Uma Arquitetura Dinamicamente Reconfigurável para Criptografia Robusta contra Ataques por Canais Colaterais

Daniel Mesquita
INESC-ID
mesquita@inesc-id.pt

Fernando Moraes, Ney Calazans
FACIN-PUCRS
{moraes, calazans}@inf.pucrs.br

Lionel Torres, Michel Robert
LIRMM
{torres, robert}@lirmm.fr

Abstract

Este trabalho explora a técnica de reconfiguração dinâmica como um mecanismo capaz de incrementar desempenho, flexibilidade e segurança de sistemas criptográficos. A arquitetura dinamicamente reconfigurável proposta é resistente a um dos mais perigosos ataques por canais laterais, o ataque por análise diferencial de consumo. Tais ataques e algumas contramedidas são brevemente descritos neste artigo, cujo foco está na demonstração do interesse da reconfiguração dinâmica para a criptografia. Resultados de simulação e de síntese corroboram essa hipótese.

1. Introdução

O ponto principal deste trabalho concerne o uso da reconfiguração dinâmica como técnica capaz de fornecer flexibilidade, permitir alto desempenho, e, sobretudo, de incrementar a robustez de sistemas criptográficos. Flexibilidade é uma das características evidentes da reconfiguração, e uma necessidade básica na área da segurança, pois as técnicas de criptanálise evoluem tão rápido quanto os métodos de criptografia. Contudo existem atualmente novas técnicas de ataques que quebram o paradigma da criptanálise convencional.

Em 1998 um ataque baseado numa informação colateral foi proposto [1]. Este tipo de ataque utiliza informações como tempo de execução, emissões eletromagnéticas, temperatura, consumo de potência, entre outras. Ataques que utilizam essas informações são chamados Ataques por Canais Colaterais (do inglês *Side Channel Attacks* - SCA), e podem ser utilizados para se descobrir chaves secretas gravadas em hardware criptográfico.

Por exemplo, a potência consumida por um mecanismo criptográfico tem uma estrita relação com os dados calculados pelo dispositivo.

Essa característica é devida ao fato que as portas lógicas consomem de forma diferente quando comutam de 0 (zero) para 1 (um), ou permanecem com a saída em 1 (um) ou em 0 (zero). Conseqüentemente, um processo de criptografia gera uma assinatura de potência dependendo do texto que está sendo criptografado. Se o dispositivo criptográfico não possui nenhuma proteção especial (contramedida), um ataque por análise simples de consumo (*Single Power Analysis* – SPA) é possível. Mas mesmo com algumas proteções um ataque por análise diferencial ainda pode ser eficiente (*Differential Power Analysis* –DPA).

Este artigo primeiramente aborda aspectos relativos ao ataque DPA e mostra algumas contramedidas. Em seguida a *Leak Resistant Reconfigurable Architecture* (LR^2A), uma arquitetura reconfigurável resistente ao ataque DPA é apresentada. Então uma discussão sobre aspectos de segurança da LR^2A é realizada, considerando a base aritmética desta abordagem. Finalmente resultados, conclusões e trabalhos futuros são apresentados.

2. O Ataque DPA

O ataque por análise diferencial de consumo de potência (*Differential Power Analysis* –DPA) consiste em um estudo estatístico e no uso de métodos de correção de erro tendo como objetivo a recuperação de chaves secretas em implementações em hardware de algoritmos criptográficos [1].

A maior parte do consumo de um circuito integrado é devida às portas lógicas e à capacitância parasítica dos fios. Contudo a parte variante do consumo é dada pelos dados computados. Desta forma o ataque DPA possibilita explorar a co-relação entre dados e o consumo do dispositivo criptográfico durante a execução, isto sem a necessidade de conhecimento de detalhes de implementação.

Para um ataque típico um adversário realiza uma amostragem do consumo de potência do dispositivo alvo através de algumas centenas de cálculos criptográficos. Essas curvas de consumo podem ser obtidas com um osciloscópio de alta velocidade e resolução. A Figura 1 ilustra uma plataforma de ataque contra um sistema criptográfico implementado em um FPGA em uma placa de prototipação.

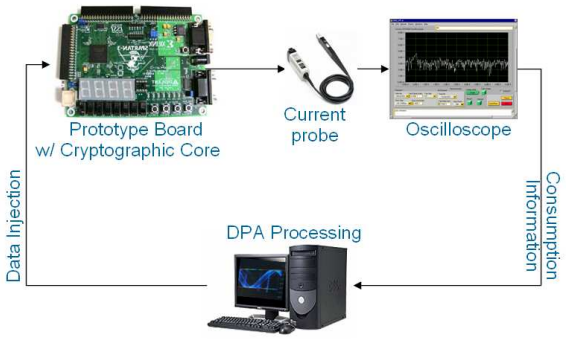

Figura 1 - Uma visão geral de uma plataforma de ataque DPA contra um circuito criptográfico em uma plataforma de prototipação com FPGA

Como ilustração, um ataque DPA contra o algoritmo DES implementado em FPGA é realizado. O Data Encryption Standard (DES) [2] é composto por 16 etapas (rounds) de substituições, com uma permutação inicial e uma permutação final. Em cada um dos 16 rounds, o algoritmo de criptografia DES realiza oito operações de substituição através de caixas de substituição (Substitution Boxes – S-Box). As 8 S-Boxes têm como entrada a saída de um XOR entre seis bits da chave criptográfica e seis bits do registrador R, e cada S-Box produz 4 bits de saída.

