eram nulos. Para cada conjunto de textos submetidos, eles medem o valor da estatística χ^2 (eq. 4.3) sobre as quantidades representadas pelos últimos 5 bits de A e de C após 2, 4 ou 6 rodadas do RC6.

Cada teste é realizado para 20 chaves escolhidas aleatoriamente, já que o algoritmo de cifragem depende da chave. Eles determinam, então, a quantidade de textos necessária para que o valor do χ^2 ultrapasse o limiar correspondente a 95% de confiabilidade *para todas as 20 chaves* Pela tabela 4.1, para 1023 graus de liberdade, este limiar vale 1098. Eles concluem que, para 2 rodadas, são necessários 2^{13} textos para que o χ^2 atinja um valor próximo a 1098 em todos os 20 testes realizados. Para 4 rodadas, Knudsen e Meier observaram que é necessário submeter 2^{29} textos a cifragem para que 20 testes diferentes ultrapassem o limiar de 1098. Daí, com o auxílio de simulações semelhantes para um RC6 com blocos de 64 bits, eles concluem que o log da quantidade de textos está crescendo linearmente em função do número de rodadas, ou seja, a cada duas rodadas o expoente da quantidade de textos cresce 16. Eles também "confirmam" esta hipótese com o auxílio das mesmas simulações sobre uma versão ainda mais enfraquecida do RC6 em que uma das rotações dependentes dos dados na primeira rodada é artificialmente anulada. Neste último caso, eles obtêm resultados para 2, 4 e 6 rodadas.

Assumindo-se, então, que esta "lei" de crescimento linear possa ser *extrapolada* para um número maior de rodadas, e misturando este ataque com um tratamento especial para as 3 primeiras rodadas, chega-se facilmente à conclusão de que a quantidade de textos necessária para detectar não-uniformidade na saída de um RC6 reduzido cresce de acordo com:

$$\#textos = 2^{13.8+r\times 16.2} \tag{4.17}$$

para um RC6 com $3 + 2 \times r$ rodadas.

Pela equação 4.17, este ataque consegue quebrar o cifrador RC6 com esforço menor do que força bruta se ele tiver até 15 rodadas. Para 16 rodadas, a quantidade de textos exigida (que vale $2^{119.1}$ excede a quantidade máxima de textos disponíveis (que é de 2^{128-10}). Para uma certa classe de chaves fracas, eles ainda conseguem estender este resultado para até 17 rodadas.

4.2.1.3 Ataques de recuperação de chave

Com base na fraqueza exposta pelo teste χ^2, Knudsen e Meier conseguiram ainda propor um ataque para recuperação de chaves. Baudron *et al.* [12] até já haviam comentado sobre a possibilidade de reforçar um ataque linear utilizando o χ^2, mas não chegaram a apresentar um algoritmo específico para usar esta idéia em um ataque para recuperação de chaves.

O ataque de recuperação de chaves de Knudsen e Meier parte do seguinte princípio: as fraquezas detectadas pelo χ^2 são resultado da fixação dos últimos 5 bits das palavras A e C em um grande volume de textos que vai ser submetido a um RC6 reduzido. Agora, supondo-se que, dentre estes textos, sejam utilizados apenas aqueles cuja palavra D gere rotações nulas - ou seja, gere valores de u cujos últimos 5 bits são iguais a 00000 - (ver figura 4.1), é de esperar que a fraqueza linear induzida pela fixação dos 10 bits na entrada *fique ainda mais evidente*. Com base nisto, pode-se montar um ataque para recuperar os 32 bits da sub-chave $S[1]$, da seguinte forma:

1. fixa-se (escolhe-se) $S[1]$ como uma das suas 2^{32} possibilidades

2. com o valor de sub-chave escolhido, é possível montar todas as 2^{27} possibilidades de D que, após passarem pela função f e sofrerem rotação de 5 bits, geram strings cujos últimos 5 bits são nulos

3. gera-se então vários blocos (128 bits) e seleciona-se somente aqueles cujo D é uma das 2^{27} possibilidades acima

4. manda cifrar, com o cifrador real e com a chave que se quer descobrir, somente os selecionados do item anterior

5. se a sub-chave escolhida era a certa, *a rotação vai mesmo dar zero* e o χ^2 resultante vai mesmo ser o maior de todos

    - se a chave escolhida não for a certa, *as rotações não vão se anular* e o valor de χ^2 correspondente tende a ser menor do que se elas se anulassem

6. repetir o ataque com outros conjuntos de textos, até que todos os falsos positivos sejam eliminados.

A complexidade de tempo deste ataque é maior do que a complexidade espacial porque envolve contadores para as 2^{27} possibilidades de chaves. Daí, levando em conta os detalhes de implementação do ataque acima e também experimentos específicos com rotação nula, além das extrapolações feitas para o ataque de distinção, Knudsen e Meier concluem que o seu ataque conseguiria recuperar a chave de um RC6 com r rodadas com um esforço da ordem de:

$$2^{21.7+r\times 8.1} \text{ unidades de trabalho} \tag{4.18}$$

Estas unidades de trabalho correspondem, aproximadamente, a um teste de chave. Com isto, se pode concluir que o máximo a que este ataque pode chegar é a quebrar 12 rodadas do RC6. Após este ponto, o esforço exigido fica maior do que a própria força bruta (ver tabela 10 em [4]).

4.2.2 O ataque estatístico de Gilbert *et al.*

No mesmo congresso em que Knudsen e Meier apresentaram o seu trabalho sobre "Correlações no RC6", o "Fast Software Encryption", um outro grupo de autores, Henri Gilbert, Helena Handschuh, Antoine Joux e Serge Vaudenay apresentou uma análise muito semelhante, mas adotando uma abordagem mais teórica e

melhor fundamentada. Os dois trabalhos estão fortemente relacionados. A tal ponto que é possível observar facilmente que o ataque delineado por Knudsen e Meier consiste na exploração de um caso particular de um teorema apresentado por Gilbert *et al.* em [33]. Este teorema merece, portanto, ser aqui discutido, pois as suas diversas possibilidades de ataque são importantes para entender como a variante do ataque de Knudsen e Meier descrita e implementada no cap. 5 pode realmente ser mais eficaz do que o ataque original.

4.2.2.1 O evento que permite vazamento de informação

Segundo a notação de [33], para $1 \leq i \leq r$ e $0 \leq j \leq 3$, seja $R_{i,j}$ o valor do registrador com índice j (aonde 0 corresponde à palavra A, 1 à palavra B, ...) após a i-ésima rodada de cifragem de um dado bloco, para uma chave específica S. Define-se $R_{0,j}$ como as palavras que servem de input para a primeira rodada. O índice j deve sempre ser tomado "módulo 4".

Gilbert *et al.* incluem a rotação fixa na definição da função quadrática, ou seja, para eles, afunção f é dada por:

$$f(x) = g(x) \bmod 2^w \ll log_2(w) \;=\; x.(2x+1) \bmod 2^w \ll log_2(w) \tag{4.19}$$

A figura 4.2 mostra a imagem de uma rodada do RC6 conforme a notação de Gilbert *et al.*

O ponto de partida é uma observação simples da estrutura de uma rodada do RC6, resumida no lema a seguir.

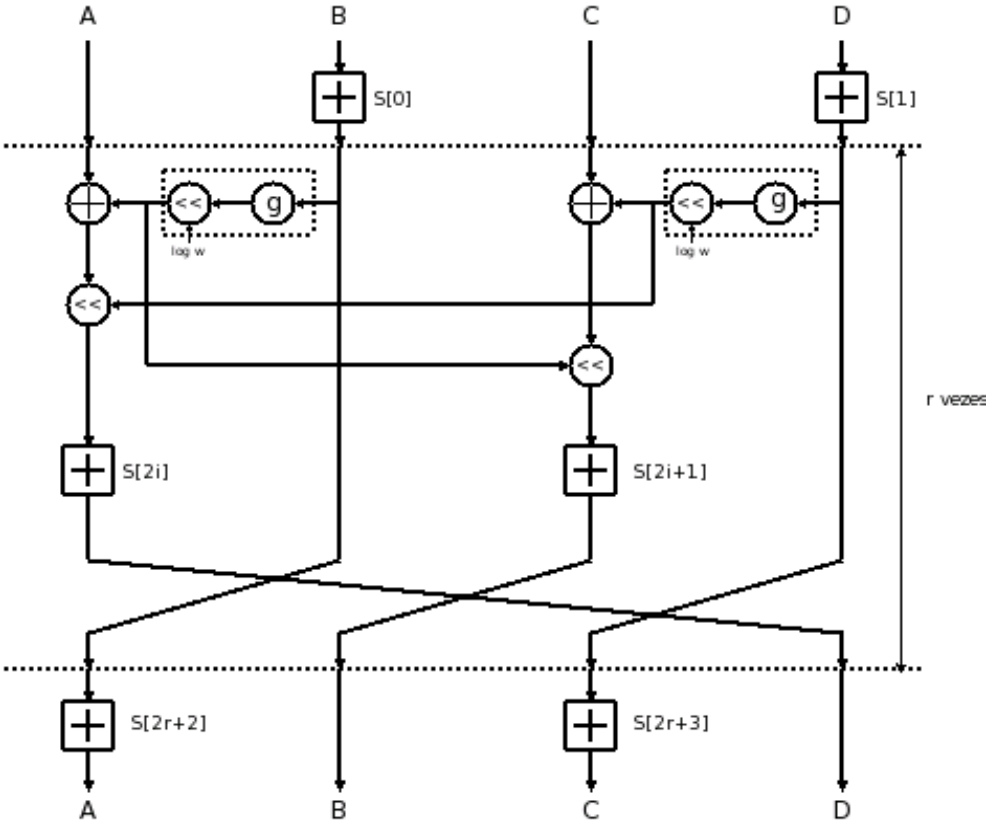


Figura 4.2: Cifragem com RC6-w/r/b, aonde $g(x) = x \times (2.x + 1)$.

Lema 4.1 *Para todo i e para qualquer chave e qualquer input, temos:*

- *SE:* $f(R_{i-1,1}) \equiv 0 \ (mod\ w) \quad E \quad f(R_{i-1,3}) \equiv 0 \ (mod\ w)$
- *ENTÃO:*

$$R_{i,3} - R_{i-1,0} \equiv S_{2i} (mod\ w) \quad E \quad R_{i,1} - R_{i-1,2} \equiv S_{2i+1} (mod\ w) \qquad (4.20)$$

- *Adicionalmente:* $R_{i,0} = R_{i-1,1}$ E $R_{i,2} = R_{i-1,3}$.

Este lema significa simplesmente que, se a parte "mod w" de $f(B)$ e de $f(D)$ são ambas zero na i-ésima rodada, então nada será somado (xor) sobre a parte "mod w" de A e de C e nenhum dos dois será rotacionado. O importante é que este fato se estende naturalmente para o seguinte:

Lema 4.2 *Se tivermos* $f(R_{i-1,1}) \equiv f(R_{i-1,3}) \equiv 0 \ (mod\ w)$ *para* $i = k, k+2, \ldots, k+2.\ell$*:*

- *Então:* $R_{k+2.\ell,2.\ell+3} - R_{k-1,0}(mod\ w)$ *E* $R_{k+2.\ell,2.\ell+1} - R_{k-1,2}(mod\ w)$

  são ***constantes*** *em relação ao texto, ou seja,* ***dependem somente da chave*** S*.*

O que isto quer dizer é que, se assumirmos que a função quadrática produz rotação nula a cada duas rodadas em um certo trecho da cifragem com o RC6, então a diferença entre certos sub-blocos de tamanho w na saída e outros na entrada deste trecho *não dependerá dos dados que estão sendo processados.*

Exemplo 4.1 *A figura 4.3 mostra esquematicamente o caso* $k = 2$*, ou seja, assume-se que as saídas das funções quadráticas são nulas nas rodadas 2, 4, 6,... Então, seja* $\ell = 1$*. Neste caso, temos que:*

- $(R_{4,1} - R_{1,0})\ mod\ w$ *e* $(R_{4,3} - R_{1,2})\ mod\ w$ *são constantes que dependem só de* S

- *Justificativa (ver figura 4.3):*

  - *se* $f(R_{1,1}) \equiv 0 \ (mod\ w)$*, então:* $(R_{2,3} - R_{1,0})\ mod\ w$ *depende só da chave*

  - *se* $f(R_{1,3}) \equiv 0 \ (mod\ w)$*, então:* $(R_{2,1} - R_{1,2})\ mod\ w$ *depende só da chave*

  - *se* $f(R_{3,1}) \equiv 0 \ (mod\ w)$*, então:* $(R_{4,3} - R_{3,0})\ mod\ w$ *depende só da chave*

  - *se* $f(R_{3,3}) \equiv 0 \ (mod\ w)$*, então:* $(R_{4,1} - R_{3,2})\ mod\ w$ *depende só da chave*

  - *adicionalmente:* $R_{3,2} = R_{2,3}$ *e* $R_{2,1} = R_{3,0}$ □

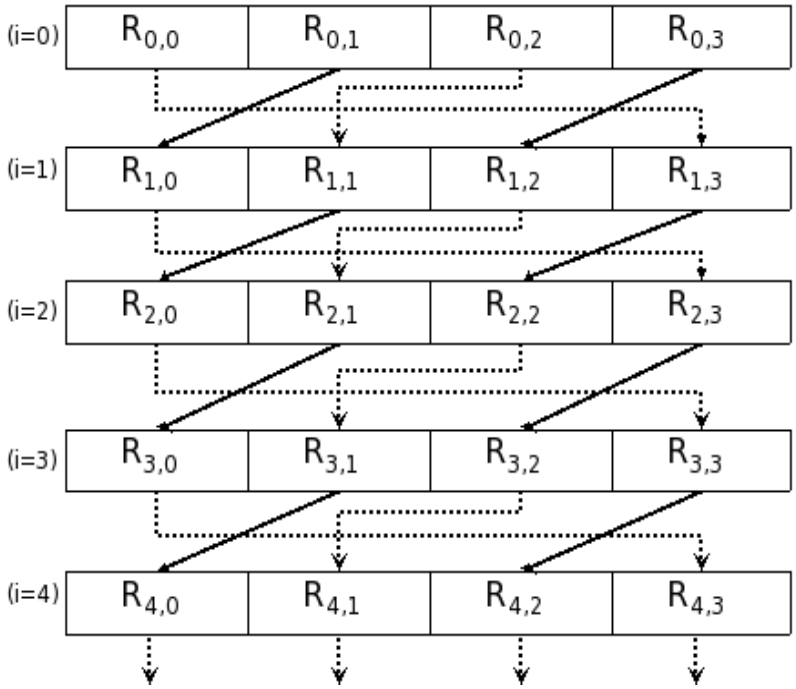


Figura 4.3: Representação esquemática da cifragem no RC6 - linhas contínuas ligam palavras que não são alteradas pelo processo.

Assumindo-se que as saídas de $f \ mod \ w$ se comportam como números aleatórios, este evento vale com probabilidade $w^{-2\ell}$, ou seja, $(1/w) \times (1/w)$ a cada duas rodadas. Chega-se, portanto, ao seguinte resultado heurístico, o qual foi confirmado experimentalmente por Gilbert *et al.*.

Teorema 4.2 *Sob hipóteses heurísticas, existem funções* $c_1(S)$ *e* $c_2(S)$ *tais que, para um* $(R_{0,0}, \ldots, R_{0,3})$ *e uma chave* S *aleatória, temos:*

$$Pr \left[\begin{array}{l} R_{r,1-r}(S) - R_{0,1}(S) \ mod \ w = c_1(S) \\ R_{r,3-r}(S) - R_{0,3}(S) \ mod \ w = c_2(S) \end{array} \right] \approx w^{-2.\lfloor \frac{r}{2} \rfloor}$$

Exemplo 4.2 *Este resultado vale tanto para números de rodadas pares quanto ímpares. Basicamente, nas rodadas pares ele aproveita o resultado do lema 4.2 e nas rodadas ímpares segue as palavras que não são alteradas (linhas contínuas na figura 4.3). Alguns exemplos:*

- *para* $r = 2$*:* $(R_{2,3} - R_{0,1})\ mod\ w$ *e* $(R_{2,1} - R_{0,3})\ mod\ w$ *são constantes*
- *para* $r = 3$*:* $(R_{3,2} - R_{0,1})\ mod\ w$ *e* $(R_{3,0} - R_{0,3})\ mod\ w$ *são constantes*

4.2.2.2 Ataque de distinção

Gilbert *et al.* propõem um ataque do tipo texto em claro conhecido baseado no resultado acima para distingüir entre o RC6 e uma permutação aleatória, que consiste, basicamente, em:

- cifrar muitos textos aleatórios com o RC6
- concatenar os últimos 5 bits da diferença entre as palavras B na entrada e na saída com os últimos 5 bits da diferença entre as palavras D na entrada e na saída
- determinar as freqüências de cada possibilidade dos 1024 valores correspondentes aos 10 bits acima
- se a freqüência máxima for maior do que um certo limiar t, a saída é 1, senão a saída é 0.

O valor de t é determinado com base na "vantagem" para distingüir o RC6 de uma permutação aleatória. Maximizando a diferença entre a probabilidade deste ataque fornecer 1 para o RC6 com a probabilidade dele fornecer 1 para uma permutação aleatória, e assumindo que, quando o número de textos é grande, cada contador acima tende a se comportar de acordo com a curva normal, Gilbert *et al.* chegam a um valor de freqüência individual que, se ultrapassado por qualquer um dos contadores, significa que a distribuição dos textos cifrados é distinta da uniforme.

Trabalhando com este resultado, Gilbert *et al.* concluem que o ataque deles é significativo para um número de rodadas dado por:

$$r \leq 2\lfloor \frac{4w - 7 - log_2 log(w)}{4.log_2(w)} + \frac{1}{2} \rfloor + 1 \tag{4.21}$$

de onde se tira que, com $w = 32$:

- obtém-se uma vantagem maior do que $1 - 2^{-60}$
- o que leva a uma complexidade de $n \approx 2^{20\lfloor \frac{r}{2} \rfloor - 2}$
- o que *mostra* que o ataque deles é capaz de quebrar até $r = 13$ rodadas (com $n = 2^{118}$)

4.2.2.3 Ataque de recuperação de chave

Gilbert *et al.* ainda apresentam um ataque de recuperação de chaves, baseado no ataque de distinção acima, sobre um RC6 com 14 rodadas:

- parte-se de muitos pares texto em claro-texto cifrado
- seleciona-se aqueles que levam a $f(B+S_0) \equiv 0 \bmod w$ e $f(D+S_1) \equiv 0 \bmod w$
- executa-se busca exaustiva sobre as chaves S_0 e S_1 para descobrir quais as subchaves que fariam com que a distribuição de freqüências de $(\Delta A, \Delta C)$ associada aos textos selecionados combinasse com a situação em que:
  1. não ocorrem rotações sobre A e C na primeira rodada
  2. as outras 13 rodadas seguem o ataque de distinção acima

Eles observam que, para $w = 32$ e $r = 14$, a quantidade de textos necessária é praticamente igual ao limite de 2^{128}.

