CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

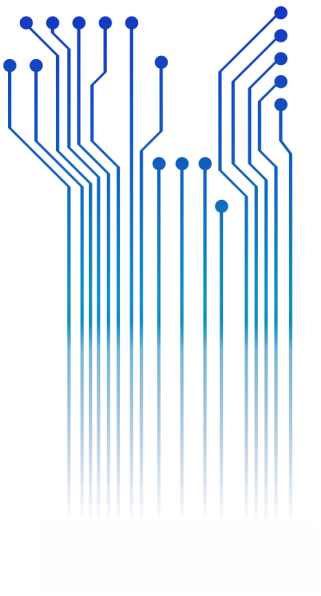
ROBERTO LUIZ PIMENTEL COSTA JUNIOR

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

PROJETO DE IP DE CRIPTOGRAFIA UTILIZANDO CURVAS ELÍPTICAS (ECC)

Departamento de
Engenharia Elétrica



Campina Grande
2018

ROBERTO LUIZ PIMENTEL COSTA JUNIOR

PROJETO DE IP DE CRIPTOGRAFIA UTILIZANDO CURVAS ELÍPTICAS (ECC)

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador:
Professor Marcos Ricardo Alcântara Morais, D.Sc.

Campina Grande
2018

ROBERTO LUIZ PIMENTEL COSTA JUNIOR

PROJETO DE IP DE CRIPTOGRAFIA UTILIZANDO CURVAS ELÍPTICAS (ECC)

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Microeletrônica

Aprovado em __/__/____

_____

Professor Marcos Ricardo Alcântara Morais, D.Sc.
Universidade Federal de Campina Grande
Orientador

_____

Professor Gutemberg Gonçalves dos Santos Júnior, D.Sc.
Universidade Federal de Campina Grande
Avaliador

Dedico este trabalho a todos que
me apoiaram ao longo do curso.

AGRADECIMENTOS

Agradeço a primeiramente aos meus pais, por terem se esforçado para me proporcionar uma boa educação, e terem me alimentado com o apoio durante a minha estadia em Campina Grande, essencial para superar todas as adversidades ao longo dessa caminhada.

Agradeço também a minha namorada, Juliana de la Flor, por todo o tempo dedicado e apoio fornecido desde o início do curso.

Aos meus orientadores Prof. Marcos Morais, Prof. Elmar Melcher e Prof. Gutemberg Gonçalves, pelas oportunidades e orientações concedidas ao longo dos últimos anos.

A todos os membros do projeto de Excelência em Microeletrônica do Nordeste, por todos os ensinamentos que tive ao longo do tempo em que participei do laboratório, em especial Felipe Assis, por estar sempre disponível para tirar dúvidas quando o *StackOverFlow* não foi capaz de saná-las.

Enfim, agradeço a todos que de alguma forma, passaram pela minha vida e contribuíram para a construção da pessoa que sou hoje.

“A persistência é o menor caminho do êxito.”

Charles Chaplin

RESUMO

A criptografia assimétrica é de grande importância para a comunicação de dispositivos atuais. Sua utilização permite além da troca de mensagens codificadas, a validação de documentos e compartilhamento de chaves utilizando um canal inseguro. Entretanto, devido ao crescente aumento do poder computacional nos últimos anos, tem-se tornado cada vez mais difícil manter a relação da chave pública com a chave privada em segredo dos atacantes, resultando em um aumento exponencial do tamanho da chave pública como método de prevenir-se desse avanço. A criptografia de curvas elípticas se insere nesse contexto como alternativa a estes algoritmos, fornecendo grande segurança de informação utilizando uma chave de tamanho significativamente menor.

Palavras-chave: Criptografia, curvas elípticas, segurança da informação.

ABSTRACT

Asymmetric encryption is in charge of great importance for the communication of nowadays devices. It is applicable in a variable of ways, including exchanging codded messages, document validations and key sharing, using a insecure channel. However, due to the increase computing power in recent years, it has become incredible difficult to keep the private and public key's relationship in secret from attackers, resulting in a exponencial increase in the size the public key as a previnition of this advance. Elliptic curve encryption is inserted in this context as an alternative to those algorithms, providing high security of information using a key of significantly smaller size.