O ataque consiste em supor as sub-chaves na entrada de uma S-Box e predizer a saída correspondente. Por exemplo, pode-se predizer os seis bits de entrada da sub-chave da S-Box 1 como sendo "100101". Se correta, essa hipótese de chave permitirá ao atacante calcular os 4 bits de saída da S-Box 1, que são usados como parte da entrada o *round* seguinte de um cálculo do DES.

Para qualquer um dos quatro bits preditos traços de consumo são divididos em dois subconjuntos: o primeiro corresponde às hipóteses onde o bit menos significativo (chamado bit *b*) vale 0 (zero); e o segundo corresponde ao bit *b* igual à 1 (um). Em seguida, uma curva média é calculada para cada subconjunto, onde a *n*-ésima amostra em cada curva média é a média de todos as *n* amostras em todas as curvas do subconjunto. Finalmente o atacante calcula a diferença entre as curvas médias.

Se a hipótese original é errada, o critério utilizado para criar os subconjuntos será aproximadamente aleatório. Qualquer subconjunto escolhido aleatoriamente em um conjunto de dados suficientemente grande terá a mesma média. Como resultado, o traço diferencial tenderá à zero, e o adversário terá então que repetir o processo com uma nova hipótese de chave.

Se a hipótese tiver sido correta, contudo, a escolha dos subconjuntos será relacionada com o cálculo real (com a chave secreta original). Quando o bit *b* da sub-chave predita corresponde à chave real, sua manipulação terá um efeito irrisório nas curvas de consumo analisadas (comparado com o consumo real), o que vai aparecer como um desvio estatístico de zero significante na curva diferencial.

A principal idéia deste método de ataque é que a predição de um único bit de uma sub-chave de uma S-Box permite ao atacante descobrir os 6 bits da sub-chave e então, descobrir todos os 56 bits da chave em questão.

3. Métodos para dificultar ataques DPA

Algumas abordagens para diminuir as informações colaterais através do consumo de potência têm sido propostas. Isto pode ser obtido através da inclusão de um certo grau de aletoriedade aos cálculos criptográficos. Porém a inclusão de cálculos aleatórios tende a uma média que é simples de ser removida com uma análise estatística, com um número suficiente de amostras.

No nível arquitetural, uma solução poderia ser adicionar um módulo paralelo que atuariam como um espelho para as reais operações de criptografia, mas tratando dados complementares ou aleatórios. Por exemplo, se o módulo principal está calculando com uma sub-chave "010101", o módulo complementar estaria calculando com uma chave "101010" [3]. Isto poderia esmaecer a assinatura de consumo, tornando-a praticamente constante, ou praticamente aleatória. Contudo não é evidente que esta solução funcione na prática devido a aspectos de síntese física, podendo não mascarar corretamente o consumo real.

Entretanto, algumas contramedidas eficientes foram propostas, tanto em nível algorítmico quanto em nível material (*hardware*).

3.1. Contramedidas materiais

Os métodos materiais para dificultar o ataque DPA diferem expressivamente dos métodos algorítmicos. Para a abordagem material os resultados intermediários do cálculo criptográfico não são afetados. Como uma alternativa, a contribuição desta abordagem é esconder a parte atacável do consumo de potência com diferentes tipos de ruídos.

A adição de ruído tem uma relação direta com as necessidades de medidas (amostras) para o ataque. Isto não impede o ataque DPA, mas o torna sensivelmente mais difícil. Essas medidas são ainda mais eficientes quando combinadas com contramedidas algorítmicas [4].

No sentido de diminuir a correlação entre os dados de entrada e o consumo de potência de um determinado circuito, o número de amostras necessárias à realização de um ataque DPA deve ser aumentado. Em geral duas linhas maiores de contramedidas materiais têm sido propostas.

A primeira concerne a diminuição da Relação

Sinal Sobre Ruído (*Signal to Noise Ratio* -SNR), o que acarreta na diminuição da correlação entre o consumo de corrente hipotético e o consumo real do circuito. Para reduzir o SNR existem alguns trabalhos no estado-da-arte que sugerem a utilização de uma lógica especial para diminuir a dependência do consumo de corrente dos dados calculados.

Em [5], [6] e [7] a lógica balanceada (lógica dupla-porta) é proposta. A idéia básica é que uma porta lógica deve consumir uma potência equivalente independentemente dos valores em suas entradas. O SNR é reduzido através desta comutação independente de dados. Infelizmente, as experiências mostram que este objetivo é apenas parcialmente alcançada. A abordagem dupla-porta não é suficiente para garantir uma assinatura de consumo completamente independente de dados. Um problema potencial é que a carga de uma porta pode variar em função de diferenças no roteamento. O projeto de cada porta dupla-porta deve garantir uma carga de entrada nos pinos e uso balanceado de potência. Para conseguir isto, o processo de agrupamento de células durante o posicionamento (*placement*) deve ser feito de maneira cuidadosa, o que implica em um alto esforço de desenvolvimento. Apesar disto, o circuito final com lógica dupla-porta pode acarretar em um acréscimo de até 15 vezes no tamanho do circuito (comparado com o circuito em lógica normal), e em um aumento de até 6 vezes no consumo.

A segunda abordagem material para evitar ataques DPA trata da redução da correlação entre dados de entrada e potência consumida através de um desarranjo aleatório no tempo de cálculo das operações criptográficas. Se um tempo t_c é diferente em cada curva de consumo, a correlação entre o consumo de potência hipotético e o consumo real é grandemente reduzida. A contramedida proposta em [8] reside na inserção de atrasos aleatórios. A contramedida proposta em [4] dificulta o ataque DPA através do uso de blocos com consumo gerenciado afim de mascarar o consumo real. Os trabalhos [9] e [10] também dificultam o ataque DPA, mas como mostrado em [11], mesmo se um cálculo direto da probabilidade máxima de um dado consumo de potência em um momento determinado não é factível, ainda é possível realizar uma aproximação empírica baseada num modelo em software da contramedida.