4.2.3 Observações finais

Os ataques de Gilbert *et al.* exploram, essencialmente, as mesmas fraquezas do que os ataques de Knudsen e Meier. De um lado, Gilbert *et al.* fornecem um tratamento teórico mais rigoroso para o seu método. Por outro lado, ao utilizar a estatística χ^2, Knudsen e Meier utilizam argumentos estatísticos de uma maneira que pode mais facilmente comprovada com resultados experimentais. Ambos os trabalhos são complementares e resumem, de certo modo, a essência dos ataques estatísticos ao RC6.

Capítulo 5

Novas correlações no RC6

Durante a análise do artigo de Knudsen e Meier [4], diversos estudos foram conduzidos, na tentativa de responder às dúvidas que surgiram. Estes estudos geraram algumas observações que acabaram sendo úteis para a própria compreensão do artigo, assim como diversos resultados adicionais e até mesmo idéias para intensificar o efeito dos ataques descritos. Todos estes comentários adicionais foram testados sobre uma versão mais simples do RC6, o RC6-16/r/16, com blocos de 64 bits. Os resultados destes testes estão resumidos aqui neste capítulo.

5.1 Sugestão de mudanças ao ataque de Knudsen e Meier

Note-se que o ataque χ^2, na forma como é implementado no trabalho de Knudsen e Meier, *não representa* uma melhora sobre a criptoanálise linear de Contini *et al.* [2] que lhe serviu de fundamento. O ataque linear de Contini *et al.* consegue quebrar, a um custo menor do que força bruta, até 16 rodadas do RC6 com chave e bloco de 128 bits. Já o ataque de Knudsen e Meier só é efetivo contra um RC6 nas mesmas condições para, no máximo, 12 rodadas. Além deste limite, é preciso que a chave tenha mais do que 128 bits, a fim de aproveitar rotações nulas em B e em D.

Isto é surpreendente, pois o argumento de Knudsen e Meier é bastante consistente: em vez de *uma* falha linear, correspondente a *um bit* monitorado em A e um em C, o teste χ^2 aproveita *o grupo* de falhas correspondente a *5 bits* monitorados ao mesmo tempo. Esta característica indica que o teste χ^2 deva ter desempenho comparável diretamente com o "fecho" de uma aproximação linear (ver cap. 3), o que acaba não ocorrendo.

Esta observação sugere que o ataque χ^2 ainda possa ser melhorado, de modo a ter um desempenho, pelo menos, equivalente ao de uma criptoanálise linear sobre fechos lineares. Para ilustrar este ponto, nesta seção são apresentadas algumas modificações cujo efeito é verificado logo a seguir.

5.1.1 Na precisão do teste χ^2

Por que usar 95% de precisão no teste chi-quadrado? Knuth [34], ao discutir gerações de seqüências aleatórias de números, sugere que, dependendo das circunstâncias, valores menores de precisão já fornecem certeza suficiente de não-uniformidade em experimentos que envolvem dados gerados sob controle - em um computador, por exemplo. Uma exigência menor na precisão leva a uma menor quantidade de textos exigida para o ataque, pois sempre é preciso mais testes para se ter uma garantia maior na resposta de um teste χ^2. Considerando-se as complexidades típicas de um ataque deste tipo, qualquer diminuição na carga exigida pode significar a diferença entre obter ou não um resultado experimental para uma certa situação (um dado número de rodadas, por exemplo).

Porém, o principal objetivo de um ataque estatístico deste tipo a um cifrador é detectar não-uniformidade em dados por ele cifrados. Um cifrador consiste em um algoritmo, bem conhecido e bem controlado, de modo que não é de esperar que ocorram mudanças bruscas de comportamento após o estabelecimento de uma tendência de crescimento, por exemplo. Mais especificamente, os experimentos com baixos números de rodadas e com RC6 com blocos de 64 bits mostram que, uma vez que tenha sido considerada uma quantidade de textos suficiente para que

o valor do χ^2 ultrapasse o limiar correspondente a 75% de confiabilidade, *já não há mais risco das quantidades de textos superiores a esta produzirem um valor de* χ^2 *inferior a este limiar*. Ou seja, esta observação empírica permite assumir que, uma vez atingido o limiar de 75% de confiabilidade, não é mais necessário continuar aumentando o número de textos: a não-uniformidade já está detectada com garantia suficiente.

5.1.2 No critério de decisão para o teste χ^2

Embora isto não esteja explícito no texto de Knudsen e Meier, simulações numéricas mostram que o critério por eles utilizado para definir como "sucesso" a busca de não-uniformidade foi:

- "a quantidade de textos necessária para detectar não-uniformidade é aquela para a qual todos os testes, correspondentes às 20 chaves, produzirem um valor de χ^2 acima do limiar, o qual fica definido como o valor de χ^2 correspondente a 95% de certeza".

Conforme mostram os testes numéricos, pode-se avaliar que este critério tem dois problemas:

1. isto é muito exigente: muito antes de *todos* os 20 testes ultrapassarem o limiar, já há evidência estatística suficiente para concluir que a não-uniformidade foi detectada

2. ele não tem consistência estatística: o fato de todos os 20 testes ficarem acima do limiar garante, evidentemente, que a média *destes 20 testes* também ficará acima do mesmo limiar; só que isto *não é suficiente* para garantir que a média *real* (definida sobre todas as 2^{128} possibilidades de chaves) dos valores de χ^2 também estará acima do limiar; de fato, em muitos casos, mesmo com todos os 20 resultados de χ^2 acima do limiar, o desvio padrão correspondente pode ser tão grande que apenas uma parte do intervalo de confiança ultrapassa este limiar

5.1.3 Na geração de textos em claro

Por que os textos têm que ser aleatórios? Isto caracteriza um ataque com texto em claro escolhido. Mas a custoza geração aleatória de textos deste ataque poderia ser substituída por uma geração controlada e seqüencial de textos. Em um ataque com tal volume de processamento, qualquer ganho pode significar uma conclusão mais consistente. Por exemplo, com uma geração sistemática, seqüencial, dos textos em claro para serem testados, a cara geração aleatória passaria a um simples "+1". A conseqüência disto poderia ser um ganho em eficiência que pode significar a possibilidade de obtenção de resultados para uma maior quantidade de rodadas. De qualquer forma, uma demonstração de fraqueza de um cifrador em um esquema com textos escolhidos não pode ser ignorada na sua avaliação de segurança.

5.2 Novos resultados para o RC6 com $w = 16$ *bits*

Esta seção trata de uma implementação da modificação do ataque χ^2, descrita na seção anterior. Como objeto de teste, tomamos o cifrador RC6, mas com blocos de apenas 64 bits, ou seja, sub-blocos de 16 bits em vez dos 32 bits normais. Blocos menores permitem a obtenção de resultados para um número maior de rodadas, o que facilita a compreensão dos detalhes do ataque, quando colocado em prática. Ocorre que estes resultados são exatamente aqueles que foram utilizados para a elaboração de um artigo a respeito desta nova abordagem para o ataque χ^2.

5.2.1 Ataque χ^2 modificado

A discução anterior nos leva naturalmente a propôr uma revisão no ataque realizado por Knudsen e Meier, de maneira a incluirmos as seguintes modificações:

1. O desvio padrão é também computado em cada amostra dos valores χ^2 e a média χ^2 é reportada no contexto de intervalo de confiança de 69%: a condição

para detecção de não-uniformidade é que o limite inferior neste intervalo precisa estar acima do threshold; então, qualquer conclusão de que o threshould de 75% na distribuição padrão χ^2 (por exemplo) foi excedido pelos dados experimentais irá ter uma probabilidade de 85% (=70%+15% da parte superior da curva normal) de estar correto; foi observado experimentalmente que isto já tem significado suficiente para garantias para um algoritmo como o RC6;

2. textos em claro são gerados de maneira controlada (sistemática): esta mudança leva a uma versão enfraquecida do ataque original, mas permite maior impacto e conclusões mais fortes com o mesmo esforço computacional do que com geração aleatória, como é demonstrado na próxima sessão.

5.2.2 Resultados do ataque χ^2 modificado

Esta seção apresenta os resultados da aplicação do ataque χ^2 modificado descrito na seção 5.2.1 ao RC6-12. Todos os resultados foram obtidos com amostras de 20 chaves randomicas demonstradas na tabela 5.1.

key number:	value:	key number:	value:
0	81ed.1d0a.6756.6fb6	10	713b.effd.d789.5df7
1	3dc8.e054.49ed.30ac	11	277a.6d18.f138.2f28
2	a7aa.29ad.d583.ce91	12	6e3a.2dcd.dc1a.909e
3	b473.9d6d.ed5d.e193	13	5978.76fc.5889.3a7f
4	0be3.262a.adf4.3e9a	14	1d93.0cf6.7339.a000
5	d029.e67b.a88d.b3db	15	ab1f.0481.4f0d.bd38
6	140a.44d7.fe76.bc8a	16	fd49.61f8.7297.b00f
7	49aa.939b.b5de.7dac	17	a4e2.488f.332b.ae55
8	93fb.f72a.4252.7e0a	18	ba0a.9bb5.779e.0aef
9	84bf.4c2e.dad0.ab41	19	fe7a.1492.aca0.1c2b

Tabela 5.1: Valores das 20 chaves utilizadas nas simulações com o RC6-16

Inicialmente mostramos que nossas simulações são consistentes com

os resultados apresentados em [4]. A tabela 5.2 compara os resultados deles com os obtidos com o código utilizado neste trabalho, para 2 e 4 rodadas. Neste caso, os textos de entrada são gerados randomicamente e depois os quatro bits menos significativos (lsb_4) em A e em C são definidos como zero. O bloco de texto original é gerado por um gerador pseudo-aleatório, e cada um dos novos textos em claro é gerado pela encriptação do texto anterior com 20 rodadas do RC6. E como pode ser observado, os resultados são bem similares.

número de rodadas	número de textos	média χ^2	média χ^2 [4]	mínimo χ^2 no IC
2	2^{10}	275.9	283	270.9
2	2^{11}	311.2	308	304.4
2	2^{12}	360.0	364	353.4
4	2^{22}	287.8	286	283.4
4	2^{23}	321.0	318	315.6

Tabela 5.2: Dependência dos valores χ^2 pelo número de textos para 20 testes no RC6-16, com entrada aleatória e $lsb_4(A) = lsb_4(C) = 0000$

A tabela 5.3 mostra os resultados para o mesmo tipo de simulações feitas com o ataque revisado descrito na seção 5.2.1. Antes de mais nada, é evidente por estes resultados que uma vez que o limite inferior do valor do intervalo de confiança para a média χ^2 exceda 269 (o threshold de 75% de confiança com 255 graus de liberdade), não existe volta na tendência de crescimento . Outro aspecto importante dos resultados é a drastica redução no número de textos necessários para passar do threshold. Se os textos em claro são gerados de maneira controlada, os resultados podem ser *facilmente obtidos em até 8 rodadas*, necessitando de cerca de 2^{28} textos em claro para chegar ao threshould de 75%. De acordo com as inferências de Knudsen e Meier, a utilização de textos gerados randomicamente irá provavelmente demandar uma quantidade gigantesca como 2^{43} para chegar no mesmo nível.

número de rodadas	número de textos	média χ^2	mínimo χ^2 no IC
2	2^2	277.6	257.7
2	2^3	321.6	290.7
2	2^4	385.6	353.8
4	2^{10}	269.6	262.8
4	2^{11}	271.0	264.8
4	2^{12}	292.8	286.2
4	2^{13}	323.7	314.5
4	2^{14}	392.2	371.7
6	2^{18}	251.5	245.2
6	2^{19}	259.7	253.9
6	2^{20}	270.9	263.8
6	2^{21}	338.6	315.1
6	2^{22}	418.1	364.2
8	2^{25}	263.2	258.4
8	2^{26}	258.5	253.1
8	2^{27}	266.9	260.9
8	2^{28}	284.9	277.0
8	2^{29}	322.0	305.6
8	2^{30}	378.4	351.6

Tabela 5.3: Dependência dos valores do χ^2 com a quantidade de textos para 20 testes no RC6-16 com o input controlado

Acontece que, quando os textos em claro no ataque são gerados de maneira controlada, a maioria das vezes existe apenas uma mudança em *uma palavra* para outra dos 64-bits de um bloco de entrada para outro. O primeiro texto em claro em cada conjunto demonstrado na tabela 5.3 é sempre um bloco todo zero de 64 bits. Cada novo texto em claro é produzido pegando o anterior e adicionando um na palavra D. Então quando os 2^{16} textos em claro são gerados, o sub-bloco D

é zerado e um é adicionado no sub-bloco B. Outros blocos são gerados em D, até que um limite de 2^{32} textos em claro seja alcançado. Se mais textos em claro forem necessários as palavras A e C podem ser utilizadas para incrementar o número de possibilidades.

As linhas continuas grossas na figura 5.1 são o caminho percorrido pelas palavras que são completamente diferentes nos primeiros 2^{16} textos em claro gerados desta maneira, enquanto as linhas continuas finas representam palavras que nunca mudam para estes mesmos textos em claro. Pode ser observado que, se somente a diferença entre dois textos em claro se mantiver na palavra D, depois de uma rodada ainda há uma palavra inteira que não passou pela operação de encriptação. Seria completamente diferente se os textos em claro fossem gerados de maneira randômica. É claro que a geração controlada sobre C seria ainda melhor, já que, neste caso, somente uma palavra (a saber, B') teria mudado depois de uma rodada de encriptação. Mas a geração controlada sobre D deixa as palavras A e C completamente zeradas, garantindo que os últimos 4 bits menos significativos sejam zero, exatamente como o ataque χ^2 de Knudsen e Meier. Seria ainda possível gerar textos sobre C "da esquerda para a direita", deixando os 4 bits menos significativos como zero em C de qualquer maneira. Mas então, somente 2^{12} textos poderiam ser gerados depois envolvendo mudanças em outras palavras do texto em claro. Claramente, menos palavras envolvidas levam a uma maior não uniformidade a ser detectada pelo teste χ^2 na saída.

Os resultados obtidos com a entrada controlada representam um ganho impressionante sobre a entrada aleatória e bits fixos. Entretanto, uma nota de precaução precisa ser considerada. A seqüência natural para os resultados obtidos de longe poderia ser utilizada para uma extrapolação destes resultados para um maior número de rodadas, de maneira similar ao que foi feito por Knudsen e Meier com os resultados que eles obtiveram para o RC6-32. Por exemplo, a tabela 5.4 mostra que o número de textos necessários para se chegar ao valor mínimo de χ^2 por volta de 300 - este valor é escolhido porque existe muita variação nos valores obtidos. A primeira vista, os expoentes do número de textos parecem crescer linearmente, aumentando

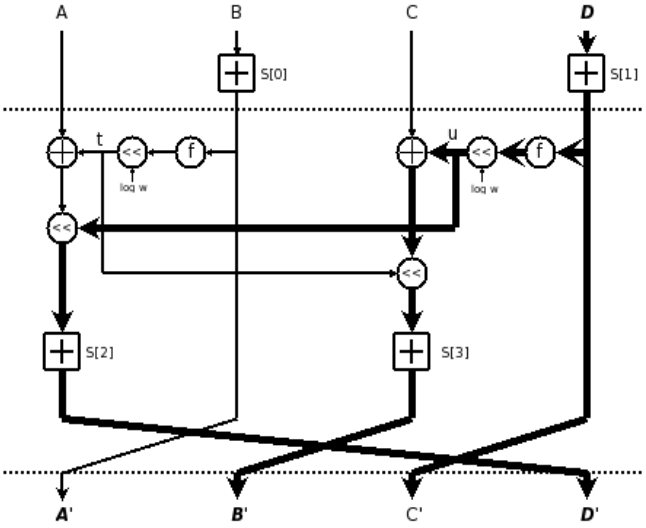


Figura 5.1: Efeito da mudança de somente uma palavra na encriptação com o RC6-$w/r/b$. As linhas mais grossas significam palavras diferentes no textos em claro

8 unidades para cada incremento de 2 no número de rodadas. Isto poderia levar a conclusão que, deixando pelo menos 8 bits (de um total de 64) definidos como zero, esta nova aproximação para o ataque χ^2 seria capaz de distinguir 14 rodadas do RC6-16 de uma geração aleatória, ao custo de 2^{53} encriptações. Entretanto, a idéia principal em usar entradas controladas é de manter os bits tão constantes quanto possível no conjunto de textos em claro enviados para o teste χ^2. Então, quando o número de textos aumentar e mais bits precisarem ser utilizados, não é possivel prever (sem experimentos) se a tendencia linear irá se manter.

5.2.3 Conclusões sobre as modificações

Nossos resultados mostram uma redução do número de texto em claro escolhidos (ver por exemplo as tabelas 5.3 e 5.4) necessários para que os valores de χ^2 excedam o threshold definido. De fato, o resultado tem um redução tão significante na complexidade do ataque χ^2 original contra o RC6-16 que nos não temos problemas em obter resultados experimentais em até 8 rodadas para o RC6-

número de rodadas	número de textos	mínimo χ^2
2	2^3	290.7
4	2^{13}	314.5
6	2^{21}	315.1
8	2^{29}	305.6

Tabela 5.4: Número de textos para o mínimo $\chi^2 \sim 300$.

16, enquanto que o ataque original dificilmente chega a 6 rodadas.

Entretanto, permanece obscura se a extrapolação dos resultados obtidos para o RC6-16 com menos rodadas pode ser extrapolada para um número maior de rodadas, como foi feito no caso do ataque original ao RC6-32 [4]. Acontece que a efetividade do ataque revisado depende em ter vários bits constantes no conjunto de textos em claro preparados para o ataque, uma situação que modifica-se com o aumento do número de rodadas e com o aumento do número de textos utilizados.

Capítulo 6

Criptoanálise χ^2 do cifrador MRC6

Para obtenção dos dados apresentados neste capítulo utilizamos um computador com um procesador AMD Athlon 64 X2 Dual Core 6000+ a 3000Mhz com 4G de memória RAM DDR2 a 800Mhz utilizadas com leitura em paralelo de duas placas de 2G, sendo os programas feitos em c e compilados com o gcc versão 4.1.2 20070925 em um linux com kernel Fedora Core release 8 (2.6.23.15-137) de 64 bits.

O processador utilizado foi AMD pela tendência histórica desta marca trabalhar com um pipeline otimizado em relação às concorrentes no mercado, o que poderia nos auxiliar já que lidamos com implementações desenvolvidas para 32 bits mas em processadores de 64 bits. Os algorítmos utilizados não foram otimizados de forma a trabalhar com o processador dual, sendo que os resultados que obtemos são normalmente referêntes à utilização de "um processador" a cerca de 99%, de modo que o sistema operacional e outras funções ficaram lidando com o "outro processador", ficando um dedicado ao processamento dos resultados que buscamos de modo a ser executado pelo usuário *root* com o comando *nice -n -20*.

Não lidamos com os novos processadores X4 pelo fato de terem sido lançados durante a redação desta dissertação e porque não estamos lidando com algoritmos otimizados, e nem mesmo que lidam com blocos de 64 bits. Quanto a memória RAM pelo fato de não sabermos a quantidade que seria necessária nos

atemos a DDR2 de 800Mhz da RAM em vez de tentarmos a utilização da DDR3 da placa de video, já a 1.7Ghz e com a possibilidade de utilização de duas em paralelo ou até mesmo 3 nas placas mãe lançadas durante este estudo, porém com quantidade limitada - no caso da RAM poderiamos expandir para até 8G se necessário.

Os tempos apresentados serão normalmente relativos a todos os testes realizados independente do número de textos gerado para o experimento, podendo o tempo para um número maior (ou menor) de textos ser calculado facilmente com base no fato de que para metade do número de textos teremos cerca de metade do tempo e para o dobro do número de textos teremos cerca do dobro de tempo.

Em alguns casos o tempo apresentado é de um experimento menor, calculado depois do processamento do experimento maior, mas que permite o cálculo do tempo que foi utilizado no experimento principal - neste caso os tempos estimados estarão apresentados em itálico.

É claro que existem "erros" de alguns segundos devido ao processamento e utilização do sistema e então quando de experimentos que envolvam poucos ou menos de um segundo, os mesmos não são utilizados como base para cálculo de tempo.

Inicialmente tinhamos em mente demonstramos os resultados de ataques realizados ao MRC6 com as definições padrão com blocos de texto em claro de 512 bits e demonstrarmos o comportamento previsto se mantendo para o cifrador com blocos de 256 bits, porém não foi necessária utilização do cifrador com um bloco menor, já que obtemos dados já bastante consistentes para a versão oficial do cifrador.

Quanto ao crescimento logarítmico verificamos que este *parece estabilizar* seu crescimento em um certo ponto mas não temos como assertir sobre o mesmo se estabilizar como uma reta quando lidando com os valores logarítmicos como demonstrado em [33] no ataque que também é considerado em [4] ao lidarmos com o cifrador com mais blocos, porém experimentalmente chegamos a considerações parecidas quando lidando com os dados para o teste χ^2 em até 10 rodadas, só que a partir deste ponto verificamos que seria melhor no caso de uma extrapo-

lação a consideração de um crescimento super-exponencial, como demonstraremos nas análises apresentadas a seguir. Ou seja, ao demostrarmos os dados para 11 e 12 rodadas mostramos que assumir que o comportamento se mantém como uma reta independente do número de rodadas não é algo seguro a se fazer quando lidando com cifradores em blocos, principalmente por sabermos que a iteração crescente com o passar das rodadas pode gerar comportamentos não previstos inicialmente. A idéia de extrapolação é matinda apenas por analogia ao ataque demonstrado em [4].

Ao final dos experimentos verificamos a necessidade para o caso considerado de apenas cerca de 8,192 MBytes de memória RAM (0,2% de 4GBytes), o que permite que com as novas placas de vídeo já com 1GByte de memória e a velocidades muito maiores que as obtidas com a RAM em DDR2 poderiamos obter até mesmo uma melhor performance mesmo com mais considerações dependentes de memória, sendo que o algorítmo ainda pode ser otimizado para a utilização do processador dual, ou mesmo dos novos quad, e mesmo sem nada disso chegamos a fazer verificações de até 2^{41} textos cifrados em um computador pessoal. Ou seja, sem a otimização da utilização de memórias e apenas otimizando o algorítmo para trabalhar com o processador dual poderiamos verificar até 2^{42} textos ou até 2^{43} textos com o processador quad em tempos relativamente pequenos para este tipo de verificações (cerca de duas semanas).

Acreditamos que com todas as otimizações possíveis, já que poderiamos utilizar a memória com 3 placas de vídeo acessadas em paralelo a velocidades de 2 GHz e otimizando para a utilização de um processador quad a 3 Ghz, chegariamos a fazer a análise de 2^{45} textos cifrados em perido de cerca de um mês. Ou seja, poderiamos realizar nossa análise do χ^2 para 32 chaves em até 2^{40} textos de entrada. O que é um fato muito interessante de ser verificado já que a poucos anos não teriamos como considerar um processamento deste tipo em computadores pessoais.

Como estas considerações são para um cifrador quatro vezes maior que o RC6 então poderiamos obter até mesmo mais resultados quando da análise do RC6 (4 vezes mais) ou do RC5. E esperamos que trabalhos futuros levem nossas

considerações de implementação em conta para a verificação de novas características em cifradores da linha RC em conjunto com nossas novas considerações de ataque para a verificação do teste χ^2.

6.1 Considerações sobre o MRC6 pertinentes aos novos ataques

Quando lidando com o cifrador em blocos MRC6 temos um cifrador com 512 bits de tamanho de bloco e sub-blocos de tamanho de 32 bits para a análise, quando da consideração do cifrador em sua definição padrão, e apesar de o mesmo pertencer a família dos cifradores RC5 e RC6 este é um cifrador com muito mais possibilidades para que possamos trabalhar, sendo um fato muito importante a existência de 16 sub-blocos e não apenas 4 como no caso do RC6, e com base em aspectos específicos do cifrador podemos fazer uma nova série de considerações, sendo também muito interessante o fato de que o comportamento para um baixo número de rodadas permite uma análise diferenciada neste cifrador. Sendo que o número de rodadas considerado baixo neste caso é muito maior do que um baixo número de rodadas para os cifradores da mesma família. Sem contar que ainda temos o fato de lidarmos com um cifrador de 512 bits com chave de 128 bits, o que permite muito mais considerações quando lidando com a complexidade do ataque.

Para início das considerações sobre o cifrador partimos do movimento interno dos sub-blocos. Nós não temos controle sobre a rotação interna de bits que o sub-bloco sofre a cada duas rodadas (a partir da segunda rodada) com informação provinda de outro sub-bloco, mas se considerarmos apenas uma rotação então ainda temos toda a informação anterior que tinhamos no sub-bloco. A cada rodada temos também a adição de sub-chave de rodada e uma mistura com o bloco à direita a cada duas rodadas (a partir da segunda rodada).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Início	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**
Rodada 1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—
Rodada 2	—	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—	—
Rodada 3	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—	—	—
Rodada 4	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—	—	—	—
Rodada 5	—	—	—	—	—	—	—	—	—	—	**XXX**	—	—	—	—	—
Rodada 6	—	—	—	—	—	—	—	—	—	**XXX**	—	—	—	—	—	—
Rodada 7	—	—	—	—	—	—	—	—	**XXX**	—	—	—	—	—	—	—
Rodada 8	—	—	—	—	—	—	—	**XXX**	—	—	—	—	—	—	—	—
Rodada 9	—	—	—	—	—	—	**XXX**	—	—	—	—	—	—	—	—	—
Rodada 10	—	—	—	—	—	**XXX**	—	—	—	—	—	—	—	—	—	—
Rodada 11	—	—	—	—	**XXX**	—	—	—	—	—	—	—	—	—	—	—
Rodada 12	—	—	—	**XXX**	—	—	—	—	—	—	—	—	—	—	—	—
Rodada 13	—	—	**XXX**	—	—	—	—	—	—	—	—	—	—	—	—	—
Rodada 14	—	**XXX**	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Rodada 15	**XXX**	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Tabela 6.1: Movimento de um sub-bloco com o passar das rodadas no MRC6

Porém podemos considerar estas transformações como transformações “aleatórias” feitas durante a passagem dos dados pelo cifrador e nos ater a informação de movimento de um sub-bloco com o passar das rodadas para percebermos que o mesmo leva 15 rodadas para passar por todos os pontos possíveis para um sub-bloco como apresentado na tabela 6.1.

Note que para o sub-bloco P “chegar” na posição do sub-bloco A o mesmo leva 15 rodadas, ou seja muito mais do que no caso do RC6, que leva apenas 3 rodadas para fazer o “movimento completo”. Esta observação pode parecer ínfima quando lidando com o cifrador com uma entrada de dados aleatória - como no caso do primeiro ataque a ser reproduzido -, mas como perceberemos nos novos ataques apresentados mais à frente pode ser uma característica com implicações muito sérias, especialmente quando lidarmos com o cifrador com um baixo número de rodadas e a entrada quase toda definida como zero.

Note também que já temos um indício que ao lidarmos com um único sub-bloco com os dados sendo alterados com o passar das rodadas e o restante das informações fixas ao fazermos uma analise estatística como a de χ^2 poderemos obter informações que diferenciem o cifrador de um conjunto de dados aleatórios.

Outro ponto fortíssimo é que poderiamos lidar com dois ou mais sub-blocos com a distância de um sub-bloco aparentemente sem problemas, ou seja, teriamos ainda 256 bits para trabalhar de maneira a ainda podermos obter bons resultados. Porém como a chave possui apenas 128 bits nem precisamos considerar um ataque com tantos textos em claro, e quanto menor o número de textos envolvidos existe a possibilidade de trabalharmos com um maior número de bits fixos na entrada, o que deve permitir resultados ainda melhores.

Como demonstraremos nas próximas sessões, onde são apresentados os ataques, para ataques menores que o tamanho definido em um sub-bloco quando do ataque com input controlado até mesmo as considerações sobre rotação de outros blocos não incomodam tanto quando lidando com menos de 16 rodadas. Como ao lidarmos com a versão de 512 bits não conseguimos passar de 11 rodadas (12 dependendo do ponto de verificação), devido ao processamento do computador

utilizado, o restante das verificações foram deduzidas com base nas informações obtidas até então, mas voltamos a frisar que considerações sobre o *comportamento se manter* com um número diferente de rodadas é sempre perigoso para cifradores em blocos e fazemos esta análise apenas por analogia ao artigo [4].

Um outro ponto relevante é que quando da consideração de todas as modificações no ataque original que podem levar a obtenção de bons resultados para o MRC6 levamos em consideração também o fato de que estamos lidando com um cifrador de 512 bits, e que em teoria cada bit da entrada afetaria todos os bits de saída, e apesar de estarmos medindo a variação do χ^2 a cada duas rodadas, por estarmos lidando com a idéia de que os sub-blocos que sofrem a análise voltam a ser os mesmos analisados, só que em posições diferentes, neste período indicado também apresentamos as informações para número de rodadas ímpares, de modo a demonstrarmos um estranho comportamento quando da análise principal apresentada no ataque principal deste estudo. Este comportamento se dá pelo fato de que na consideração principal do ataque os últimos cinco bits de todos os sub-blocos são definidos como zero, e como foi demonstrado no ataque realizado em [4] quando destas considerações é possível fazer a análise tanto nos blocos pares como nos blocos impares porém com ganhos um pouco diferentes, e sabemos que a soma da sub-chave não faz diferença quando da análise dos dados no teste χ^2.

Neste ponto frizamos também que a análise se manter no mesmo ponto (mesmos sub-blocos ao final da cifragem) tanto para rodadas pares como para rodadas ímpares apesar de parecer não ter *sentido bem definido*, já que estamos medindo sub-blocos de entrada diferentes, na verdade tem como base principal que estes "outros blocos" acabaram de sofrer um ou-exclusivo com os blocos analisados e então sofrem novas alterções ao passar pelo cifrador, e então utilizamos estes dados para acompanhar mais *de perto* o crescimento da complexidade do cifrador, já que pretendemos fazer estimatívas de crescimentos do algorítmo com o ataque segundo nossas novas definições, e nas rodadas pares pudemos obter bons resultados somente em até 10 rodadas de maneira a não termos segurança em fazer extrapolações a partir deste ponto.

De maneira objetiva, temos que ao fazermos a análise χ^2 a cada duas rodadas dos ultimos 5 bits dos blocos A, C, E, G, I, K, M e O, se analisarmos uma rodada a frente só que em B, D, F, H, J, L, N e P temos os mesmos resultados já que os blocos passam de forma "ilesa", mas se verificamos os últimos 5 bits nas rodadas ímpares em A, C, E, G, I, K, M e O estaremos fazendo na verdade a análise para os 5 últimos bits dos blocos B, D, F, H, J, L, N e P iniciais, e podemos então notar que no novo ataque considerado os bits analisados não fazem tanta diferença quanto no ataque que serviu de base para o mesmo, já que o crescimento logarítmico parece "estabilizar" no mesmo ponto, ou seja, abrimos possibilidade para novas considerações sobre o ataque original adaptado para o caso do MRC6.

Outro ponto importantíssimo é que temos com base nestas informações a explicação para o ganho das informações para o ataque χ^2 quando trabalhamos com os blocos A, C, E, G, I, K, M e O em vez dos blocos B, D, F, H, J, L, N e P diferenciando ainda mais a nossa concepção de ataque do ataque original. Porém o ponto que parece aumentar a dificuldade do ataque quando lidando com os blocos B, D, F, H, J, L, N e P, quando da adaptação do novo ataque se medirmos este valor para a próxima rodada, que seria o mesmo valor anterior, verificamos que os valores *parecem se encaixar* entre os valores para as rodadas pares dos blocos A, C, E, G, I, K, M e O.

Também não podemos deixar de relevar o fato de que os mesmos ataques realizados para chaves de 128 bits são válidos se forem utilizadas chaves de 512 bits, ou seja estamos partindo de um cifrador que deveria ter complexidade 2^{512} independente da chave analisada, de modo que a complexidade deveria ser a mesma mesmo quando da utilização de chaves de 128 bits.

Quanto ao número de chaves utilizadas para a análise escolhemos 32 chaves devido ao fato de aparentemente termos resultados estáveis com cerca de 20 chaves, exatamente como da consideração dos ataques ao RC6, porém devido a pequenas variações nos resultados prefirimos dar margem à um aumento na segurança dos dados analisados, além de utilizarmos um número gerado pela potência de dois, o que facilita o cálculo quando da consideração do número de cifragens

envolvidas.

Para finalizar as consideraões frizamos que as análises χ^2 apresentadas são feitas nos 2 últimos bits dos sub-blocos (iniciais) A, C, E, G, I, K, M e O em rodadas pares ou em B, D, F, H, J, L, N e P nas rodadas ímpares, ou seja em metade dos sub-blocos. O que nos dá um total de 16 bits a serem analisados. Um dado importante é que notamos que a partir do momento em que a média da análise passava de 75% dos 65536 graus de liberdade o experimento mantinha o crescimento da distância de 65779. Porém apesar de termos notado que alcançar esta média já tornava o experimento *sem volta* resolvemos nos concentrar no momento em que o experimento mantinha 68.27% dos dados dentro do intervalo para a aproximação de uma normal. Ou seja, teriamos que cerca de 84.14% estariam com o valor de χ^2 acima de 65779. O que diferencia nosso ataque do original que considerava a análise para 95% dos graus de liberdade e que não fazia considerações sobre aproximações em uma normal, de maneira a se concentrar apenas no valor da média.

6.1.1 Difusão

Algumas explicações sobre a difusão do cifrador são necessárias para que a eficácia do ataque faça sentido e para que as verificações feitas posteriormente, que chegam a atacar o cifrador com eficácia em até 16 rodadas, possam ser entendidas de maneira mais clara.

Inicialmente levamos em consideração que os sub-blocos do cifrador uma vez afetados por qualquer tipo de operação estariam então afetados como demonstrado na tabela 6.2, porém com as considerações do ataque controlado, como demonstraremos a seguir, é preciso manter o foco nos pontos principais do cifrador.

Nas tabelas que demonstram a difusão marcamos como XXX o bloco referênte ao bloco que sofre mudanças na entrada, como BBB blocos que são modificados ao fim da rodada, e como AAA blocos que já sofreram a modificação considerada - diferente em cada uma das tabelas considerada.

Então com as considerações de ou-exclusivo e rotação definida por

blocos já alterados teriamos a "difusão completa" já ao fim da quinta rodada. Mas resolvemos considerar também só as *modificações fortes* nos blocos, que seriam os ous-exclusivos, de maneira que com o ataque controlado as rotações só começariam a ter força a partir do momento em que os ou-exclusivos começassem a lidar com todos os blocos e fizessem com que as rotações variassem nas análises realizadas.

Na tabela 6.3 mostramos que quando destas considerações somente em 15 rodadas todos os blocos estariam afetados, mas pelas considerações dos ataques da maneira como os realizamos apresentamos os dados na tabela 6.4 de maneira que teriamos a força do cifrador aparecendo somente com 16 rodadas, quando considerando modificações em apenas um sub-bloco.

Por fim na tabela 6.5 demonstramos as considerações para quando lidando com dois sub-blocos, que serão necessários no nosso ataque principal, em que lidamos com os sub-blocos O e M, de maneira a crermos que teremos resultados eficazes em pelo menos até 14 rodadas, de maneira mais eficaz do que o ataque original , sendo que então o crescimento poderia ser maior, e talvez começando a se aproximar do ataque com textos de entrada aleatórios, mas ainda muito longe de 2^{512}, de maneira a não podermos realizar testes.

Estas considerações servem para demonstrar de início porque este cifrador aparenta a necessidade de lidar com pelo menos 16 rodadas, sendo que os ataques que serão demonstrados levam em conta complexidades inferiores a 2^{128} e não 2^{512} - complexidade que deveria ser apresentada por este cifrador.