Outra forma de diminuir o SNR é apresentada em [12], e consiste no mascaramento do consumo de potência não mais tornando aleatório o consumo ou criando ruídos, mas sim gerando, no nível de transistores, um consumo constante. Esta abordagem é similar ao trabalho proposto em [13], mas o circuito descrito em [13] considera somente o fato de o atacante monitorar o V_{cc}, deixando o fio do G_{nd} vulnerável, enquanto em [12] a solução proposta provê um mascaramento em ambos fios.

3.2. Contramedidas algorítmicas

Existem diversas contramedidas algorítmicas que dificultam os ataques DPA. Algumas das primeiras foram propostas em [14], e essas contramedidas foram eficientes contra ataques DPA clássicos. Para proteger o sistema criptográfico RSA [15] o primeiro método descrito por Coron [14] é aplicável, e o segundo método é uma adaptação da "assinatura cega" proposta por Chaum [16]. O terceiro método é interessante apenas para o algoritmo de criptografia baseado em curvas elípticas (*Elliptic Curve Cryptography* – ECC). Contudo, o ataque recentemente proposto chamado Análise Refinada de Potência (*Refined Power Analysis* – RPA) [17] torna essas contramedidas ineficazes.

O mascaramento de mensagem proposto por Paul Kocher [19] parece funcionar contra o ataque MRED [20] (um ataque objetivando implementações do RSA usando o teorema do resto chinês – CRT).

Em geral as contramedidas que protegem o RSA contra o ataque DPA baseiam-se no mascaramento da mensagem a ser criptografada ou do expoente (chave criptográfica do RSA). Esses métodos podem contribuir, ou não, com a segurança do sistema, dependendo da forma em que são implementados e do tipo de ataque realizado. Não é raro que uma contramedida que defende um tipo de ataque acabe por beneficiar outro tipo de ataque.

Este trabalho não detalha nem compara aqui as contramedidas algorítmicas, pois nosso método concerne somente implementações de algoritmos de criptografia que utilizam operadores modulares baseados no teorema do chinês do resto (CRT).

Diferentemente dos trabalhos que geralmente propõem o uso do CRT para acelerar o RSA (como o artigo [23]), outra pesquisa propõe uma completa modificação na representação numérica para computar o RSA, utilizando o RNS (*Residue Number System* – representação baseada no CRT) para calcular o RSA [24], [25]. Considerando que a melhor forma de proteção contra ataques DPA trata-se da eliminação da correlação entre dados calculados e consumo de potência, nossa abordagem combina métodos algorítmicos com implementação em hardware para ter sucesso nesta empreitada.

Além da acelerar o cálculo da exponenciação modular, uma completa implementação do RSA em RNS usada de forma inteligente pode impedir ataques DPA, conforme mostrado na Seção 5.

4. Descrição da arquitetura

Nós concebemos uma arquitetura reconfigurável de grão grande, chamada *Leak Resistant Reconfigurable Architecture* (LR²A) capaz de

realizar exponenciais modulares (operação básica do RSA) de forma segura contra ataques DPA. A LR²A executa uma aritmética baseada no teorema do resto chinês descrita em [24]. Além de ser uma solução robusta do ponto de vista da segurança, a LR²A também é uma solução flexível para implementar algoritmos de criptografia; além disso, ela pode ser vista como uma arquitetura capaz de realizar outros tipos de aplicações criptográficas baseadas na aritmética modular, como o ElGammal, o ECC e o RC6.

A LR2²A pode ainda ser facilmente modificada para executar aplicações suplementares, tais como compressão de dados ou tratamento de imagens, pois seus elementos reconfiguráveis de cálculo são semelhantes à microprocessadores, fornecendo grande flexibilidade.

A LR²A é construída em torno de três estruturas principais: um controlador de configurações e de injeção de dados, elementos de cálculo homogêneos (PE – *processing elements*), e uma hierarquia de memória.

O controlador é encarregado do monitoramento da arquitetura, interferindo quando necessário para enviar novos dados ou novas configurações aos PEs. Existem *k* PEs, onde *k* é dado pelo número de bases RNS necessárias ao cálculo do RSA (Ver Seção 5). O esquema de memória da LR²A é o acesso não-uniforme (NUMA), ou seja, a memória é distribuída e pode ser vista como uma memória local à cada PE, mas acessível para todos PEs e para o controlador.

As Subseções seguintes descrevem o controlador, o PE, a estrutura de memória e o modelo de configurações da LR²A.

4.1. O controlador

A principal diferença entre a LR²A e a maioria das arquiteturas reconfiguráveis é que a LR²A não é somente um núcleo de laço para um algoritmo específico. A arquitetura proposta é composta por uma estrutura de controle que a torna mais flexível. Um processador controla o processo de reconfiguração e a injeção e recuperação de dados nas memórias distribuídas. A Figura 2 mostra uma vista geral da LR²A com esse controlador.

O microprocessador Plasma foi escolhido para realizar o procedimento de controle. Esta escolha deu-se pelo fato do Plasma ser distribuído livremente, mas principalmente porque possui um compilador C associado, o que facilita a programação do sistema de controle. Este microprocessador é baseado no conjunto de instruções do MIPS R3000, o que tem a vantagem de permitir a compatibilidade com outros processadores utilizados em sistemas embarcados.