Ou seja, o ataque controlado proposto realmente parece ser um ataque que terá bons resultados mesmo quando de uma análise preliminar.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Rodada 1	—	—	—	—	—	—	—	—	—	—	—	—	BBB	—	BBB	**XXX**
Rodada 2	—	—	—	—	—	—	—	—	BBB	—	BBB	AAA	BBB	AAA	**XXX**	—
Rodada 3	—	—	—	—	BBB	—	BBB	AAA	BBB	AAA	AAA	AAA	AAA	**XXX**	BBB	—
Rodada 4	BBB	—	BBB	AAA	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	BBB	—
Rodada 5	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	AAA	BBB	AAA
Rodada 6	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	AAA	AAA	AAA	AAA

Tabela 6.2: Difusao para um sub-bloco com o passar das rodadas no MRC6 considerando as rotações e ou-excluvios

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Rodada 1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	BBB	**XXX**
Rodada 2	—	—	—	—	—	—	—	—	—	—	—	—	BBB	AAA	**XXX**	—
Rodada 3	—	—	—	—	—	—	—	—	—	—	BBB	AAA	AAA	**XXX**	—	—
Rodada 4	—	—	—	—	—	—	—	—	BBB	AAA	AAA	AAA	**XXX**	—	—	—
Rodada 5	—	—	—	—	—	—	BBB	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—
Rodada 6	—	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—
Rodada 7	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—
Rodada 8	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	—
Rodada 9	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	BBB	AAA
Rodada 10	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	BBB	AAA	AAA	AAA
Rodada 11	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA
Rodada 12	AAA	AAA	AAA	AAA	**XXX**	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 13	AAA	AAA	AAA	**XXX**	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 14	AAA	AAA	**XXX**	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 15	AAA	**XXX**	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 16	**XXX**	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA

Tabela 6.3: Difusao para um sub-bloco com o passar das rodadas no MRC6 considerando somente ou-excluvios

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Rodada 1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—
Rodada 2	—	—	—	—	—	—	—	—	—	—	—	—	BBB	**XXX**	—	—
Rodada 3	—	—	—	—	—	—	—	—	—	—	BBB	AAA	**XXX**	—	—	—
Rodada 4	—	—	—	—	—	—	—	—	BBB	AAA	AAA	**XXX**	—	—	—	—
Rodada 5	—	—	—	—	—	—	BBB	AAA	AAA	AAA	**XXX**	—	—	—	—	—
Rodada 6	—	—	—	—	BBB	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—
Rodada 7	—	—	BBB	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	—
Rodada 8	BBB	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	—	—
Rodada 9	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	—	BBB	AAA
Rodada 10	AAA	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	—	BBB	AAA	AAA	AAA
Rodada 11	AAA	AAA	AAA	AAA	**XXX**	—	—	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA
Rodada 12	AAA	AAA	AAA	**XXX**	—	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 13	AAA	AAA	**XXX**	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 14	AAA	**XXX**	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 15	**XXX**	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 16	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**

Tabela 6.4: Difusao para um sub-bloco com o passar das rodadas no MRC6 considerando somente ou-excluvios começando em O

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Rodada 1	—	—	—	—	—	—	—	—	—	—	—	—	**XXX**	—	**XXX**	—
Rodada 2	—	—	—	—	—	—	—	—	—	—	BBB	**XXX**	BBB	**XXX**	—	—
Rodada 3	—	—	—	—	—	—	—	—	BBB	AAA	**XXX**	AAA	**XXX**	—	—	—
Rodada 4	—	—	—	—	—	—	BBB	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—
Rodada 5	—	—	—	—	BBB	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	—
Rodada 6	—	—	BBB	AAA	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	—	—
Rodada 7	BBB	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	—	—	—
Rodada 8	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	—	—	BBB	AAA
Rodada 9	AAA	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	—	BBB	AAA	AAA	AAA
Rodada 10	AAA	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA
Rodada 11	AAA	AAA	**XXX**	AAA	**XXX**	—	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 12	AAA	**XXX**	AAA	**XXX**	—	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 13	**XXX**	AAA	**XXX**	—	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA
Rodada 14	AAA	**XXX**	BBB	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**
Rodada 15	**XXX**	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA
Rodada 16	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	AAA	**XXX**	AAA	**XXX**

Tabela 6.5: Difusao para um sub-bloco com o passar das rodadas no MRC6 considerando somente ou-excluvios começando em O e M

6.2 Ataques χ^2 ao MRC6 com blocos de 512 bits

Os ataques apresentados ao MRC6 com bloco de 512 bits tomam inicialmente como base o ataque realizado ao RC6 em [4], de forma adaptada ao MRC6 apenas para demonstrar o comportamento do ataque originalmente como concebido ao cifrador (RC6), e em seguida utilizamos de novas definições quando se tratando dos blocos de texto em claro, sendo estas novas considerações muito importantes para a realização do ataque ao MRC6.

Quanto ao ataque original nos focamos na utilização de 75% dos 65535 graus de liberdade de forma a mantermos as mesmas considerações que serão feitas para o novo ataque.

Com o bloco definido como 512 bits demonstramos resultados contundentes de ataques em até 10 rodadas (com o mesmo valor válido para 11 rodadas em posição diferente), e inferimos sobre ataques com mais rodadas, de modo a demonstrarmos que com a utilização de 15 rodadas a complexidade do ataque ao algorítmo aparenta ser até mesmo menor do que 2^{128} - com uma aproximação *menos otimista* -, o tamanho da chave considerada quando da utilização padrão, ou seja, muito menor do que os 2^{512} como considerado quando da definição do mesmo, mas relembrando que não achamos muito seguro prever o comportamento de cifradores em blocos para um número maior de rodadas do que o realmente analisado, em especial com nossa novas definições para o ataque.

Os resultados que buscamos lidam com a tentativa de obtenção de valores que estejam *todos*, já que lidamos normalmente apenas com 32 chaves diferentes, acima de 75% dos 65535 graus de liberdade quando da analise por χ^2, ou seja queremos que o valor mínimo dos 16 bits analisados dê sempre acima de 65779.

Estes 16 bits são os dois últimos bits dos blocos 0, 2, 4, 6, 8, 10, 12 e 14, ou como definido na figura 2.4 (página 31) os últimos dois bits dos blocos A, C, E, G, I, K, M e O - apenas voltando a frisar que quando apresentando dados para número de rodadas ímpar estes blocos equivalem aos blocos B, D, F, H, J, L, N e P iniciais.

chave 0	e031480e.3fae9326.e3b82cfb.95a9fc79.
chave 1	4b178b17.513d83c3.f75d567d.d11d3906.
chave 2	f7abc9ec.6b7762ef.0227f785.e76deb71.
chave 3	1fb778f3.edb9f28d.cd168c62.7d68ff76.
chave 4	f8a9146f.7ee3d812.aec84ea9.151c83d2.
chave 5	dd150976.7e2eb95b.db3dd1e1.b99b1a92.
chave 6	4d1e432c.c26a8091.b9d65225.10cfb2ec.
chave 7	2eccba26.ba74cbb6.6fb711f7.92c98d5d.
chave 8	4c2478c5.45af030b.89570ca3.8a22e170.
chave 9	0a0cf20c.a3abc035.31303562.665197f4.
chave 10	edd6ab59.b23aed77.e24ead01.6525198a.
chave 11	d56874c8.fd5aebe6.86da7f9a.488e4a0b.
chave 12	939a2b5c.4919131b.6e2b1931.8078d2d3.
chave 13	e68994f9.9e83e24b.cb4d0a81.504ad2d6.
chave 14	2b3a63fb.b63814cb.c6090acd.7ceecd32.
chave 15	886c0dc2.58097720.8c1d79e1.54f19754.
chave 16	cc5944fe.07437463.2bc0203c.fa71577f.
chave 17	4ed5e660.a015ef28.a0d3af46.06eee227.
chave 18	2173365a.888288ea.0946d111.88edfdf1.
chave 19	ff77b4ee.7103b6a2.26c91a4b.6fd440a5.
chave 20	c9f1b55e.b4ca45cb.5681b8ae.a16b9157.
chave 21	17f67b7f.40d40d52.4c1d7461.ba001874.
chave 22	3211ac46.65c318e1.b5a5aa2b.6eef8362.
chave 23	c6fffd69.5a0b02fb.fe8c01ac.53794b58.
chave 24	844e279b.cddcdbfa.26541cd5.e4f7d183.
chave 25	e5b6fdb2.7cba5f84.ff941e7a.6455cbea.
chave 26	9d25d506.9760ff49.4ba5f399.1c460bbc.
chave 27	1c54595f.6a63ba76.ebe2b6df.5d604d0c.
chave 28	5b085829.bbb80193.6bf29f91.134bab5b.
chave 29	e56874fc.d0f8f81b.a06345d9.a2b2fb03.
chave 30	d041a2f2.1882a086.e0481fa6.c5fec211.
chave 31	7337efa8.34ae882d.dea48309.f173b078.

Tabela 6.6: Chaves utilizadas em todas as cifragens analisadas

Desta maneira ao fazermos a aproximação em uma normal estamos dizendo que temos esta tendência no caso de uma normal, mas como já temos os valores todos acima do valor que consideramos poderiamos ter na verdade uma normal com um dos lados mais alto, e não uma aproximação da maneira como

estamos considerando, mas que preferimos considerar um pior caso quando da análise de dados.

Apesar de termos os valores das análises a partir de 2^1 textos, ao obtermos um mínimo acima de 65779 apresentamos apenas alguns valores anteriores que não tenham passado deste valor por termos de simplicidade de apresentação das informações.

Como citado anteriormente o número de 32 chaves é escolhido como padrão para realização do ataque por notarmos que o comportamento aparenta já se estabelecer quando da utilização de 20 chaves do mesmo modo como os ataques quando concebidos originalmente ao RC6 [4] e então definimos um valor um pouco superior como medida de segurança quanto aos dados analisados. As 32 chaves utilizadas nos ataques são apresentadas na tabela 6.6.

6.2.1 Sem imposição de rotações nulas

Os ataques apresentados nessa seção são ataques que não consideram a escolha de textos em claro relacionados com a chave analisada de modo que os mesmos causem a rotação inicial em algum bloco como sendo igual a zero. Com estas considerações em mente primeiro demonstramos o ataque como concebido em [4] adaptado ao MRC6 com sub-blocos de 32 bits e em seguida apresentamos o ataque com nossas novas considerações quanto aos valores de texto em claro atacados, de modo a aproveitarmos o fato de o MRC6 possuir 16 blocos e não apenas 4 como o RC6.

6.2.1.1 Com entrada aleatória e $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$

Este ataque é na verdade uma modificação do ataque realizado em [4] adaptado ao MRC6, de modo que os cinco bits menos significativos de metade dos sub-blocos do texto em claro são definidos como zero, e os outros bits são obtidos através de geração pseudo-aleatória, mais especificamente o próximo texto em claro é o anterior cifrado mais o resultado da função lrand48()*lrand48()+lrand48() de c.

Isto se dá pelas considerações sobre rotação dos blocos feitas em [4].

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	2	2^7	65504.0	303.3	65408.0	65200.7	3	3	00:00:00:00
32	2	2^8	65600.0	384.7	65280.0	65215.3	15	14	00:00:00:00
32	2	2^9	65616.0	301.1	65024.0	65314.9	14	13	00:00:00:00
32	2	2^{10}	65588.0	332.9	65024.0	65255.1	12	15	00:00:00:00
32	2	2^{11}	65560.0	278.1	64960.0	65281.9	7	16	00:00:00:00
32	2	2^{12}	65534.0	350.4	64928.0	65183.6	8	18	00:00:00:01
32	2	2^{13}	65513.0	301.9	64912.0	65211.1	7	17	00:00:00:01
32	2	2^{14}	65477.0	363.7	64880.0	65113.3	8	14	00:00:00:01
32	2	2^{15}	65452.5	351.6	64584.0	65100.9	6	16	00:00:00:02
32	2	2^{16}	65438.2	383.7	64714.0	65054.5	8	15	00:00:00:03
32	2	2^{17}	65465.2	365.3	64824.0	65099.9	9	14	00:00:00:04
32	2	2^{18}	65412.7	347.8	64694.5	65064.9	7	15	00:00:00:07
32	2	2^{19}	65445.1	342.2	64701.8	65102.9	6	15	00:00:00:12
32	2	2^{20}	65484.7	316.5	64762.2	65168.2	6	16	00:00:00:24
32	2	2^{21}	65464.1	380.5	64406.4	65083.6	6	17	00:00:00:46
32	2	2^{22}	65531.4	349.4	64743.3	65182.0	8	17	00:00:01:31
32	2	2^{23}	65524.3	365.7	64969.4	65158.6	7	14	00:00:03:01
32	2	2^{24}	65500.2	361.8	64925.1	65138.3	7	13	00:00:06:00
32	2	2^{25}	65452.7	256.6	64983.8	65196.1	3	18	00:00:12:27
32	2	2^{26}	65441.1	330.6	64563.0	65110.4	3	18	00:00:25:35
32	2	2^{27}	65544.1	317.4	64724.7	65226.7	8	15	00:00:51:42
32	2	2^{28}	65529.2	355.8	64735.0	65173.4	7	18	00:01:43:45
32	2	2^{29}	65456.9	331.6	64809.6	65125.2	5	19	00:03:27:01
32	2	2^{30}	65593.9	322.2	64977.3	65271.7	8	13	00:07:03:15

Tabela 6.7: Ataque χ^2 ao MRC6 com w=32 em 2 rodadas até 2^{30} textos com textos de entrada aleatórios sem os últimos bits zerados

Note que é muito importante a consideração de $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$, porque sem a mesma não teriamos como obter bons resultados nem mesmo quando da análise para duas rodadas, como demonstrado em até 2^{30} textos analisados na tabela 6.7. Outro ponto de observação importante é que quando lidando com a entrada com input pseudo-aleatório temos todos os dados sendo afetados de maneira pseudo-aleatória pelo algorítmo, ou seja, é justamente pela definição destes bits como tal que podemos obter algum resultado quando deste ataque.

Outro ponto importante de observação é que assim como o ataque realizado ao RC6, pelo fato de estarmos lidando com quase todos os bits definidos de forma pseudo-aleatória, temos que lidar com a idéia de que um sub-bloco volta para o ponto em que fazemos a medição para a verificação χ^2 a cada duas rodadas. Então apresentaremos os dados para um número de rodadas pares quando da consideração do ataque realizado desta maneira sabendo também que os valores são os mesmos para a próxima rodada nas posições B, D, F, H, J, L, N e P.

Estas considerações, em relação aos últimos cinco bits dos sub-blocos, buscam avançar em 1 a cada 32 casos, quando considerando um sub-bloco com tamanho de 32 bits por exemplo, o sub-bloco em duas rodadas com o "final" (5 bits) constante, de modo a ter um ganho nas considerações para a análise do χ^2 do mesmo modo como são apresentadas nas considerações feitas no ataque ao RC6.

Note que zeramos apenas 40 bits dos textos em claro utilizados no cifrador de modo a mantermos o ataque análogo ao apresentado em [4], e mesmo assim ainda temos 472 bits para trabalhar, mas como estamos lidando com complexidades que consideramos ser no máximo 2^{128} - já que consideramos obter os dados com complexidade menor do que a chave de 128 bits - e não 2^{512} como deveria ser pelo número de bits do cifrador, então temos textos suficiente para trabalhar, de maneira diferente do ataque original em que o número de textos considerado fazia alguma diferença nas considerações do ataque.

O ataque, como apresentado na tabela 6.8, demonstra resultados que para 2 rodadas já temos a média acima dos 75% considerados em 2^{17} textos em claro, sendo que com 2^{18} textos já temos 84.14% dos valores acima dos 75% considerados já com a diferenciação da amostra de uma amostra aleatória quando da análise χ^2, mas para 4 rodadas ainda não obtemos bons resultados mesmo com 2^{30} textos em claro apesar de o estudo de [4] já apresentar bons resultados para o RC6 com 2^{29} textos em claro.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	2	2^{7}	65504.0	303.3	65408.0	65200.7	3	3	00:00:00:00
32	2	2^{8}	65616.0	403.1	65280.0	65212.9	15	14	00:00:00:00
32	2	2^{9}	65488.0	334.3	65024.0	65153.7	9	15	00:00:00:00
32	2	2^{10}	65516.0	373.7	64640.0	65142.3	9	17	00:00:00:00
32	2	2^{11}	65618.0	345.7	64832.0	65272.3	11	13	00:00:00:00
32	2	2^{12}	65620.0	301.6	64992.0	65318.4	9	14	00:00:00:00
32	2	2^{13}	65673.0	355.7	65040.0	65317.3	14	16	00:00:00:00
32	2	2^{14}	65578.0	340.5	64792.0	65237.5	9	16	00:00:00:00
32	2	2^{15}	65529.5	352.6	64696.0	65176.9	9	19	00:00:00:01
32	2	2^{16}	65631.5	408.1	64666.0	65223.4	11	15	00:00:00:01
32	**2**	$\mathbf{2^{17}}$	**65898.5**	**323.3**	**64990.0**	**65575.2**	**20**	**17**	**00:00:00:03**
32	**2**	$\mathbf{2^{18}}$	**66120.7**	**314.6**	**65539.5**	**65806.0**	**28**	**15**	**00:00:00:06**
32	**2**	$\mathbf{2^{19}}$	**66719.6**	**319.7**	**66084.0**	**66400.0**	**32**	**18**	**00:00:00:11**
32	2	2^{20}	67976.8	317.3	67239.1	67659.5	32	16	00:00:00:23
32	4	2^{16}	65564.4	379.2	64732.0	65185.2	8	14	00:00:00:02
32	4	2^{17}	65493.9	434.3	64624.0	65059.6	9	17	00:00:00:03
32	4	2^{18}	65551.8	328.6	65014.0	65223.1	8	14	00:00:00:07
32	4	2^{19}	65550.6	313.5	64908.5	65237.1	7	13	00:00:00:13
32	4	2^{20}	65414.2	333.3	64748.9	65081.0	5	13	00:00:00:26
32	4	2^{21}	65466.3	299.6	64960.3	65166.7	5	14	00:00:00:52
32	4	2^{22}	65493.5	311.2	64919.9	65182.3	7	13	00:00:01:44
32	4	2^{23}	65462.3	422.5	64361.7	65039.8	8	17	00:00:03:28
32	4	2^{24}	65507.6	350.1	64676.3	65157.5	9	16	00:00:06:57
32	4	2^{25}	65518.1	346.3	64838.6	65171.8	6	17	*00:00:13:54*
32	4	2^{26}	65584.3	380.1	64821.2	65204.2	8	12	*00:00:27:48*
32	4	2^{27}	65535.2	383.8	64790.5	65151.4	9	16	*00:00:55:36*
32	4	2^{28}	65557.4	346.4	64940.2	65211.0	8	15	*00:01:51:12*
32	4	2^{29}	65509.4	326.5	64768.9	65182.9	7	18	*00:03:42:24*
32	4	2^{30}	65558.4	389.9	64708.8	65168.4	9	15	*00:07:24:48*

Tabela 6.8: Ataque χ^2 ao MRC6 com w=32 em 2 e 4 rodadas até 2^{30} textos com textos de entrada aleatórios com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$

Não vamos muito além da análise de 2^{30} textos para 4 rodadas já que também não esperamos por bons resultados com base no mesmo tipo de ataque realizado no RC6.

O ponto principal de quando lidando com este ataque de forma análoga ao apresentado ao RC6 em [4] é que no caso em que zeramos um mínimo

número de bits realmente o ataque ao MRC6 não vai tão longe quanto o ataque pode ir quando lidando com o RC6.

6.2.1.2 Com entrada aleatória, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

Então com base nas nossas considerações feitas no ínicio deste capítulo sobre novas considerações para os ataques ao MRC6 resolvemos realizar o mesmo ataque feito anteriormente só que com a adição da condição $\{B, D, F, H, J, K, N, P\} = 0x0$, o que causaria o efeito de manter as rotações constantes na primeira rodada, já que de modo diferente do RC6 no MRC6 temos bits "de sobra" para trabalharmos e então fixamos um ataque com a possibilidade de até 2^{216} textos em claro, porém gerando os mesmo de forma pseudo-aleatória.

Para o novo ataque concebido os dados para 2 rodadas ainda são confusos demais para definirmos onde começa a distinção devido ao rápido avanço, e então quando considerando o ataque para 4 rodadas já temos a média acima dos 75% considerados para o valor do ataque em 2^{25} textos em claro, mas verificamos que o desvio padrão continua se mantendo muito elevado de maneira que ainda não obtemos os resultados para 84.14% dos textos acima dos 75% considerados como demonstrado na tabela 6.9.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	2	2^{6}	65856.0	812.2	65472.0	65043.8	6	5	00:00:00:00
32	**2**	$\mathbf{2^{7}}$	**65792.0**	**623.7**	**65408.0**	**65168.3**	**10**	**9**	**00:00:00:00**
32	2	2^{8}	66592.0	1363.6	65280.0	65228.4	24	15	00:00:00:00
32	2	2^{9}	67808.0	3106.0	65024.0	64702.0	24	10	00:00:00:00
32	2	2^{10}	70020.0	6990.0	65024.0	63030.0	26	8	00:00:00:00
32	2	2^{11}	74602.0	13397.2	65216.0	61204.8	25	8	00:00:00:00
32	2	2^{12}	83929.0	26970.8	65056.0	56958.2	24	8	00:00:00:00
32	2	2^{13}	101843.0	53465.1	64912.0	48377.9	25	8	00:00:00:00
32	2	2^{14}	138096.5	106103.0	65056.0	31993.5	25	8	00:00:00:00
32	2	2^{15}	210753.8	212029.7	64728.0	—	25	8	00:00:00:01
32	2	2^{16}	356302.9	424800.0	65026.0	—	25	8	00:00:00:01
32	2	2^{17}	647064.2	850294.9	64518.0	—	25	8	00:00:00:03
32	2	2^{18}	1228914.5	1701271.3	65072.0	—	26	8	00:00:00:06
32	2	2^{19}	2391988.3	3402812.4	64808.8	—	25	8	00:00:00:12
32	4	2^{6}	65536.0	362.0	65472.0	65174.0	1	1	00:00:00:00
32	4	2^{7}	65504.0	303.3	65408.0	65200.7	3	3	00:00:00:00
32	4	2^{8}	65408.0	290.8	65280.0	65117.2	6	6	00:00:00:00
32	4	2^{9}	65488.0	279.2	65024.0	65208.8	10	18	00:00:00:00
32	4	2^{10}	65564.0	468.6	64768.0	65095.4	9	13	00:00:00:00
32	4	2^{11}	65542.0	431.1	64640.0	65110.9	12	15	00:00:00:00
32	4	2^{12}	65540.0	338.9	64704.0	65201.1	8	14	00:00:00:00
32	4	2^{13}	65579.0	292.7	65008.0	65286.3	8	13	00:00:00:00
32	4	2^{14}	65584.2	299.7	65080.0	65284.5	9	14	00:00:00:00
32	4	2^{15}	65655.4	424.2	64664.0	65231.2	14	14	00:00:00:01
32	4	2^{16}	65638.2	326.2	64962.0	65312.0	9	13	00:00:00:02
32	4	2^{17}	65693.7	317.0	65219.0	65376.6	13	13	00:00:00:04
32	4	2^{18}	65526.7	386.0	64576.0	65140.7	9	18	00:00:00:08
32	4	2^{19}	65540.3	349.7	64527.5	65190.6	9	15	00:00:00:14
32	4	2^{20}	65521.3	349.4	64526.4	65171.9	8	15	00:00:00:29
32	4	2^{21}	65563.7	317.3	64953.8	65246.5	8	14	00:00:00:57
32	4	2^{22}	65559.1	327.9	64782.8	65231.2	9	16	00:00:01:53
32	4	2^{23}	65580.8	450.5	64681.8	65130.3	9	17	00:00:03:44
32	4	2^{24}	65723.0	636.4	64834.3	65086.7	12	11	00:00:07:28
32	**4**	$\mathbf{2^{25}}$	**66023.0**	**1027.7**	**65047.5**	**64995.3**	**16**	**10**	***00:00:14:56***
32	4	2^{26}	66534.3	2035.7	64921.0	64498.6	16	8	*00:00:29:52*
32	4	2^{27}	67597.5	4074.7	64847.3	63522.8	20	8	*00:00:59:44*
32	4	2^{28}	69867.4	8138.1	64811.1	61729.3	24	7	*00:01:59:28*
32	4	2^{29}	74285.0	16364.1	64672.8	57920.9	27	7	*00:03:58:58*
32	4	2^{30}	82923.9	32866.9	64994.6	50057.1	22	7	*00:07:57:56*

Tabela 6.9: Ataque χ^2 ao MRC6 com w=32 em 2 e 4 rodadas até 2^{30} textos com textos de entrada aleatórios com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

Neste ponto voltamos a ressaltar o que foi discutido no início do capítulo que quando apresentados os dados para rodadas ímpares estariamos demonstrando os dados como são refletidos em relação aos blocos B, D, F, H, J, K, N e P, sendo que este comportamento de crescimento parece se manter quando da adição da condição $\{B, D, F, H, J, K, N, P\} = 0x0$.

E então apresentamos os dados para 3 rodadas na tabela 6.10, demonstrando um comportamento interessante no cifrador quando destas considerações já que a partir deste tipo de considerações podemos lidar com os resultados para os blocos de texto em claro iniciados em B, D, F, H, J, K, N e P.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	3	2^5	65504.0	0.0	65504.0	65504.0	0	31	00:00:00:00
32	3	2^6	65536.0	362.0	65472.0	65174.0	1	1	00:00:00:00
32	3	2^7	65568.0	377.8	65408.0	65190.2	5	5	00:00:00:00
32	3	2^8	65632.0	399.5	65280.0	65232.5	17	17	00:00:00:00
32	3	2^9	65568.0	372.1	65024.0	65195.9	12	12	00:00:00:00
32	3	2^{10}	65572.0	385.0	64768.0	65187.0	10	13	00:00:00:00
32	3	2^{11}	65480.0	339.7	65024.0	65140.3	5	10	00:00:00:00
32	3	2^{12}	65453.0	346.0	64800.0	65107.0	4	16	00:00:00:00
32	3	2^{13}	65618.5	395.7	64832.0	65222.8	9	15	00:00:00:00
32	3	2^{14}	65764.2	447.2	65016.0	65317.0	13	13	00:00:00:00
32	**3**	$\mathbf{2^{15}}$	**65999.5**	**471.8**	**64980.0**	**65527.7**	**21**	**16**	**00:00:00:01**
32	3	2^{16}	66236.5	816.4	64958.0	65420.1	23	15	00:00:00:02
32	3	2^{17}	66957.0	1481.0	65017.0	65476.0	26	14	00:00:00:03
32	3	2^{18}	68385.1	2898.5	65128.0	65486.5	29	13	00:00:00:06
32	3	2^{19}	71430.8	5841.7	65721.5	65589.1	31	13	00:00:00:12
32	**3**	$\mathbf{2^{20}}$	**77362.7**	**11709.5**	**65932.2**	**65653.2**	**32**	**12**	**00:00:00:25**
32	3	2^{21}	89166.1	23430.0	66547.1	65736.0	32	13	00:00:00:51

Tabela 6.10: Ataque χ^2 ao MRC6 com w=32 em 3 rodadas até 2^{21} textos com textos de entrada aleatórios com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

6.2.1.3 Com entrada aleatória e $\{B, D, F, H, J, K, N, P\} = 0x0$

Depois da verificação de que uma nova combinação de fatores resultava em resultados ainda melhores resolvemos então fazer a verificação de qual seria o efeito de lidarmos apenas com $\{B, D,\ F, H, J, K, N, P\} = 0x0$, para termos certeza de que era a combinação dos fatores que estaria causando o ganho nas condições anteriores.

Porém mesmo para 2 rodadas não obtemos resultados quando desta consideração sozinha, como apresentado na tabela 6.11.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	2	2^4	65776.0	1448.2	65520.0	64327.8	1	1	00:00:00:00
32	2	2^5	65632.0	724.1	65504.0	64907.9	1	1	00:00:00:00
32	2	2^6	65728.0	862.6	65472.0	64865.4	3	2	00:00:00:00
32	2	2^7	65600.0	482.2	65408.0	65117.8	5	4	00:00:00:00
32	2	2^8	65520.0	367.5	65280.0	65152.5	11	10	00:00:00:00
32	2	2^9	65544.0	334.6	65024.0	65209.4	12	11	00:00:00:00
32	2	2^{10}	65584.0	290.3	64896.0	65293.7	11	15	00:00:00:00
32	2	2^{11}	65528.0	436.7	64768.0	65091.3	8	14	00:00:00:00
32	2	2^{12}	65586.0	388.5	64672.0	65197.5	10	14	00:00:00:00
32	2	2^{13}	65606.0	316.2	64864.0	65289.8	9	14	00:00:00:00
32	2	2^{14}	65643.5	332.0	65112.0	65311.5	13	14	00:00:00:00
32	2	2^{15}	65700.2	385.7	65068.0	65314.6	12	13	00:00:00:01
32	2	2^{16}	65566.2	413.1	64822.0	65153.1	10	15	00:00:00:02
32	2	2^{17}	65595.0	350.2	64804.0	65244.8	9	18	00:00:00:04
32	2	2^{18}	65587.9	314.3	64996.0	65273.6	8	12	00:00:00:06
32	2	2^{19}	65496.6	420.3	64424.2	65076.3	7	17	00:00:00:12
32	2	2^{20}	65601.6	361.8	64952.1	65239.8	10	17	00:00:00:24
32	2	2^{21}	65605.1	449.7	64761.8	65155.3	13	14	00:00:00:47
32	2	2^{22}	65526.1	298.2	64896.4	65227.9	7	13	00:00:01:33
32	2	2^{23}	65454.7	414.2	64580.2	65040.4	9	16	00:00:03:06
32	2	2^{24}	65466.4	374.6	64727.4	65091.7	5	18	00:00:06:12
32	2	2^{25}	65448.4	314.7	64905.1	65133.8	5	13	00:00:12:24
32	2	2^{26}	65544.6	394.1	64669.3	65150.5	9	16	00:00:24:45

Tabela 6.11: Ataque χ^2 ao MRC6 com w=32 em 2 rodadas até 2^{26} textos com textos de entrada aleatórios com $\{B, D,\ F, H, J, K, N, P\} = 0x0$

Então podemos deduzir que foi a combinação dos fatores que tornou possível um aumento no ganho do ataque. Estas considerações são importantes para podermos considerar os próximos ataques para progressões futuras quando lidando com mais do que 2^{32} textos, momento em que estariamos então lidando com mais do que um bloco com os dados sendo alterados na seqüencia dos textos considerados, de modo a parecer mais consistente a idéia de lidarmos então com 2^{27} variações em um bloco antes de passarmos para o próximo em vez de 2^{32}.

6.2.1.4 Com entrada controlada, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

Aqui lidamos com a aplicação de nossa nova idéia, a qual define que os bits dos sub-blocos B, D, F, H, J, K, N e P de texto em claro utilizados são definidos como sendo nulos, ou seja zero, de modo a trabalharmos somente com os bits nas palavras O e M quando considerando ataques com até 2^{54} textos em claro, já que os 5 bits menos significativos são definidos como zero. Começamos na verdade apenas lidando com o sub-bloco O até 2^{27} textos e então passamos para o sub-bloco M, e como não podemos nem mesmo chegar em 2^{54} textos então não chegamos a lidar com os valores nos próximos sub-blocos com a possibilidade de trabalho.

É interessante notar que nos dados apresentados nas tabelas 6.12, 6.13 e 6.14 chegamos a lidar com 2^{41} textos criptografados, sendo na verdade 2^{36} para 32 chaves. Porém sabemos que para lidarmos com mais do que 2^{41} textos criptografados necessitamos de um pouco mais do que duas semanas com o processamento disponível, e evitamos ir tão longe.

E então poderemos trabalhar somente com os dados para o ataques realizados em até 12 rodadas, mas tomando como base os dados de crescimento do logarítmo base 2 e com as informações de crescimento a cada rodada, temos informações que julgamos suficientes para estabelecermos um crescimento aproximado a partir de então - apesar de não podermos trabalhar com a quantidade de texto necessária a partir daqui para podermos conferir os dados gerados a partir da análise.

Em duas rodadas podemos verificar pelos dados apresentados na tabela 6.12 que ao passar de 2^1 textos a média cresce de modo realmente rápido assim como esperavamos, já que sabemos que a mistura das palavras ainda não está boa o suficiente.

Nós não esperavamos que o comportamento do cifrador "estabilizasse" neste ataque com menos de 8 rodadas de maneira distante dos 75% - sendo que terminamos as considerações sobre crescimento a partir de 9 rodadas -, mas queriamos demonstrar como ele reagiria com menos rodadas. E também não temos uma grande confiança quando lidando com baixas quantidades de texto.

Apenas relembramos que nos focamos nas rodadas pares inicialmente e passamos a verificação das rodadas ímpares devido ao ou-exclusivo como comentado anteriomente apenas para podermos fazer um acompanhamento do crescimento com mais detalhes.

Em 4 rodadas podemos verificar nos dados apresentados o aumento da mistura das palavras começando refletir no processo de definição da média, mas com 2^3 textos já temos a média bem distânte dos 75% considerados e mesmo quando considerando 84.14% dos dados já obtemos com 2^5 textos o valor acima dos 75% considerados para a análise.

Neste ponto podemos verificar que o avanço quando comparando as rodadas pares e as rodadas ímpares pode não ser definido exatamente como o mesmo, já que nas rodadas ímpares temos a soma com um "valor desconhecido" a mais. E podemos observar que em 4 rodadas as distâncias dos valores obtidos já são como as distâncias em 6 e 7 rodadas, mas em 5 rodadas temos uma pequena diferença. É claro que como em 7 rodadas os valores estão seguindo a mesma tendência preferimos assumir que este acaso ocorreu devido ao ainda baixo número de textos analisados, ou até mesmo que o crescimento não se dê de maneira a somar números inteiros no expoente, o que é muito mais provável.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	**1**	$\mathbf{2^1}$	**131070.0**	**0.0**	**131070.0**	**131070.0**	**32**	**31**	**00:00:00:00**
32	1	2^2	262140.0	0.0	262140.0	262140.0	32	31	00:00:00:00
32	**2**	$\mathbf{2^1}$	**126974.0**	**16117.6**	**65534.0**	**110856.4**	**30**	**29**	**00:00:00:00**
32	2	2^2	245756.0	52639.8	65532.0	193116.2	30	28	00:00:00:00
32	**2**	$\mathbf{2^3}$	**479224.0**	**122818.3**	**131064.0**	**356405.7**	**32**	**27**	**00:00:00:00**
32	2	2^4	884720.0	295908.6	262128.0	588811.4	32	23	00:00:00:00
32	**3**	$\mathbf{2^1}$	**75774.0**	**24176.4**	**65534.0**	**51597.6**	**5**	**5**	**00:00:00:00**
32	**3**	$\mathbf{2^2}$	**100348.0**	**34253.8**	**65532.0**	**66094.2**	**22**	**6**	**00:00:00:00**
32	3	2^3	139256.0	56603.1	65528.0	82652.9	28	18	00:00:00:00
32	**3**	$\mathbf{2^4}$	**206576.0**	**109428.2**	**90096.0**	**97147.8**	**32**	**13**	**00:00:00:00**
32	3	2^5	348000.0	223100.1	147424.0	124899.9	32	12	00:00:00:00
32	**4**	$\mathbf{2^3}$	**69112.0**	**6881.5**	**65528.0**	**62230.5**	**7**	**7**	**00:00:00:00**
32	4	2^4	74736.0	10558.8	65520.0	64177.2	20	9	00:00:00:00
32	**4**	$\mathbf{2^5}$	**82656.0**	**14394.9**	**65504.0**	**68261.1**	**31**	**8**	**00:00:00:00**
32	**4**	$\mathbf{2^6}$	**101312.0**	**25446.9**	**77760.0**	**75865.1**	**32**	**10**	**00:00:00:00**
32	4	2^7	134048.0	43087.0	91008.0	90961.0	32	10	00:00:00:00
32	**5**	$\mathbf{2^4}$	**66288.0**	**2426.0**	**65520.0**	**63862.0**	**3**	**3**	**00:00:00:00**
32	5	2^5	66912.0	2233.6	65504.0	64678.4	10	9	00:00:00:00
32	5	2^6	67584.0	2727.1	65472.0	64856.9	19	7	00:00:00:00
32	**5**	$\mathbf{2^7}$	**69696.0**	**2540.9**	**65408.0**	**67155.1**	**31**	**14**	**00:00:00:00**
32	**5**	$\mathbf{2^8}$	**75792.0**	**3341.0**	**69376.0**	**72451.0**	**32**	**13**	**00:00:00:00**
32	5	2^9	86672.0	5683.9	78080.0	80988.1	32	10	00:00:00:00
32	6	2^6	65664.0	606.5	65472.0	65057.5	3	3	00:00:00:00
32	**6**	$\mathbf{2^7}$	**66016.0**	**857.1**	**65408.0**	**65158.9**	**13**	**13**	**00:00:00:00**
32	6	2^8	66304.0	946.8	65280.0	65357.2	25	18	00:00:00:00
32	**6**	$\mathbf{2^9}$	**67288.0**	**951.5**	**65024.0**	**66336.5**	**31**	**17**	**00:00:00:00**
32	**6**	$\mathbf{2^{10}}$	**68820.0**	**1056.3**	**66560.0**	**67763.7**	**32**	**14**	**00:00:00:00**
32	6	2^{11}	71940.0	2031.2	69056.0	69908.8	32	12	00:00:00:00
32	7	2^6	65600.0	503.7	65472.0	65096.3	2	2	00:00:00:00
32	7	2^7	65536.0	344.1	65408.0	65191.9	4	4	00:00:00:00
32	7	2^8	65568.0	342.5	65280.0	65225.5	15	14	00:00:00:00
32	7	2^9	65600.0	379.2	65024.0	65220.8	13	13	00:00:00:00
32	7	2^{10}	65712.0	370.3	65152.0	65341.7	12	11	00:00:00:00
32	**7**	$\mathbf{2^{11}}$	**65944.0**	**442.7**	**65088.0**	**65501.3**	**21**	**14**	**00:00:00:00**
32	7	2^{12}	66316.0	542.4	65184.0	65773.6	27	14	00:00:00:00
32	**7**	$\mathbf{2^{13}}$	**67247.5**	**713.9**	**65760.0**	**66533.6**	**31**	**13**	**00:00:00:00**
32	**7**	$\mathbf{2^{14}}$	**68864.2**	**1099.2**	**67176.0**	**67765.1**	**32**	**16**	**00:00:00:00**
32	7	2^{15}	72282.4	1996.6	68552.0	70285.8	32	16	00:00:00:00

Tabela 6.12: Ataque χ^2 ao MRC6 com w=32 em 1, 2, 3, 4, 5, 6 e 7 rodadas até 2^{15} textos com textos de entrada controlados, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D,\ F, H, J, K, N, P\} = 0x0$