O processo de configuração controlado pelo Plasma dá-se da seguinte forma:

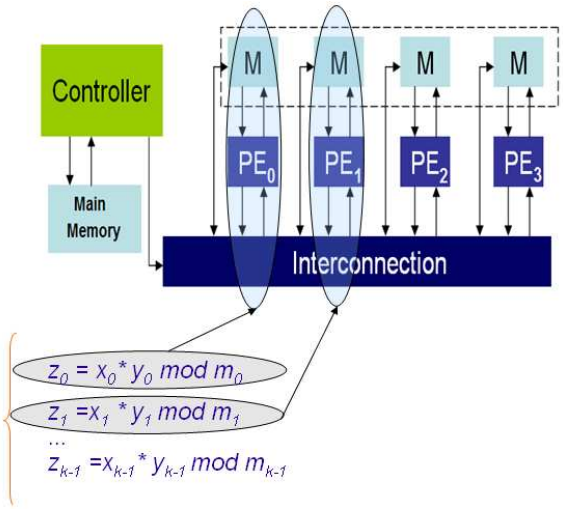

Figura 1 – Uma visão da LR²com algumas das operações realizadas pelos PEs

- No início, o controlador leva uma configuração inicial à cada PE.
- O estado das configurações é armazenado em uma estrutura de dados associada à cada nodo.
- Em seguida, os dados necessários aos cálculos de cada PE são encaminhados à memória local alocada à cada um deles.
- Uma vez que configurações e dados estão em seus devidos lugares, a computação inicia.
- Finalmente, o controlador recupera, no momento adequado, os dados e os registra na memória principal.

Para as mudanças seguintes (por exemplo, chegada de novos dados, ou necessidade de novas configurações), o controlador salva ou descarta o estado de cada PE e de sua memória, dependendo do caso, antes de enviar a nova informação. Em verdade o processo de controle consiste em rodar o sistema RSCM, descrito na subseção 4.3.

4.2. Os elementos de cálculo

Os elementos de cálculo da LR²A têm uma arquitetura RISC, *load-store*: as instruções lógicas e aritméticas são executadas somente através de registradores internos, enquanto as instruções que acessam a memória executam a leitura (*load*) ou a escrita (*store*) de uma posição na memória.

Devido à essa opção de arquitetura *load-store*, o processador deve ter um conjunto relativamente grande de registradores de propósito geral para manipulação de dados, afim de reduzir o número de acessos à memória (acessos estes que sempre representam uma penalidade em termos de tempo em relação às operações internas ao processador). No que concerne o formato das instruções, todas elas

têm o mesmo tamanho, ocupando uma palavra de memória cada. A instrução contém o código da operação e a especificação dos operandos, caso existam.

Outras características presentes na LR²A são comuns aos processadores RISC:

1. Endereços e caminho de dados de 32 bits.
2. Endereçamento de memória baseado em palavra.
3. O banco de registradores contém 16 registradores de propósito geral, todos com 32 bits.
4. Existem quatro sinalizadores de estado (*flags*): Negativo, Zero, Vai-um e estouro (*overflow*).

O caminho de dados inclui operadores específicos à criptografia de chave pública, essencialmente relacionados com operações modulares. Os seguintes operadores são incorporados à ULA:

- Redução Modular (X mod M)
- Redução Modular por um fator 2^k (X mod 2^k)
- Adição Modular ($X+Y$ mod M)
- Subtração Modular ($X-Y$ mod M)
- Multiplicação Modular ($X \times Y$ mod M)
- Multiplicação Modular por um fator 2^k ($X \times Y$ mod 2^k)

Vale ressaltar que deslocamentos e rotações são realizadas através de um deslocador de Barrel, o que consome mais área, mas permite que essas operações sejam realizadas em apenas um ciclo.

4.3. O controle de configurações

Freqüentemente a eficiência de uma arquitetura reconfigurável é comprometida pela ausência de métodos de controle do processo de reconfiguração e de injeção de dados.

Para preencher esta lacuna, um modelo de controle de configurações associado à LR²A é proposto. Este modelo é composto por diferentes módulos: memória de configuração (CM), monitor de reconfiguração (RM), expedidor de reconfiguração (RD), agendador de configuração (SC), e o controle central de configuração (CCC).

Este modelo de controle de reconfiguração é baseado no *Reconfigurable System Configuration Manager* (RSCM) idéia apresenteda em [26]. Uma visão geral do novo RSCM é mostrada na Figura 2.

Como o nome indica, a memória de configuração (CM) é um espaço reservado na memória do controlador que armazena todas as configurações que podem ser necessárias à uma dada aplicação.

O Monitor de Reconfiguração (RM) detecta situações onde reconfigurações são necessárias, chamadas de eventos de reconfiguração, e notifica o CCC de tais eventos, para que este último aja de forma apropriada.

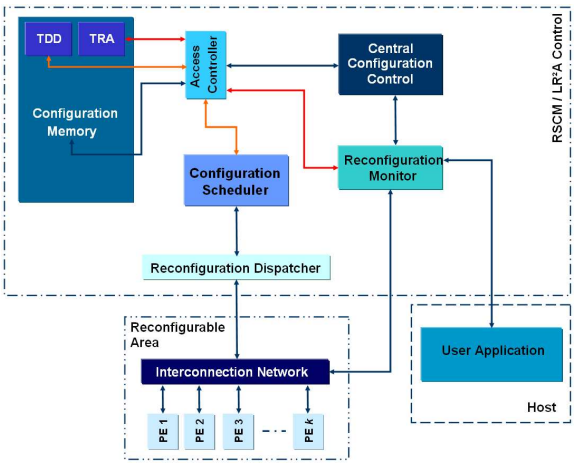

Figura 2 – O Controle da LR²A baseado no modelo RSCM

Todas as informações sobre a alocação dos PEs e suas tarefas correspondentes são armazenadas na Tabela de Alocação de Recursos (TRA), associada ao CCC. O CCC recebe requisições do RM e expede os serviços necessários ao CS e ao RD.