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	8	2^{12}	65472.0	391.2	64608.0	65080.8	7	13	00:00:00:00
32	8	2^{13}	65574.5	469.9	64720.0	65104.6	10	13	00:00:00:00
32	8	2^{14}	65711.8	407.2	64840.0	65304.6	13	15	00:00:00:00
32	**8**	$\mathbf{2^{15}}$	**65828.6**	**385.7**	**65168.0**	**65442.9**	**17**	**16**	**00:00:00:00**
32	8	2^{16}	66144.7	414.0	65022.0	65730.6	26	17	00:00:00:01
32	**8**	$\mathbf{2^{17}}$	**66617.9**	**475.5**	**65576.0**	**66142.5**	**31**	**15**	**00:00:00:02**
32	**8**	$\mathbf{2^{18}}$	**67691.1**	**793.3**	**66357.5**	**66897.8**	**32**	**15**	**00:00:00:04**
32	8	2^{19}	69795.1	1538.0	67365.2	68257.1	32	14	00:00:00:08
32	9	2^{14}	65484.0	363.4	64704.0	65120.6	5	16	00:00:00:00
32	9	2^{15}	65498.4	435.2	64296.0	65063.2	8	18	00:00:00:01
32	9	2^{16}	65591.2	343.9	64866.0	65247.3	11	16	00:00:00:02
32	9	2^{17}	65572.9	324.2	64924.0	65248.8	10	17	00:00:00:03
32	9	2^{18}	65665.3	306.3	65116.0	65359.1	12	13	00:00:00:05
32	9	2^{19}	65757.6	352.1	64957.5	65405.5	18	17	00:00:00:10
32	**9**	$\mathbf{2^{20}}$	**65980.5**	**374.2**	**65181.5**	**65606.3**	**25**	**15**	**00:00:00:19**
32	9	2^{21}	66228.8	505.4	65130.6	65723.3	27	16	00:00:00:39
32	**9**	$\mathbf{2^{22}}$	**66834.0**	**689.1**	**65872.5**	**66144.9**	**32**	**15**	**00:00:01:14**
32	9	2^{23}	68176.6	1218.5	66195.0	66958.1	32	14	00:00:02:24
32	10	2^{22}	65490.2	422.1	64710.0	65068.1	8	16	00:00:01:21
32	10	2^{23}	65576.4	383.9	64587.5	65192.6	9	16	00:00:02:44
32	10	2^{24}	65713.6	331.5	64810.2	65382.1	15	15	00:00:05:20
32	**10**	$\mathbf{2^{25}}$	**65908.9**	**374.5**	**65221.0**	**65534.3**	**18**	**14**	**00:00:10:33**
32	**10**	$\mathbf{2^{26}}$	**66358.0**	**491.7**	**65454.1**	**65866.3**	**30**	**12**	**00:00:20:55**
32	10	2^{27}	67184.9	765.4	65463.2	66419.5	31	14	00:00:41:31
32	**10**	$\mathbf{2^{28}}$	**67125.5**	**703.8**	**65822.2**	**66421.7**	**32**	**13**	**00:01:22:42**
32	10	2^{29}	67020.8	724.9	65983.8	66295.9	32	12	00:02:45:06

Tabela 6.13: Ataque χ^2 ao MRC6 com w=32 em 8, 9 e 10 rodadas até 2^{29} textos com textos de entrada controlados, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

Em 6 rodadas podemos verificar que o número de textos necessários para que a média passe dos 75% já sobe para 2^7 textos e que para 84.14% dos valores acima dos 75% precisamos de 2^9 textos. A partir deste ponto podemos também verificar que o valor do mínimo para 84.14% dos valores está cada vez mais

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	11	2^{30}	65567.4	332.9	64880.9	65234.5	10	12	00:06:00:21
32	11	2^{31}	65651.3	361.8	64926.1	65289.5	10	13	00:11:58:37
32	11	2^{32}	65672.7	445.3	64658.2	65227.4	13	14	00:23:55:14
32	11	2^{33}	65604.0	360.0	64885.7	65244.0	13	16	01:23:48:16
32	11	2^{34}	65617.7	412.5	64811.7	65205.2	14	17	03:23:46:12
32	11	2^{35}	65663.7	377.8	64521.2	65285.8	14	15	07:23:38:28
32	11	2^{36}	65692.6	352.4	64922.0	65340.2	14	16	15:23:15:25
32	12	2^{32}	65523.3	373.0	64788.8	65150.4	6	14	*01:00:00:00*
32	12	2^{33}	65535.0	384.2	64725.5	65150.8	10	14	*02:00:00:00*
32	12	2^{34}	65505.2	360.1	64879.6	65145.0	6	16	*04:00:00:00*
32	12	2^{35}	65553.8	405.6	64975.5	65148.3	9	12	*08:00:00:00*
32	12	2^{36}	65503.6	347.2	64857.6	64462.0	6	16	*16:00:00:00*

Tabela 6.14: Ataque χ^2 ao MRC6 com w=32 em 11 e 12 rodadas até 2^{36} textos com textos de entrada controlados, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

se aproximando do valor da média.

Em 8 rodadas chegamos no ponto em que queriamos. Neste ponto o valor mínimo esta perto (distância do número de textos para ultrapassar os 75%) o suficiente da média assim como esperavamos, já que o cifrador esta funcionando com uma melhor mistura entre as palavras iniciais neste ponto, e a partir dos dados que obtivermos neste ponto poderemos então especular sobre o que acontece nas próximas rodadas.

E mais importante do que isso, no crescimento acontecido a partir daqui começamos a ver uma *mudança de comportamento* nos dados obtidos na análise do cifrador, já que em 9 rodadas a tendência começa a mudar de maneira que a partir de 10 rodadas parece que o cifrador pode ter uma melhora na distância entre a média e os 84.14%.

Na análise do crescimento de 8 para 10 rodadas temos que para que a média se mantenha em 75% crescemos de 2^{15} para 2^{25}, mas não confirmamos o crescimento de expoente em cerca de 10 com a obtenção do resultado para 12 rodadas em 2^{36} como demonstrado na tabela 6.14, com os dados de crescimento

envolvendo as rodadas ímpares sendo apresentados na tabela 6.15.

Com base nos dados obtidos em até 10 rodadas temos também que quando da obtenção da estabilidade dos dados a média passa primeiro dos 75% para que com o dobro de textos tenhamos que 84.14% dos valores analisados também fique acima dos 75% considerados.

E com base nos dados das tabelas 6.15 e 6.16 podemos estimar uma necessidade de textos super exponencial de cerca de $f(x) = 3.5 * x^2 - 61.5x + 290$ para x maior ou igual a 9 rodadas.

Rodadas	1	2	3	4	5	6	7	8	9	10	11	12
média >= 75%	2^1	2^1	2^1	2^3	2^4	2^7	2^{10}	2^{15}	2^{20}	$\mathbf{2^{25}}$	$\mathbf{2^{37}}$	$\mathbf{2^{56}}$
diferença	—	0	0	2	1	3	3	5	5	5	**12**	**19**
mudança	—	—	0	2	1	2	0	2	0	0	**7**	**7**

Tabela 6.15: Crescimento no ataque χ^2 ao MRC6 com w=32 em 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 rodadas e estimado para 11 e 12 rodadas para a média

Rodadas	1	2	3	4	5	6	7	8	9	10	11	12
média >= 75%	2^1	2^1	2^2	2^3	2^4	2^9	2^{13}	2^{17}	2^{22}	$\mathbf{2^{26}}$	$\mathbf{2^{38}}$	$\mathbf{2^{57}}$
diferença	—	0	1	1	1	5	4	4	5	4	**12**	**19**
mudança	—	—	0	0	0	4	1	0	1	1	**8**	**7**

Tabela 6.16: Crescimento no ataque χ^2 ao MRC6 com w=32 em 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 rodadas e estimado para 11 e 12 rodadas para 84.14% dos valores

As informações que temos até aqui nos levam a crer que quando atingimos uma média de aproximadamente 65779 o crescimento não tende mais a diminuir. Ou seja ao passar de 75% na análise χ^2 os dados obtidos utilizando-se o cifrador realmente já se definem diferentes de um conjunto de dados aleatórios sem tendência a voltar.

Rodadas	Estimativa para a média	Estimativa para 84.14%
8 rodadas	2^{15} textos	2^{17} textos
9 rodadas	2^{20} textos	2^{22} textos
10 rodadas	2^{25} textos	2^{26} textos
11 *rodadas*	**2^{37} *textos***	**2^{38} *textos***
12 rodadas	2^{56} textos	2^{57} textos
13 rodadas	2^{82} textos	2^{83} textos
14 rodadas	2^{115} textos	2^{116} textos
15 *rodadas*	**2^{155} *textos***	**2^{156} *textos***
20 rodadas	2^{460} *textos*	2^{461} *textos*

Tabela 6.17: Estimativa de crescimento para a média e 84.14% baseado em $f(x) = 3.5 * x^2 - 61.5 * x + 290$ a partir de 9 rodadas

A estimativa para o crescimento super-exponencial dos textos necessário para a diferenciação da média para até 15 rodadas e a estimativa para o crescimento dos textos necessários para a diferenciação de 84.14% dos dados são apresentadas na tabela 6.17 - o valor para 20 rodadas é apresentado apenas de maneira ilustrativa.

Note que pelos dados que apresentamos na tabela 6.17 o cifrador MRC6 já poderia ser diferenciado de uma amostra aleatória quando lidando com 2^{156} textos em 15 rodadas - mais do que 2^{128} - com as considerações extras para rodadas ímpares, mas com a medição acompanhando os blocos medidos na rodada 14 teriamos a diferenciação em 15 rodadas com 2^{116} textos em claro.

É muito importante lembrarmos que a estimativa de crescimento super-exponencial leva em consideração o pior caso como pudemos considerar, já que não temos segurança em lidar com aproximações para mais rodadas, e que o crescimento exponencial pode até mesmo ser um crescimento fixo a partir do ponto em que conseguimos realizar as medições. Ou seja, o cifrador MRC6 pode ser inseguro em até mesmo mais do que 15 rodadas - isso que estamos considerando complexidade 2^{128} e não 2^{512}.

6.2.1.5 Sobre estimativas de crescimento

Nossas considerações sobre complexidade do cifrador levam em consideração que os resultados obtidos para 8 rodadas seriam resultados aceitáveis se a oitava rodada fosse a primeira, e com base nos dados a partir de 8 rodadas, quando os blocos começam a iteragir com um pouco mais de força, então começamos a fazer estimativas.

Inicialmente, com os teste sendo realizados para até 10 rodadas achamos erroneamente que o crescimento exponencial poderia de dar de forma linear somente com a adição de 10 a cada duas rodadas, levando em consideração o crescimento apresentado em [4], mas é claro que a mesma consideração não deveria fazer sentido, já que nosso ataque melhora o rendimento do ataque original e com o passar das rodadas, devido à necessidade de um maior número de textos e então aproximação a aleatoriedade de textos de entrada, então se aproxima dos valores das considerações para o ataque aleatório original.

Como pode ser observado na tabela 6.18 na sessão de mudança tinhamos a impressão que esta mudança iria passar para zero quando do crescimento de 10 para 12 rodadas, o que deveria nos dar resultados em 2^{35}, porém ao realizarmos estes testes não obtivemos bons resultados mesmo em 2^{36}.

Rodadas	2	4	6	8	10	12
média >= 75%	2^1	2^3	2^7	$\mathbf{2^{15}}$	$\mathbf{2^{25}}$	2^{xx}
diferença	—	2	4	8	**10**	x
mudança	—	—	2	4	2	x

Tabela 6.18: Crescimento no ataque χ^2 ao MRC6 com w=32 em 2, 4, 6, 8 e 10 rodadas para a média

Então como explicado anteriormente, levando em consideração que os blocos medidos eram somados por ou-exclusivo com os blocos ao lado e sofriam novamente operações no cifrador para os dados sendo medidos nos mesmos blocos

nas rodadas ímpares resolvemos medir os resultados nestas rodadas de maneira a tentar obter dados mais concisos sobre o crescimento da complexidade com o passar das rodadas, já que no estudo original feito em [4] foi também demonstrado que com os outros blocos tendo os últimos bits definidos como zero, e aqui são todos os bits destes definidos como zero, obtinhamos resultados, mas é claro que a consideração do ou-exclusivo aqui nos chamava mais a atenção.

Nosso crescimento exponencial apresentado anteriormente, o qual consideramos um pior caso pelos dados verificados, pode não ser o real crecimento de complexidade de ataque ao cifrador, afinal pela verificação apresentada em 6.19, poderiamos ter que o crescimento de 10 para 12 rodadas poderia ser o mesmo de 9 para 11 por exemplo.

Rodadas	1	3	5	7	9	11
média >= 75%	2^1	2^1	2^4	$\mathbf{2^{10}}$	$\mathbf{2^{20}}$	$\mathbf{2^{37}}$
diferença	—	0	3	6	**10**	**17**
mudança	—	—	3	3	4	**7**

Tabela 6.19: Crescimento no ataque χ^2 ao MRC6 com w=32 em 1, 3, 5, 7, 9, e 11 rodadas para a média

Estes dados são aqui apresentados com o intuito de demonstrar que o crescimento como assumimos se baseia nos dados obtidos de uma maneira a tentar obter "por cima", de uma maneira não otimista, o valor do crescimento do experimento, e gostariamos de frizar que até mesmo as considerações envolvendo rodadas pares com rodadas ímpares podem não ser consistentes o suficiente para interligarmos elas de maneira exata - já que existem rotações a mais e mais somas de chaves - porém que o crescimento realmente deve se manter com a interrelação das rodadas como apresentado anteriormente.

6.2.2 Com a imposição de rotações nulas

Assim como a análise apresentada em [4] também fazemos a análise quando considerando as rotações iniciais como sendo zero calculando primeiro um valor que desse rotação nula no ponto considerado e mantendo-o fixo.

De maneira diferente do ataque original não obtemos ganho importante somente com a rotação mantida nula após f_P e então também apresentamos dados de todas as rotações sendo mantidas como nulas na primeira rodada apenas de modo ilustrativo.

6.2.2.1 Com entrada controlada e rotações nulas após f_P e após $f_{\{B,D,F,H,J,K,N,P\}}$ na primeira rodada

Com a primeira rotação em f_P definida em zero temos a rotação nula no sub-bloco M na primeira rodada, porém os dados obtidos são quase os mesmos que os que obtivemos sem fazer esta consideração como podemos verificar na tabela 6.20.

Note que na tabela 6.20 que para 10 rodadas o valor tanto para a média quanto para para 84.14% dos valores passa dos 75% considerados no mesmo ponto em que quando não consideramos nenhuma rotação como sendo nula, então nem chegamos a realizar os testes para 12 rodadas.

Então a título de verificação resolvemos ver o que ocorria quando considerassemos todas as rotações nulas quando da primeira rodada. E com os dados apresentados na tabela 6.21 podemos verificar que o que importa é que os dados nos sub-blocos B,D,F,H,J,K,N e P se mantenham constantes, causando rotação fixa, e não necessariamente causando rotação zero quando da consideração do nosso novo ataque, já que não obtemos nenhum ganho quando destas considerações também.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média
32	**2**	$\mathbf{2^1}$	**126974.0**	**16117.6**	**65534.0**	**110856.4**	**30**	**29**
32	2	2^2	245756.0	52639.8	65532.0	193116.2	30	28
32	2	2^3	479224.0	122818.3	131064.0	356405.7	32	27
32	2	2^4	884720.0	295908.6	262128.0	588811.4	32	23
32	**4**	$\mathbf{2^3}$	**68600.0**	**7715.7**	**65528.0**	**60884.3**	**5**	**5**
32	4	2^4	72944.0	9155.2	65520.0	63788.8	18	18
32	**4**	$\mathbf{2^5}$	**81120.0**	**14839.2**	**65504.0**	**66280.8**	**31**	**11**
32	4	2^6	96448.0	22031.6	77760.0	74416.4	32	10
32	4	2^7	127360.0	36049.4	94080.0	91310.6	32	11
32	6	2^6	65728.0	688.2	65472.0	65039.8	4	4
32	**6**	$\mathbf{2^7}$	**66080.0**	**959.5**	**65408.0**	**65120.5**	**13**	**13**
32	6	2^8	66560.0	964.5	65280.0	65595.5	27	13
32	**6**	$\mathbf{2^9}$	**67464.0**	**1101.5**	**66304.0**	**66362.5**	**32**	**12**
32	6	2^{10}	69252.0	2108.1	66688.0	67143.9	32	11
32	8	2^{12}	65487.0	315.9	64896.0	65171.1	6	13
32	8	2^{13}	65524.5	376.6	64720.0	65147.9	10	13
32	8	2^{14}	65633.5	312.3	65008.0	65321.2	9	17
32	**8**	$\mathbf{2^{15}}$	**65843.4**	**291.8**	**65196.0**	**65551.6**	**20**	**17**
32	**8**	$\mathbf{2^{16}}$	**66245.2**	**399.3**	**65510.0**	**65845.9**	**29**	**15**
32	8	2^{17}	66869.7	608.4	65956.0	66261.3	32	13
32	8	2^{18}	67956.9	882.8	66546.5	67074.1	32	13
32	10	2^{20}	65494.8	347.2	64856.5	65147.7	6	12
32	10	2^{21}	65564.0	384.8	64815.8	65179.2	8	18
32	10	2^{22}	65583.3	418.1	64642.3	65165.2	11	19
32	10	2^{23}	65559.2	313.8	64999.3	65245.3	10	16
32	10	2^{24}	65746.0	262.3	65205.6	65483.7	14	16
32	**10**	$\mathbf{2^{25}}$	**65955.0**	**357.2**	**65199.6**	**65597.8**	**22**	**16**
32	**10**	$\mathbf{2^{26}}$	**66334.0**	**394.1**	**65467.5**	**65939.9**	**30**	**17**
32	10	2^{27}	67035.7	595.0	66140.7	66440.7	32	12
32	10	2^{28}	66933.7	646.0	65930.0	66287.6	32	13

Tabela 6.20: Ataque χ^2 ao MRC6 com w=32 em 2, 4, 6, 8 e 10 rodadas até 2^{28} textos com rotações nulas após f_p na primeira rodada

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média
32	**2**	$\mathbf{2^1}$	**131070.0**	**0.0**	**131070.0**	**131070.0**	**32**	**31**
32	2	2^2	262140.0	0.0	262140.0	262140.0	32	31
32	**4**	$\mathbf{2^3}$	**68600.0**	**7715.7**	**65528.0**	**60884.3**	**5**	**5**
32	4	2^4	71664.0	6901.1	65520.0	64762.9	18	18
32	**4**	$\mathbf{2^5}$	**80224.0**	**10500.1**	**65504.0**	**69723.9**	**31**	**14**
32	4	2^6	95232.0	11964.3	77760.0	83267.7	32	12
32	4	2^7	121376.0	17449.0	94080.0	103927.0	32	17
32	6	2^6	65600.0	503.7	65472.0	65096.3	2	2
32	**6**	$\mathbf{2^7}$	**65888.0**	**779.6**	**65408.0**	**65108.4**	**10**	**9**
32	**6**	$\mathbf{2^8}$	**66416.0**	**620.1**	**65792.0**	**65795.9**	**32**	**13**
32	6	2^9	67392.0	919.6	66048.0	66472.4	32	12
32	8	2^{12}	65436.0	413.4	64608.0	65022.6	8	15
32	8	2^{13}	65508.5	393.8	64816.0	65114.7	8	16
32	8	2^{14}	65520.2	383.6	64656.0	65136.6	9	15
32	8	2^{15}	65754.8	389.5	65096.0	65365.3	14	15
32	**8**	$\mathbf{2^{16}}$	**66150.9**	**416.0**	**65442.0**	**65734.9**	**26**	**13**
32	**8**	$\mathbf{2^{17}}$	**66589.3**	**526.7**	**65450.0**	**66062.6**	**31**	**16**
32	8	2^{18}	67763.7	892.6	66356.5	66871.1	32	12
32	8	2^{19}	70086.1	1576.7	67434.0	68509.4	32	12
32	10	2^{23}	65606.5	283.4	64958.3	65323.1	7	16
32	10	2^{24}	65715.4	379.7	64966.3	65335.7	15	15
32	**10**	$\mathbf{2^{25}}$	**65902.9**	**428.3**	**65323.0**	**65474.6**	**15**	**13**
32	**10**	$\mathbf{2^{26}}$	**66317.0**	**403.1**	**65683.3**	**65913.9**	**31**	**15**
32	10	2^{27}	66981.1	581.0	65648.3	66400.1	31	15
32	10	2^{28}	67060.3	465.2	65847.4	66595.1	32	13
32	10	2^{29}	67160.1	621.9	66174.0	66538.3	32	13

Tabela 6.21: Ataque χ^2 ao MRC6 com w=32 em 2, 4, 6, 8 e 10 rodadas até 2^{29} textos com rotações nulas após $f_{\{B,D,F,H,J,K,N,P\}}$ na primeira rodada

6.2.3 Verificação de características

6.2.3.1 Com entrada controlada, $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$, $\{B, D, F, H, J, K, N, P\} = 0x0$ e rotações nulas após $f_{\{B,D,F,H,J,K,N,P\}}$ forçadas em todas as rodadas

Neste ponto fizemos a consideração do MRC6 com rotações internas nos sub-blocos como sendo nulas para podermos demonstrar algumas características do cifrador de maneira mais eficaz.