O módulo Agendador de Configuração (CS) é responsável pela determinação de qual configuração será a próxima a ser realizada. O CS recebe requisições de serviço do CCC, e armazena uma estrutura de dados com informações sobre a dependência de configurações, chamada Tabela de Dependências e Descritores (TDD).

Finalmente, o módulo RD é encarregado da reconfiguração em si, enviando o código apropropriado para cada PE, ou para um grupo de PEs (o *broadcast* de configurações é possível).

Como mencionado em [26], o RSCM pode ser implementado em hardware, em software ou numa combinação dos dois. No caso da LR²A foi optado pela implantação do RSCM em software, rodando no controlador Plasma.

Este modelo de controle de configurações permite a centralização de todo controle e sincronização, evitando assim problemas de coerência de cache. O controlador "sabe" o que está acontecendo em cada PE porque cada PE tem uma estrutura de dados associada a si, e atualizada pelo RSCM.

5. Uma aritmética resistente à fugas

A Leak Resistant Arithmetic (doravante chamada LRA), é baseada na representação numérica RNS e na multiplicação modular de Montgomery em RNS proposta em [22]. Esta aritmética é o fundamento matemático do qual deriva a arquitetura proposta.

5.1. *Residue Number System* (RNS)

O RNS é um sistema de numeração não-posicional baseado no Teorema Chinês do Resto. Este teorema indica que é possível representar

grandes inteiros utilizando um pequeno conjunto de pequenos inteiros, de forma que os cálculos possam ser realizados de uma maneira mais eficiente.

O RNS é definido, pois, por um conjunto de k inteiros $\{m_1, m_2, m_3 .. m_k\}$, chamados de módulos. Os módulos devem todos ser co-primos, ou seja um módulo não pode ser fator de nenhum outro módulo. Seja, então, M o produto de todos m_i . Neste caso qualquer inteiro X (desde que menor que M) pode ser representado no sistema numérico residual definido como um conjunto de k pequenos inteiros $\{x_1 , x_2, x_3 .. x_k\}$, com $x_i = X$ modulo m_i representando a classe de resíduos de X relativo aos modulos escolhidos para a base.

Usando este sistema, operações aritméticas, como a adição e a multiplicação por exemplo, podem ser realizadas em paralelo, pois operações RNS não possuem propagação de vai-um. Por exemplo, para calcular a adição entre A e B em RNS com uma base composta por mi ($0 < m \leq k$) é: Addi = (ai + bi) modulo mi.

A conversão de RNS para decimal não é assim tão trivial, mas somente é necessária após a realização dos cálculos, portanto seu custo é amortizado. Para esta conversão a seguinte formula é necessária:

$$x = \sum_{i=1}^{k} x_i M_i \left| M_i^{-1} \right|_{m_i} \bmod M \qquad (1)$$

Para aplicações criptográficas de chave pública, redução, adição e exponenciação modulares são as operações mais importantes. Elas podem ser calculadas com o algoritmo de Montgomery, modificado para trabalhar com o RNS, conforme descrito na próxima subseção.

5.2. Algoritmo de multiplicação modular de Montgomery

A versão do algoritmo de Montgomery apresentada abaixo é a proposta em [24]. Na representação RNS o valor M é obtido por:

$$M = \prod_{i=1}^{k} m_i \qquad (2)$$

Então o número inteiro M é escolhido como a constante de Montgomery, ao invés de um número no formato 2^k de sua representação clássica.

O Algoritmo 1 descreve a multiplicação modular de Montgomery em RNS. Como entradas tem-se duas bases RNS β_1 e β_2, de forma que M e M' possam ser calculados como o produto dos módulos que compoem β_1 e β_2. As entradas A, B e N são também representadas em ambas bases β_1 e β_2. Além disso, um módulo redundante m_r tal que $mdc^*(m_r, m_i)=1$ é necessário. Os inteiros N e M devem também ser co-primos. O resultado final é obtido em R, na base β_1. O resultado R corresponde, portanto, à: $R = A.B.M_1^{-1} \bmod N$ na base RNS $\beta_1=\{m_1, m_2, m_3 .. m_k\}$.

$1: T \leftarrow A \otimes_{RNS} B \ \ in \ \ \beta_1 \ \ and \ \ \beta_2$
$2: Q \leftarrow T \otimes_{RNS} (-N^{-1}) \ \ in \ \ \beta_1$
$3: Extend \ \ Q \ \ from \ \ \beta_1 \ \ to \ \ \beta_2$
$4: R \leftarrow (T \oplus_{RNS} Q \otimes_{RNS} N) \otimes_{RNS} M^{-1} \ in \ \beta_2$
$5: Extend \ \ R \ \ from \ \ \beta_2 \ \ to \ \ \beta_1$

Algoritmo 1 – Multiplicação Modular em RNS com extensão de base (núcleo da LRA)

Contudo, o valor M_1^{-1} não existe em β_1. Então outra base, chamada de β_2 é definida como uma extensão de β_1, com outros módulos co-primos entre eles e também em relação à β_1. Como conseqüencia, antes de se calcular M_1^{-1} uma extensão de base, de β_1 à β_2 é realizada (ver algoritmo 1, linhas 3 e 5).

As linhas 1,2 e 4 do mesmo algoritmo consistem de operações RNS completamente realizadas em paralelo. Portanto a real complexidade do algoritmo está na realização das extensões de base (linhas 3 e 5). Existem alguns métodos para realizar essa tarefa, dos quais o Mixed Radix System, descrito em [21] parece ser o mais interessante, pois necessita que apenas k valores sejam estocados para cada modificação de base [22].

5.3. Aspectos de segurança da LRA

Apesar do desempenho intrínseco ao grau de paralelismo obtido, o interesse da LRA reside no fato de que a implementação da multiplicação modular em RNS também provê a possibilidade de tornar aleatórias as bases: a segurança deste método reside nesta possibilidade.

O maior objetivo da LRA é tornar aleatórios os dados intermediários calculados pelo circuito durante uma operação criptográfica, para as mesmas entradas e saídas. Baseado no mesmo princípio do ataque DPA, se um dado muda durante uma operação, consequentemente o consumo de potência também sofre modificações, deixando de ser constante. Desta forma torna-se difícil a realização de um ataque por análise estatística de consumo.

A LRA propõe duas abordagens para gerar essa aleatoriedade dos dados intermediários: uma no nível do circuito (aleatoriedade espacial) e outra no nível dos dados (mascaramento aritmético). Elas representam um bom compromisso entre segurança e custos de implementação. As abordagens consideradas são:

- **Escolha aleatória das bases iniciais**: A Aleatoriedade dos dados de entrada é conseguida pela escolha aleatória dos elementos dos módulos que compoem β_1 e β_2 antes de cada exponenciação modular.

* Maior divisor comum

- **Mudança aleatória de bases antes e durante uma exponenciação:** Um algoritmo genérico é proposto em [22], propondo vários graus de aleatoriedade para a implementação, dependendo do nível de segurança desejado.

A segurança do método *Leak Resistant Arithmetic* é descrita na referência [22].

6. Resultados

6.1. Área e desempenho

A Tabela I mostra o tamanho de cada PE para diferentes larguras de caminhos de dados. O tamanho é dado por milhares de portas lógicas equivalentes (ple).

TABELA I
Resultados de síntese para PEs com diferentes tamanhos de caminhos de dados

Caminho (bits)	ple (k)	Clock (MHz)
16	4,3	60
32	8,5	40
64	15,8	30
128	32	20

Considerando a configuração com caminho de dados de 32bits, e levando em conta o tamanho do controlador que é de 40k portas, uma LR²A composta por 32 PEs (necessários para realizar exponenciações modulares de 1024 bits), tem uma área total de aproximadamente 352k portas. Comparando com aceleradores em hardware do estado-da-arte com o mesmo propósito, a LR²A consome de 5 a 8 vezes o tamanho desses dispositivos.

Contudo, em termos de desempenho, como mostrado na Figura 4, para chaves criptográficas maiores que 1024 bits, a LR²A possui tempos de resposta mais interessantes. A comparação foi realizada usando o popular método *square and multiply* para a exponenciação modular. A implementação clássica mencionada na Figura 4 diz respeito à uma exponenciação modular calculada com um circuito específico que implementa o algoritmo de Montgomery para multiplicação modular, com um caminho de dados de 32 bits e rodando em uma freqüência de 40Mhz. Do outro lado tem-se, a LR²A implementada com caminho de dados também de 32 bits, com 32 processadores rodando à mesma freqüência de 40MHz.

Todavia, comparando o desempenho da LR²A com outras arquiteturas do estado-da-arte que calculam a exponenciação modular, a Figura 5 mostra claramente que os incrementos em segurança obtidos pela LR²A não comprometem a performance. A LR²A é capaz de calcular uma exponenciação modular de 1024 bits no mesmo tempo que a mais rápida arquitetura que encontramos no estado-da-arte.

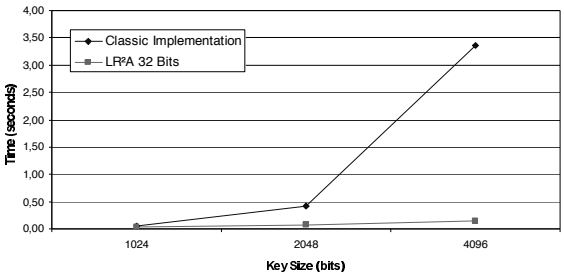

Figura 3 – Comparação de Tempo x Tamanho de chave entre uma implementação clássica do algoritmo de Montgomery e a LR²A, ambos realizando exponenciações modulares

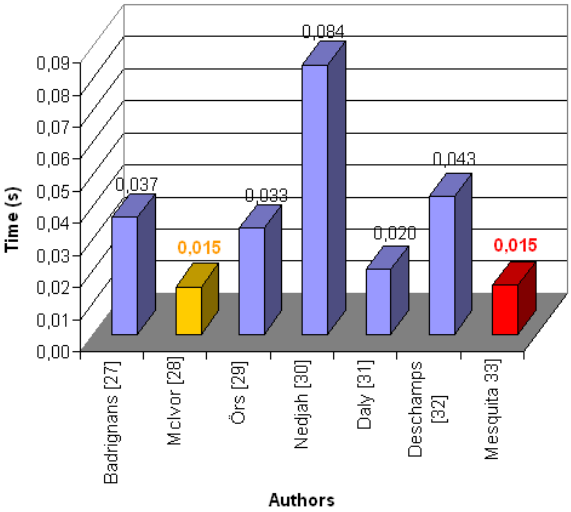

Figura 4 – Comparação de desempenho entre a LR²A e outras implementações de exponenciação modular do estado-da-arte.