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	**1**	$\mathbf{2^1}$	**131070.0**	**0.0**	**131070.0**	**131070.0**	**32**	**31**	**00:00:00:00**
32	1	2^2	262140.0	0.0	262140.0	262140.0	32	31	00:00:00:00
32	**2**	$\mathbf{2^1}$	**131070.0**	**0.0**	**131070.0**	**131070.0**	**32**	**31**	**00:00:00:00**
32	2	2^2	262140.0	0.0	262140.0	262140.0	32	31	00:00:00:00
32	**3**	$\mathbf{2^1}$	**75774.0**	**24176.4**	**65534.0**	**51597.6**	**5**	**5**	**00:00:00:00**
32	3	2^2	107516.0	54726.6	65532.0	52789.4	22	7	00:00:00:00
32	**3**	$\mathbf{2^3}$	**182264.0**	**96862.0**	**131064.0**	**85402.0**	**32**	**5**	**00:00:00:00**
32	3	2^4	306416.0	132335.3	262128.0	174080.7	32	4	00:00:00:00
32	**4**	$\mathbf{2^1}$	**79870.0**	**27526.0**	**65534.0**	**52344.0**	**7**	**6**	**00:00:00:00**
32	**4**	$\mathbf{2^2}$	**110588.0**	**27289.0**	**65532.0**	**83299.0**	**30**	**9**	**00:00:00:00**
32	**4**	$\mathbf{2^3}$	**178680.0**	**31517.8**	**147448.0**	**147162.2**	**32**	**17**	**00:00:00:00**
32	4	2^4	306416.0	28367.2	270320.0	278048.8	32	12	00:00:00:00
32	**5**	$\mathbf{2^1}$	**67582.0**	**11585.2**	**65534.0**	**55996.8**	**1**	**1**	**00:00:00:00**
32	5	2^2	79868.0	23449.1	65532.0	56418.9	11	10	00:00:00:00
32	**5**	$\mathbf{2^3}$	**94200.0**	**21624.0**	**65528.0**	**72576.0**	**28**	**15**	**00:00:00:00**
32	**5**	$\mathbf{2^4}$	**127472.0**	**27833.9**	**98288.0**	**99638.1**	**32**	**14**	**00:00:00:00**
32	5	2^5	189536.0	37945.3	151520.0	151590.7	32	8	00:00:00:00
32	**6**	$\mathbf{2^1}$	**67582.0**	**11585.2**	**65534.0**	**55996.8**	**1**	**1**	**00:00:00:00**
32	6	2^2	71676.0	12994.4	65532.0	58681.6	6	6	00:00:00:00
32	**6**	$\mathbf{2^3}$	**90616.0**	**17648.3**	**65528.0**	**72967.7**	**28**	**14**	**00:00:00:00**
32	**6**	$\mathbf{2^4}$	**131056.0**	**25228.0**	**90096.0**	**105828.0**	**32**	**16**	**00:00:00:00**
32	6	2^5	198240.0	20856.0	167904.0	177384.0	32	14	00:00:00:00

Tabela 6.22: Ataque χ^2 ao MRC6 sem rotações com w=32 em 1, 2, 3, 4, 5 e 6 rodadas até 2^5 textos com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	7	2^1	65534.0	0.0	65534.0	65534.0	0	31	00:00:00:00
32	**7**	$\mathbf{2^2}$	**67580.0**	**8058.8**	**65532.0**	**59521.2**	**2**	**2**	**00:00:00:00**
32	7	2^3	70648.0	7715.7	65528.0	62932.3	10	10	00:00:00:00
32	**7**	$\mathbf{2^4}$	**80624.0**	**8854.7**	**65520.0**	**71769.3**	**29**	**19**	**00:00:00:00**
32	**7**	$\mathbf{2^5}$	**96864.0**	**12978.1**	**77792.0**	**83885.9**	**32**	**13**	**00:00:00:00**
32	7	2^6	127808.0	9978.2	108480.0	117829.8	32	18	00:00:00:00
32	8	2^1	65534.0	0.0	65534.0	65534.0	0	31	00:00:00:00
32	**8**	$\mathbf{2^2}$	**67580.0**	**8058.8**	**65532.0**	**59521.2**	**2**	**2**	**00:00:00:00**
32	8	2^3	72184.0	10073.5	65528.0	62110.5	11	11	00:00:00:00
32	**8**	$\mathbf{2^4}$	**80112.0**	**9243.5**	**65520.0**	**70868.5**	**28**	**20**	**00:00:00:00**
32	**8**	$\mathbf{2^5}$	**98144.0**	**10429.0**	**77792.0**	**87715.0**	**32**	**18**	**00:00:00:00**
32	8	2^6	131136.0	15201.2	110528.0	115934.8	32	10	00:00:00:00
32	9	2^1	65534.0	0.0	65534.0	65534.0	0	31	00:00:00:00
32	9	2^2	65532.0	0.0	65532.0	65532.0	0	31	00:00:00:00
32	**9**	$\mathbf{2^3}$	**68088.0**	**6044.1**	**65528.0**	**62043.9**	**5**	**4**	**00:00:00:00**
32	9	2^4	70384.0	5036.7	65520.0	65347.3	17	16	00:00:00:00
32	**9**	$\mathbf{2^5}$	**74720.0**	**5602.6**	**65504.0**	**69117.4**	**30**	**10**	**00:00:00:00**
32	**9**	$\mathbf{2^6}$	**82624.0**	**6935.4**	**69568.0**	**75688.6**	**32**	**16**	**00:00:00:00**
32	9	2^7	97600.0	5474.0	86912.0	92126.0	32	15	00:00:00:00
32	10	2^1	65534.0	0.0	65534.0	65534.0	0	31	00:00:00:00
32	10	2^2	65532.0	0.0	65532.0	65532.0	0	31	00:00:00:00
32	**10**	$\mathbf{2^3}$	**66552.0**	**4029.4**	**65528.0**	**62522.6**	**2**	**2**	**00:00:00:00**
32	10	2^4	68592.0	4534.9	65520.0	64057.1	11	11	00:00:00:00
32	**10**	$\mathbf{2^5}$	**73568.0**	**6109.5**	**65504.0**	**67458.5**	**27**	**16**	**00:00:00:00**
32	**10**	$\mathbf{2^6}$	**81984.0**	**6741.2**	**69568.0**	**75242.8**	**32**	**14**	**00:00:00:00**
32	10	2^7	99200.0	5247.2	86912.0	93952.8	32	16	00:00:00:00

Tabela 6.23: Ataque χ^2 ao MRC6 sem rotações com w=32 em 7, 8, 9 e 10 rodadas até 2^7 textos com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	11	2^2	65532.0	0.0	65532.0	65532.0	0	31	00:00:00:00
32	**11**	$\mathbf{2^3}$	**66040.0**	**2896.3**	**65528.0**	**63143.7**	**1**	**1**	**00:00:00:00**
32	11	2^4	66544.0	2752.6	65520.0	63791.4	4	4	00:00:00:00
32	11	2^5	68448.0	2798.3	65504.0	65649.7	19	19	00:00:00:00
32	**11**	$\mathbf{2^6}$	**70080.0**	**2703.0**	**65472.0**	**67377.0**	**29**	**12**	**00:00:00:00**
32	**11**	$\mathbf{2^7}$	**74016.0**	**2465.6**	**69504.0**	**71550.4**	**32**	**16**	**00:00:00:00**
32	11	2^8	82224.0	3441.6	76032.0	78782.4	32	16	00:00:00:00
32	**12**	$\mathbf{2^3}$	**66040.0**	**2896.3**	**65528.0**	**63143.7**	**1**	**1**	**00:00:00:00**
32	12	2^4	66288.0	2426.0	65520.0	63862.0	3	3	00:00:00:00
32	12	2^5	67808.0	2931.1	65504.0	64876.9	14	13	00:00:00:00
32	**12**	$\mathbf{2^6}$	**70080.0**	**2896.3**	**65472.0**	**67183.7**	**29**	**14**	**00:00:00:00**
32	**12**	$\mathbf{2^7}$	**74336.0**	**3080.1**	**69504.0**	**71255.9**	**32**	**15**	**00:00:00:00**
32	12	2^8	83120.0	3159.2	76544.0	79960.8	32	15	00:00:00:00
32	**13**	$\mathbf{2^3}$	**66040.0**	**2896.3**	**65528.0**	**63143.7**	**1**	**1**	**00:00:00:00**
32	13	2^4	65776.0	1448.2	65520.0	64327.8	1	1	00:00:00:00
32	13	2^5	65888.0	1213.0	65504.0	64675.0	3	3	00:00:00:00
32	13	2^6	66304.0	1146.7	65472.0	65157.3	12	12	00:00:00:00
32	**13**	$\mathbf{2^7}$	**67648.0**	**1311.8**	**65408.0**	**66336.2**	**29**	**12**	**00:00:00:00**
32	**13**	$\mathbf{2^8}$	**69408.0**	**1287.0**	**67328.0**	**68121.0**	**32**	**14**	**00:00:00:00**
32	13	2^9	73200.0	1484.1	70656.0	71715.9	32	14	00:00:00:00
32	**14**	$\mathbf{2^3}$	**66040.0**	**2896.3**	**65528.0**	**63143.7**	**1**	**1**	**00:00:00:00**
32	14	2^4	66288.0	2426.0	65520.0	63862.0	3	3	00:00:00:00
32	14	2^5	66272.0	2426.0	65504.0	63846.0	4	4	00:00:00:00
32	14	2^6	66752.0	1624.3	65472.0	65127.7	15	14	00:00:00:00
32	**14**	$\mathbf{2^7}$	**67360.0**	**1173.6**	**65408.0**	**66186.4**	**30**	**16**	**00:00:00:00**
32	**14**	$\mathbf{2^8}$	**69456.0**	**1288.1**	**66816.0**	**68167.9**	**32**	**14**	**00:00:00:00**
32	14	2^9	73048.0	1210.6	71168.0	71837.4	32	14	00:00:00:00

Tabela 6.24: Ataque χ^2 ao MRC6 sem rotações com w=32 em 11, 12, 13 e 14 rodadas até 2^9 textos com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	15	2^5	65632.0	724.1	65504.0	64907.9	1	1	00:00:00:00
32	15	2^6	65600.0	503.7	65472.0	65096.3	2	2	00:00:00:00
32	15	2^7	65760.0	616.0	65408.0	65144.0	9	9	00:00:00:00
32	**15**	$\mathbf{2^8}$	**66176.0**	**596.0**	**65280.0**	**65580.0**	**28**	**16**	**00:00:00:00**
32	**15**	$\mathbf{2^9}$	**67032.0**	**730.2**	**65792.0**	**66301.8**	**32**	**17**	**00:00:00:00**
32	15	2^{10}	68276.0	704.7	66688.0	67571.3	32	18	00:00:00:00
32	16	2^5	65760.0	1007.3	65504.0	64752.7	2	2	00:00:00:00
32	16	2^6	65728.0	862.6	65472.0	64865.4	3	3	00:00:00:00
32	**16**	$\mathbf{2^7}$	**66016.0**	**681.2**	**65408.0**	**65334.8**	**16**	**16**	**00:00:00:00**
32	16	2^8	66448.0	1025.9	65280.0	65422.1	27	11	00:00:00:00
32	**16**	$\mathbf{2^9}$	**67272.0**	**852.1**	**65792.0**	**66419.9**	**32**	**14**	**00:00:00:00**
32	16	2^{10}	68580.0	697.9	67200.0	67882.1	32	15	00:00:00:00
32	17	2^{26}	65564.6	319.5	64835.4	65245.1	8	17	00:00:21:58
32	17	2^{27}	65601.0	334.7	64832.0	65266.3	11	14	00:00:43:57
32	17	2^{28}	65657.0	358.8	64719.1	65298.3	12	17	00:01:27:52
32	17	2^{29}	65559.4	409.2	64770.2	65150.2	7	13	00:02:55:49
32	17	2^{30}	65550.5	339.7	64677.0	65210.8	9	15	00:05:51:32
32	18	2^{30}	65439.0	288.3	64924.1	65150.7	5	14	*00:06:08:00*
32	18	2^{31}	65448.2	363.7	64638.1	65084.4	4	15	*00:12:16:00*
32	18	2^{32}	65421.4	400.5	64568.9	65020.9	6	15	*01:00:32:00*
32	18	2^{33}	65460.3	337.7	64765.6	65122.6	6	16	*02:01:04:00*
32	18	2^{34}	65537.3	352.5	65053.2	65184.7	5	13	*04:02:08:00*

Tabela 6.25: Ataque χ^2 ao MRC6 sem rotações com w=32 em 15, 16, 17 e 18 rodadas até 2^{34} textos com $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$ e $\{B, D, F, H, J, K, N, P\} = 0x0$

Como demonstrado nas tabelas 6.22, 6.23, 6.24 e 6.25 o cifrador realmente aparenta ser "mal projetado" para a sua utilização em até 16 rodadas se considerarmos as rotações internas aos sub-blocos sempre fixas - sendo que ainda não temos garantias de um bom funcionamento a partir das 16 rodadas - já que o efeito da mistura dos blocos só começa a ter uma força maior depois das mesmas.

O ponto principal é que desconsiderando *parte* da iteração entre os sub-blocos, sendo somente a parte que define a rotação, o cifrador já aparenta demontrar fraquezas acentuadas para até 16 rodadas. Afinal se a rotação fosse fixa

poderiamos ter um maior controle sobre as características lineares como demonstrado nos dados do experimento apresentado acima.

Note que conseguimos chegar em dados para 18 rodadas - sem informações expressivas ainda - mas sem um crescimento tão gigantesco como em até 16 rodadas, ou seja, começamos a verificar que para 18 rodadas a iteração da mistura dos sub-blocos começa a ganhar força quando das considerações do nosso ataque neste caso em específico.