Em relação às outras arquiteturas do estado-da-arte, a LR²A consome 5 vezes mais área para chaves de 2048 bits e 10 vezes mais para 4096 bits. Este é o custo do compromisso entre segurança e desempenho. Nenhuma das outras implementações revisadas continha medidas de prevenção contra ataques por análise de consumo, enquanto a LR²A possui a robustez necessária para resistir contra ataques DPA.

6.2. Robustez

Em relação aos aspectos de segurança, através da modificação dos dados intermediários do cálculo de uma exponenciação, via a extensão de base usada no método LRA, a arquitetura proposta consome potência diferentemente para os mesmos dados de entrada. Isto significa que é calculada uma exponenciação modular (A^e mod N) com diferentes bases RNS.

Na Figura 6 pode-se ver a ilustração dessa experiência. Num primeiro momento calcula-se uma

exponenciação modular clássica por duas ocasiões, com os mesmos valores de entrada, e armazena-se as respectivas curvas de consumo de potência. Se então calcularmos a diferença entre essas curvas de consumo obteremos um traço praticamente reto, o que significa que para os mesmos dados de entrada, na implementação clássica, teremos sempre o mesmo consumo (Figura 6-(a)). Isto confirma o princípio mesmo do ataque DPA.

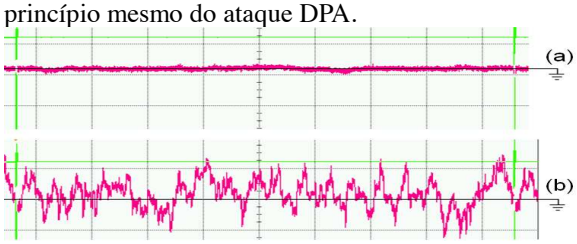

Figura 5 - Análise da robustez da LR2A

Entretanto, utilizando a mudança de bases RNS implementada na LR²A, para os mesmos dados de entrada *A, e* e *N*, temos consumos diferentes se convertermos esses valores para RNS com bases β_1 e β_2 diferentes. A figura 6-(b) ilustra este fato, em que há um consumo diferente de uma base para outra. Nesta experiência, para fins pedagógicos, foram utilizadas apenas duas bases diferentes, utilizando-se a técnica de escolha aleatória das bases iniciais, descrita na seção 5.3. Se incluirmos ainda mudanças aleatórias de base durante a exponenciação, o sucesso de um ataque DPA torna-se ainda menos provável.

6.2. Reconfigurabilidade

O principal aspecto de segurança da LR²A reside na mudança de bases RNS. Isto é realizado através de uma reconfiguração dinâmica do dispositivo.

Como pode ser visto no Algoritmo 1, em um primeiro momento a arquitetura realiza operações RNS comuns (linhas 1 e 2). Contudo, quando uma extensão de base se faz necessária (linha), devido à sua natureza algoritmica, a arquitetura deve ser reconfigurada para executar as novas instruções. Então o RSCM intervém para gerenciar o procedimento de reconfiguração.

O Monitor de Reconfiguração determina o momento em que o módulo de extensão de base é necessário e notifica o Controle Central de Configurações. O CCC chama então o Agendador de Configurações, que verifica se o módulo de extensão de base está presente na Memória de Reconfiguração. Enquanto isto, os dados necessários são armazenados nas memórias locais.

Em seguida o Expedidor de Reconfigurações leva a nova configuração aos PEs envolvidos com a extensão de base.

Depois que a extensão de base termina, as operações RNS devem então ser reativadas (linha 4 do Algoritmo 1). Isto é obtido com uma nova atuação do RSCM, repetindo os passos indicados no parágrafo precedente.

Agindo desta forma, o sistema pode ser considerado como sendo dinamicamente reconfigurável, e o contexto nunca é perdido devido ao fato que os traços de todas operações de reconfiguração da arquitetura são guardados na Tabela de Dependências e Descritores do RSCM.

7. Conclusões

A LR²A incrementa a robustez de aplicações criptográficas, evitando que ocorra o princípio que permite ataques por análise de consumo contra circuitos, como a DPA. Devido à sua natureza, esse método aritmético leva à definição de uma arquitetura reconfigurável de grão grande, necessitando de grande quantidade de recursos de hardware. Mesmo tendo um certo custo em hardware, a arquitetura proposta LR²A é competitiva em termos de desempenho, sendo equivalente a outros aceleradores de exponenciação modular encontrados no estado-da-arte. Esse acréscimo em área é um a penalidade razoável, sendo que o incremento em segurança implica freqüentemente em algum tipo de custo. Todavia a LR²A mostra que é possível aumentar a segurança e permanecer competitiva em termos de desempenho.

Este trabalho ressalta também que a técnica de reconfiguração dinâmica possui aplicações práticas também para aplicações criptográficas, evidenciando o acréscimo em segurança face aos ataques por análise de consumo de corrente.

Como perspectivas pode-se salientar que com leves modificações nos PEs, a arquitetura LR²A pode rodar outras aplicações além da criptografica, como a compressão de dados ou tratamento de imagens. Porém o mais interessante trabalho futuro é uma modificação mais profunda nos PEs, através da adição em cada um deles de um caminho de dados reconfigurável, o que permitirá a implementação eficiente de algoritmos de criptografia de chave secreta, assim como virtualmente um largo espectro de outras aplicações.

References

[1] P. Kocher, al. “Differential Power Analysis : Leaking Secrets”. *Advances in Cryptology: CRYPTO'99*, pp. 388-397. 1999.

[2] – . “ Data Encryption Standard (DES)”. Federal Information Processing Standards Publications (FIPS PUBS) Nº 46-3. EUA.October 25, 1999.

[3] C. Walter. “Sliding Windows Succumbs to Big Mac Attack”. *Cryptographic Hardware and Embedded System: CHES'01,* pp286-299. 2001.

[4] L. Benini, et al. “Energy-aware design techniques for differential power analysis protection”. *Design*

Automation Conference: DAC '03. Anaheim, USA. June, 2003.

[5] A. Razafindraibe, et al. "Asynchronous Dual rail Cells to Secure Cryptosystem Against SCA". *Sophia-Antipolis Forum on MicroElectronics.* Nice, France, 2005.

[6] H. Saputra, et al. "Masking behavior of DES encryption". *Design, Automation and Test Europe – DATE '03*. Munich, Germany, 2003.

[7] M. Simon, et al. "Balanced Self-Checking Asynchronous Logic for Smart Card Applications", *Microprocessors and Microsystems Journal*, 27. Elsevier, pp 421-430, October 2003.

[8] C. Clavier, et al. "Differential Power Analysis in the presence of hardware countermeasures". *Cryptographic Hardware and Embedded Systems – CHES '00*. Pp 252-263, 2000.

[9] J. Irwin, et al. "Instruction stream mutation for non-deterministic processors". *International Conference on Application Specific Systems, Architectures and Processors – ASAP 2002*. IEEE press. Pp 286-295. 2002.

[10] D. May, et al. "Non-deterministic processors". *Information security and privacy – ACISP'01*. Sydney, Australia. July 2001.

[11] S. Mangard, "Hardware countermeasures against DPA – a statistical analysis of their effectiveness". *Topics in Cryptology – CT-RSA'04*. pp. 222 – 235. San Francisco, USA. 2004.

[12] D. Mesquita, et al. "Current Mask Generation: A New Hardware Countermeasure for Masking Signatures of Cryptographic Cores". *International Conference on VLSI: IFIP VLSI SoC '05*. Perth, Australia, 2005.

[13] A. Shamir. "Protecting smart cards from passive power analysis with detached power supplies". *Cryptographic Hardware and Embedded Systems, CHES'00.*Pp 71-77, 2000.

[14] J-S Coron. "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems". *Cryptographic Hardware and Embedded Systems, CHES'99*. Pp 292-302, 1999.

[15] R. Rivest, et al. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *ACM Communications*, vol 21. pp 120-126. 1978.

[16] D. Chaum. "Security without identification: transaction systems to make Big Brother obsolete". *Communication of the ACM*. Vol. 8., n° 10, pp 1030-144. 1985.

[17] L. Goubin. "A refined power-analysis attack on ECC". *Publick Key Cryptography: PKC '03*. pp 199-210. 2003.

[18] M. Hideyo, et al. "Efficient Countermeasures against RPA, DPA, and SPA". *Cryptographic Hardware and Embedded Systems, CHES'04*. Pp 343-356, 2004.

[19] P. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". *16th Workshop in Cryptology: Crypto '96.* pp 104-113. Santa Barbara, USA. 1996.

[20] B. Boer. "A DPA Attack against the Modular Reduction within a CRT Implementation of RSA**".** *Cryptographic Hardware and Embedded Systems, CHES'02*. pp 228-243, 2002.

[21] H. Garner. "The Residue Number System". *IRE Transactions in electronic Computers*. Vol 8, pp. 140-147, 1959.

[22] J-C. Bajard, et al. "Leak Resistant Arithmetic". *Cryptographic Hardware and Embedded Systems CHES'04*. Pp 62-75, 2004.

[23] C. Kim, et al. "A CRT-Based RSA Countermeasure against Physical Cryptanalysis". *Conference on High Performance Computing and Communications: HPCC '05*. Pp 549-554, Naples, Italy, 2005.

[24] J-C. Bajard, et al. "A Full RNS Implementation of RSA".*IEEE Transactions on Computers.* Vol. 53, n° 6, pp. 769-774. 2004.

[25] M. Ciet, et al. "Parallel FPGA implementation of RSA with residue number systems – can side-channel threats be avoided?". *46th. International Midwest Symposium on Circuits and Systems: MWSCAS '03*. Cairo, Egypt, December 2003.

[26] E. Carvalho, et al. "Reconfiguration Control for Dynamically Reconfigurable Systems". *Conference on Design of Circuits and Integrated Systems*: *DCIS '04*. Bordeaux, France, 2004.

[27] B. Badrignans, D. Mesquita, J-C. Bajard, L. Torres, G. Sassatelli, and M. Robert. A parallel and secure architecture for asymetric cryptography. In Proceedings of ReCoSoC, page in press, Montpellier, France, 2006.

[28] C. McIvor, M. McLoone, J. McCanny, and W. Marnane. Fast montgomery modular multiplication and RSA cryptographic processor architectures. In Proceedings of the Asilomar Conference, Pacific Groove, USA, 2003. IEEE Computer Society.

[29] S. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of a montgomery modular multiplier in a systolic array. In IPDPS, page 184, Nice, France, 2003. IEEE Computer Society.

[30] N. Nedjah and L. Mourelle. A review of modular multiplication methods and respective hardware implementations. Informatica, 30 :111–130, 2006.

[31] A. Daly and W. Marnane. Efficient architectures for implementing montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic. In Proceedings of the FPGA '02, pages 40–49, 2002.

[32] J-P. Deschamps and G. Sutter. Fpga implementation of modular multipliers. In Proceedings of the DCIS '02, pages 107–112, 2002.

[33] D. Mesquita, B. Badrignans, L. Torres, G. Sassatelli, M. Robert, and F. Moraes. A leak resistant soc against side channel attacks. In Proceedings of ISSoC, page in press, Tampere, Finlande, 2006.