6.2.3.2 Com entrada controlada, $\{B, D,\ F, H, J, K, N, P\} = 0x0$ e rotações nulas após $f_{\{B,D,F,H,J,K,N,P\}}$ forçadas em todas as rodadas

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	**1**	$\mathbf{2^1}$	**131070.0**	**0.0**	**131070.0**	**131070.0**	**32**	**31**	**00:00:00:00**
32	1	2^2	262140.0	0.0	262140.0	262140.0	32	31	00:00:00:00
32	**2**	$\mathbf{2^3}$	**131064.0**	**0.0**	**131064.0**	**131064.0**	**32**	**31**	**00:00:00:00**
32	2	2^4	262128.0	0.0	262128.0	262128.0	32	31	00:00:00:00
32	**3**	$\mathbf{2^3}$	**152568.0**	**26134.2**	**131064.0**	**126433.8**	**32**	**6**	**00:00:00:00**
32	3	2^4	276720.0	17500.1	262128.0	259219.9	32	10	00:00:00:00
32	**4**	$\mathbf{2^3}$	**81400.0**	**16375.7**	**65528.0**	**65024.3**	**19**	**19**	**00:00:00:00**
32	**4**	$\mathbf{2^4}$	**111600.0**	**12614.0**	**90096.0**	**98986.0**	**32**	**17**	**00:00:00:00**
32	4	2^5	183776.0	19120.2	151520.0	164655.8	32	15	00:00:00:00
32	**5**	$\mathbf{2^3}$	**85496.0**	**15974.2**	**65528.0**	**69521.8**	**25**	**10**	**00:00:00:00**
32	**5**	$\mathbf{2^4}$	**118512.0**	**15429.2**	**90096.0**	**103082.8**	**32**	**15**	**00:00:00:00**
32	5	2^5	190944.0	23397.1	151520.0	167546.9	32	16	00:00:00:00
32	**6**	$\mathbf{2^3}$	**73208.0**	**10998.1**	**65528.0**	**62209.9**	**12**	**12**	**00:00:00:00**
32	**6**	$\mathbf{2^4}$	**79344.0**	**9406.7**	**65520.0**	**69937.3**	**29**	**16**	**00:00:00:00**
32	**6**	$\mathbf{2^5}$	**96096.0**	**10350.9**	**77792.0**	**85745.1**	**32**	**14**	**00:00:00:00**
32	6	2^6	126912.0	10786.9	108480.0	116125.1	32	16	00:00:00:00

Tabela 6.26: Ataque χ^2 ao MRC6 sem rotações com w=32 em 1, 2, 3, 4, e 6 rodadas até 2^6 textos com $\{B, D, F, H, J, K, N, P\} = 0x0$

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	**7**	$\mathbf{2^3}$	**73208.0**	**13792.5**	**65528.0**	**59415.5**	**10**	**10**	**00:00:00:00**
32	**7**	$\mathbf{2^4}$	**81392.0**	**9745.8**	**65520.0**	**71646.2**	**28**	**21**	**00:00:00:00**
32	**7**	$\mathbf{2^5}$	**94688.0**	**10233.4**	**81888.0**	**84454.6**	**32**	**14**	**00:00:00:00**
32	7	2^6	131008.0	9577.8	116672.0	121430.2	32	15	00:00:00:00
32	**8**	$\mathbf{2^3}$	**68088.0**	**6044.1**	**65528.0**	**62043.9**	**5**	**5**	**00:00:00:00**
32	8	2^4	69872.0	5879.6	65520.0	63992.4	13	12	00:00:00:00
32	**8**	$\mathbf{2^5}$	**72800.0**	**4621.7**	**65504.0**	**68178.3**	**29**	**17**	**00:00:00:00**
32	**8**	$\mathbf{2^6}$	**81920.0**	**5367.9**	**73664.0**	**76552.1**	**32**	**12**	**00:00:00:00**
32	8	2^7	97472.0	7233.5	86912.0	90238.5	32	14	00:00:00:00
32	**9**	$\mathbf{2^3}$	**66040.0**	**2896.3**	**65528.0**	**63143.7**	**1**	**1**	**00:00:00:00**
32	9	2^4	68336.0	3953.1	65520.0	64382.9	11	11	00:00:00:00
32	**9**	$\mathbf{2^5}$	**72544.0**	**5014.9**	**65504.0**	**67529.1**	**30**	**13**	**00:00:00:00**
32	**9**	$\mathbf{2^6}$	**81152.0**	**6571.9**	**69568.0**	**74580.1**	**32**	**15**	**00:00:00:00**
32	9	2^7	95552.0	5109.7	85888.0	90442.3	32	16	00:00:00:00
32	**10**	$\mathbf{2^4}$	**66288.0**	**2426.0**	**65520.0**	**63862.0**	**3**	**3**	**00:00:00:00**
32	10	2^5	67424.0	2749.5	65504.0	64674.5	12	12	00:00:00:00
32	**10**	$\mathbf{2^6}$	**68736.0**	**2645.3**	**65472.0**	**66090.7**	**26**	**13**	**00:00:00:00**
32	**10**	$\mathbf{2^7}$	**72704.0**	**3182.9**	**68480.0**	**69521.1**	**32**	**12**	**00:00:00:00**
32	10	2^8	81008.0	2710.0	75520.0	78298.0	32	15	00:00:00:00
32	**11**	$\mathbf{2^4}$	**66544.0**	**2752.6**	**65520.0**	**63791.4**	**4**	**4**	**00:00:00:00**
32	11	2^5	67808.0	3279.7	65504.0	64528.3	13	13	00:00:00:00
32	**11**	$\mathbf{2^6}$	**69184.0**	**2822.4**	**65472.0**	**66361.6**	**28**	**17**	**00:00:00:00**
32	**11**	$\mathbf{2^7}$	**73344.0**	**2999.6**	**68480.0**	**70344.4**	**32**	**15**	**00:00:00:00**
32	11	2^8	80944.0	3007.6	73984.0	77936.4	32	14	00:00:00:00
32	12	2^5	65760.0	1007.3	65504.0	64752.7	2	2	00:00:00:00
32	**12**	$\mathbf{2^6}$	**66624.0**	**1555.1**	**65472.0**	**65068.9**	**14**	**13**	**00:00:00:00**
32	**12**	$\mathbf{2^7}$	**67776.0**	**1435.0**	**65408.0**	**66341.0**	**30**	**12**	**00:00:00:00**
32	**12**	$\mathbf{2^8}$	**69472.0**	**1707.7**	**66816.0**	**67764.3**	**32**	**13**	**00:00:00:00**
32	12	2^9	73064.0	1443.6	70912.0	71620.4	32	14	00:00:00:00

Tabela 6.27: Ataque χ^2 ao MRC6 sem rotações com w=32 em 7, 8, 9, 10, 11, e 12 rodadas até 2^9 textos com $\{B, D, F, H, J, K, N, P\} = 0x0$

A partir dos dados apresentados nas tabelas 6.26, 6.27 e 6.28 podemos então verificar que mesmo considerando as rotações sempre sendo nulas com a consideração de $lsb_5(\{A, C, E, G, I, K, M, O\}) = 0x0$, como apresentado anteriormente, ganhamos duas rodada no ataque - já que em uma rodada os textos do sub-bloco passam ilesos -, e podemos notar a possibilidade do ataque subindo de 14 para 16 rodadas quando destas considerações, do mesmo modo como demonstrado no ataque ao RC6 em [4].

chaves	rodadas	textos	média χ^2	desvio padrão	minimo χ^2	min 68.27%	#>=75%	#>=média	dd:hh:mm:ss
32	**13**	$\mathbf{2^5}$	**66016.0**	**1376.3**	**65504.0**	**64639.7**	**4**	**4**	**00:00:00:00**
32	13	2^6	66432.0	1374.8	65472.0	65057.2	12	11	00:00:00:00
32	**13**	$\mathbf{2^7}$	**67744.0**	**1587.1**	**65408.0**	**66156.9**	**30**	**13**	**00:00:00:00**
32	**13**	$\mathbf{2^8}$	**69568.0**	**1607.3**	**66304.0**	**67960.7**	**32**	**14**	**00:00:00:00**
32	13	2^9	73416.0	1495.2	70656.0	71920.8	32	15	00:00:00:00
32	**14**	$\mathbf{2^6}$	**65920.0**	**1005.2**	**65472.0**	**64914.8**	**6**	**5**	**00:00:00:00**
32	14	2^7	66240.0	799.0	65408.0	65441.0	20	19	00:00:00:00
32	14	2^8	66464.0	866.9	65280.0	65597.1	27	11	00:00:00:00
32	**14**	$\mathbf{2^9}$	**67312.0**	**662.9**	**65792.0**	**66649.1**	**32**	**18**	**00:00:00:00**
32	14	2^{10}	68724.0	689.2	67072.0	68034.8	32	15	00:00:00:00
32	**15**	$\mathbf{2^6}$	**65792.0**	**755.5**	**65472.0**	**65036.5**	**5**	**5**	**00:00:00:00**
32	15	2^7	65856.0	777.6	65408.0	65078.4	10	10	00:00:00:00
32	15	2^8	66496.0	1030.2	65280.0	65465.8	26	15	00:00:00:00
32	**15**	$\mathbf{2^9}$	**67176.0**	**930.7**	**65280.0**	**66245.3**	**31**	**15**	**00:00:00:00**
32	**15**	$\mathbf{2^{10}}$	**68360.0**	**660.7**	**66816.0**	**67699.3**	**32**	**14**	**00:00:00:00**
32	15	2^{11}	71408.0	621.6	70336.0	70786.4	32	17	00:00:00:00
32	16	2^{29}	65542.5	394.6	64858.7	65147.9	7	17	00:02:46:28
32	16	2^{30}	65451.8	399.0	64517.1	65052.8	6	17	00:00:00:00
32	16	2^{31}	65571.4	323.6	64767.1	65247.9	10	15	00:00:00:00
32	16	2^{32}	65557.3	370.1	64906.9	65187.3	8	16	00:00:00:00
32	16	2^{33}	65528.4	422.5	64880.1	65105.9	12	16	00:00:00:00
32	16	2^{34}	65512.8	431.1	64748.3	65081.8	10	19	00:00:00:00

Tabela 6.28: Ataque χ^2 ao MRC6 sem rotações com w=32 em 13, 14, 15 e 16 rodadas até 2^{34} textos com $\{B, D, F, H, J, K, N, P\} = 0x0$

6.2.3.3 Considerações pertinentes aos ataques para mais de 10 rodadas

Com base nas inferências e testes realizados neste estudo para que possamos ter certeza sobre algumas considerações quando dos testes não realizados na pratica precisamos nos utilizar de considerações para termos certeza se poderiamos considerar nosso ataque para além das 11 rodadas que testamos.

Acreditamos, como demonstrado no início do capítulo, que nosso ataque seria eficaz como previsto em pelo menos até 12 rodadas, e a partir deste ponto não temos como garantir previsões sobre o crescimento de complexidade do MRC6 com base nos nossos dados.

Como demonstrado anteriormente o cifrador parece precisar de mais do que 16 rodadas para garantir, com uma aproximação de crescimento levando em consideração um pior caso previsto, complexidade acima de 2^{128}, não passando de 2^{512} nem mesmo quando da utilização de 20 rodadas.

Capítulo 7

Conclusões e Trabalhos Futuros

Assim como nas considerações feitas sobre o RC6 pudemos demonstrar o comportamento do MRC6 em relação à alguns fatores que poderiam por ele em risco, mas com a realização dos mesmos testes de χ^2 sobre as informações consideradas quando do estudo original feitas no início deste estudo tivemos a impressão que o MRC6 poderia mesmo ser mais forte do que o RC6 quando lidando com um ataque de diferenciação de uma amostra aleatória, o que faz muito sentido quando levamos em consideração estarmos comparando um cifrador de 128 bits com um de 512 bits. Porém pelo mesmo fato de estarmos lidando com um cifrador com blocos de 512 bits pudemos trabalhar com muito mais informações, em especial por ele manter os sub-blocos de tamanho 32 bits como no RC6. Novos testes permitiram considerações extras, que não teriam como ser aplicadas quando do ataque ao RC6 com 20 rodadas, mas que no caso do MRC6 para 15 rodadas seriam possíveis, mesmo considerando complexidade máxima de 2^{128}.

Com a adição de novas considerações pudemos até mesmo explicar o porque de certos comportamentos quando do estudo original ao RC6 se manterem no nosso estudo em um caso mais simples que o considerado no RC6 - de modo a demonstrarmos que tais considerações tinham um efeito diferente quando das novas considerações em relação às originais -, e com nosso novo ataque demonstramos que já obtinhamos os resultados relativos às rotações nulas mantida no ataque ao RC6

sem que as mantivéssemos, mostrando então que manter os blocos que causavam a rotação nula simplesmente em zero e mantendo a rotação estabilizada já nos levava no caminho dos mesmo resultados.

O fato mais interessante é que apesar de o cifrador MRC6 ser definido como padrão para a utilização de chaves de 128 bits ele é na verdade um cifrador de 512 bits, e como tal deveria ter uma complexida de aproximadamente 2^{512} para sua resolução. Ou que até mesmo com um número inferior a este de textos não pudessemos de maneira nenhuma detectar indícios da cifragem em relação à uma geração aleatória. Só que no estudo apresentado chegamos a definir que a diferenciação seria possível quando da utilização de 15 rodadas com a verificação de 2^{116} textos quando considerando 84.14% dos dados analisados, ou seja, muito menos do que deveria ser possível pela definição do cifrador.

Um detalhe importante é que mesmo com a consideração de alguns bits como sendo fixos na escolha dos textos em claro de entrada ainda teriamos 2^{216} possibilidades para trabalharmos, e com base nos dados obtidos temos que o cifrador teria que por exemplo trabalhar com mais de 16 rodadas, considerando complexidade de 2^{128}, para que não fosse possível a diferenciação dos dados cifrados de uma amostra aleatória quando considerando estes 2^{216} textos em claro escolhidos. Ou seja, se nem mesmo incluíssemos nenhum novo dado nas considerações do ataque não teriamos como considerar este cifrador como sendo seguro com menos do que cerca de 22 rodadas.

A conclusão que temos em relação ao crescimento do MRC6 em relação ao RC6 é que para considerarmos um cifrador com um número tão grande de sub-blocos a serem trabalhados temos que ter uma melhor escolha quando da definição das operações que interrelacionam tantos sub-blocos, ou então corremos um sério risco de não termos um bom cifrador.

Quanto à trabalhos futuros esperamos em breve ter implementações dos ataques que lidem com “threads” independentes para cada uma das chaves analisadas, de maneira que possamos utilizar uma maior performance indepedentemente do tipo de processador utilizado para o processamento dos dados, bem como

a verificação de novas possibilidades para acelerarmos o desempenho com menos processamento mantendo mais dados na memória.

Propomos também a adaptação de ataques com o mesmo tipo de controle para os cifradores RC5 e RC6 - e então o uso das otimizações citadas -, os quais esperamos em breve podermos lidar com até mais do que os 2^{42} textos como estimamos que poderiamos lidar com a otimização do algorítmo de verificação do teste χ^2 para o MRC6.

Além da adaptação para "threads" também vamos passar da utilização de duas placas de memória RAM DDR2 a 800Mhz em paralelo para duas placas de vídeo em paralelo com DDR3 a 1.7Ghz para tentarmos ganhar performance quando do acesso a memória.

Referências Bibliográficas

[1] Nawal A. El-fishawy, Talat E. El-Danaf, and Osama M. Abu zaid. A modification of RC6 block cipher algorithm for data security (MRC6). In *International Conference on Electrical, Electronic and Computer Engineering, ICEEC*, Cairo, Egypt, September 2004.

[2] Scott Contini, Ronald L. Rivest, M.J.B. Robshaw, and Yiqun Lisa Yin. The security of the RC6 block cipher, v.1.0. `ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/security.pdf`, August 1998. Technical report, RSA Security.

[3] Douglas R. Stinson. *Cryptography: Theory and Practice (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 3th edition, November 2005. ISBN: 1584885084.

[4] Lars R. Knudsen and Willi Meier. Correlations in RC6 with a reduced number of rounds. In B. Schneier, editor, *Proceedings of Fast Software Encryption: 7th International Workshop, FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2000. ISBN:3-540-41728-1.

[5] Ron L. Rivest, Matthew J. B. Robshaw, Raymond Sidney, and Yiqun Lisa Yin. The RC6 block cipher. In *1st AES Conference*, California, USA, August 1998. `ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf`.

[6] R.L. Rivest. The RC5 encryption algorithm. In *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption, 2nd International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer-Verlag, 1995.

[7] AES. The advanced encryption standard development process, 1997. `http://csrc.nist.gov/encryption/aes/`.

[8] NIST. Data encryption standard (des). FIPS PUB 46-2 Federal Information Processing Standard Publication 46-2, December 1993.

[9] *The Design of Rijndael - AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer-Verlag, 2002. ISBN:3540425802.

[10] E. Biham and A. Shamir. A differential cryptanalysis of the data encryption standard. Springer-verlag, 1993.

[11] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994. ISBN: 978-3-540-57600-6.

[12] O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. Report in the aes candidates, 1999. `http://csrc.nist.gov/encrypt/aes/round1/conf2/papers/baudron1.pdf`.

[13] A. Ragab, N. Ismail, and O. Faragallah. Enhancement and implementation of RC6 block cipher for data security. In *Electronic Engineering Bulletin*, number 21, 2001.

[14] Nawal A. El-fishawy and Osama M. Abu zaid. Quality of encryption measurement of bitmap images with RC6, MRC6, and rijndael block cipher algorithms. In *Intl Journal of Network Security*, volume 5, no.3, pages 241–251, November 2007.

[15] Gonzalo Alvarez, Dolores de la Guia, Fausto Montoya, and Alberto Peinado. Akelarre: a new block cipher algorithm. In *Proceedings of SAC'96, Third Annual Workshop on Selected Areas in Cryptography*, pages 1–14, Queen's University, Kingston, Ontario, 1996.

[16] Lars R. Knudsen and Vincent Rijmen. Ciphertext-only attack on akelarre. In *Cryptologia*, volume XXIV, no. 2, pages 135–147, April 2000.

[17] J. Nakahara Jr. and Daniel Santana de Freitas. Cryptanalysis of ake98. In *INDOCRYPT 2004, 5th International Conference on Cryptology in India*, volume 3348 of *Lecture Notes In Computer Science*, pages 162–174. Springer-Verlag, 2004.

[18] RSA Laboratories - The Security Division of EMC. http://www.rsa.com/rsalabs/.

[19] Burton S. Kaliski Jr. and Yiqun Lisa Yin. On differential and linear cryptanalysis of the RC5 encryption algorithm. In *Advances in Cryptology - CRYPT0 '95*, volume 963 of *Lecture Notes in Computer Science*, pages 171–184. Springer-Verlag, 1995. ISBN: 978-3-540-60221-7.

[20] Lars R. Knudsen and Willi Meier. Improved differential attacks on RC5. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 216–228, 1996. ISBN: 3-540-61512-1.

[21] Alex Biryukov and Eyal Kushilevitz. Improved cryptanalysis of RC5. In *Advances in Cryptology - EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 85–99, 1998. ISBN: 978-3-540-64518-4.

[22] Ali Aydin Selçuk. New results in linear cryptanalysis of RC5. In *Proceedings of 5th International Workshop on Fast Software Encryption*, volume 1372 of *Lecture Notes In Computer Science*, pages 1–16, Paris, France, March 1998. Springer-Verlag. ISBN:3-540-64265-X.

[23] National Institute for Standards and Technology. Federal information processing standard 180-2, secure hash standard, August 2002. supersed FIPS 180-1.

[24] Lars R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption*, volume 1008, pages 96–211, 1995.

[25] X. Lai. Higher order derivatives and differential cryptanalysis. In *In: Communication and Cryptography, Two Sides of One Tapestry*, pages 85–99. Kluwer Academic Publishers, 1994.

[26] X. Lai, J. L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In D. W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer-Verlag, 1992.

[27] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer-Verlag, March 2003. ISBN: 978-3-540-43101-5.

[28] Serge Vaudenay. An experiment on DES statistical cryptanalysis. In *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pages 139–147, New Delhi, India, March 1996. ACM Press. ISBN:0-89791-829-0.

[29] H. Handschuh and H. Gilbert. Cryptanalysis of the SEAL encryption algorithm. In *Proceedings of the Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1997.

[30] John Kelsey, Bruce Schneier, and David Wagner. Mod n cryptanalysis, with applications against RC5P and M6. In *Fast Software Encryption: 6th International Workshop, FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 1999. ISBN: 978-3-540-66226-6.

[31] Takeshi Shimoyama, Kiyofumi Takeuchi, and Juri Hayakawa. Correlation attack to the block cipher RC5 and the simplified variants of RC6. In *Proceedings of 3th AES Candidate Conference (AES3)*, April 2000. `http://csrc.nist.gov/encryption/aes/round2/conf3/papers/36-tshimoyama.pdf`.

[32] D. Lane, J. Lu, C. Peres, and E. Zitek. Online statistics: An interactive multimedia course of study, 2007.

[33] Henri Gilbert, Helena Handschuh, Antoine Joux, and Serge Vaudenay. A statistical attack on RC6. In *Proceedings of Fast Software Encryption: 7th International Workshop, FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 64–74. Spring-Verlag, April 2000. ISBN:3-540-41728-1.

[34] Donald E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley Professional, October 1981.

[35] NIST. Advanced encryption standard (AES). FIPS PUB 197 Federal Information Processing Standard Publication 197, November 2001. U.S. Department of Commerce.