Keywords: Crytography, elliptic curves, information security.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

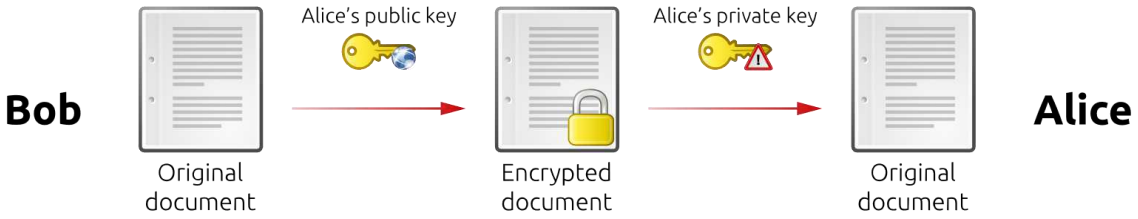
SUMÁRIO

1 INTRODUÇÃO

Com o avanço de tecnologias como a Internet das Coisas, os objetos de uma residência tendem a ficar cada vez mais conectados, gerando uma enorme quantidade de dados. A proteção destes dados, bom como a privacidade das pessoas que utilizam de tais objetos é um tema bastante explorado, o que leva ao estudo de diferentes formas de proteção, dentre as quais iremos ressaltar, os algoritmos de criptografia.

A criptografia assimétrica é um tipo de protocolo onde o usuário possui duas chaves, uma delas privada e a outra, pública, onde apesar de diferentes, é possível determinar uma ligação matemática entre ambas. A chave pública é utilizada para encriptação de texto e validação de assinaturas digitais, já a chave privada, faz justamente a operação oposta, ou seja, decodificação do texto e a assinatura digital.

Figura 1 – Demonstração de fluxograma de criptografia de chave pública.



Fonte: wikimedia.org

Embora pareça bastante conveniente a utilização de duas chaves distintas para as operações realizadas, a criptografia assimétrica tende a requerer mais tempo de processamento e maior memória do hardware utilizado, pois a ligação matemática entre as chaves pública e privada precisa ser de difícil determinação para que a codificação seja segura.

Esses sistemas realizam duas funções: autenticação, onde a chave pública verifica se foi o detentor da chave privada que enviou a mensagem, e a encriptação, onde com a chave privada é possível decodificar a mensagem enviada com a chave pública. Em um sistema de chave pública, qualquer usuário pode criptografar uma mensagem utilizando a chave pública do receptor.

Inicialmente, os protocolos de criptografia assimétrica propostos eram baseados na utilização de números primos grandes, onde o usuário escolhia aleatoriamente dois

números para serem sua chave privada, e a multiplicação destes seria a sua chave pública, de forma que a obtenção deste número através de força-bruta fosse uma atividade computacionalmente muito custosa, o que impossibilitaria a um atacante a determinação da relação entre a chave e a mensagem capturada.

Devido à complexidade matemática de execução desses algoritmos, normalmente são utilizados apenas para pequenas trocas de dados, normalmente, para o envio inicial de uma chave simétrica que será posteriormente utilizada.

No entanto, com o avanço do poder computacional observado ao longo dos últimos anos, houve um aumento considerável no número de bits necessários para armazenar os tais números primos grandes, de tal forma que atualmente, um projeto utilizando estes algoritmos deve utilizar 3072 bits para o armazenamento de cada um dos números primos. Ainda, a escolhe destes números primos grandes, não poderia ser de forma predeterminada, pois a utilização de números não aleatórios prejudica a proteção dos dados.

Tendo em vista tais limitações, vários protocolos de criptografia assimétrica são utilizados apenas como forma de conexão segura para determinar-se uma chave comum para a utilização de um canal de comunicação utilizando criptografia simétrica, onde, por não existir uma chave pública, a determinação da chave privada por força bruta torna-se muito mais complicada.

É nesse contexto onde existe um aumento exponencial do tamanho da chave pública, em que se encaixa a utilização da Criptografia de Curvas Elípticas, utilizando equações matemáticas definidas por curvas elípticas, esse tipo de sistema consegue fornecer alta capacidade de codificação utilizando chaves públicas bem menores do que as propostas pelos protocolos tradicionais, nesse caso representado pelo protocolo RSA (Rivest-Shamir-Adleman), um dos primeiros sistemas de criptografia de chave pública, conforme pode ser observado na Tabela 1.

Tabela 1 - Comparativo entre RSA e ECC

Segurança (bits)	**RSA**	**ECC**	**Protegido até**
80	1024	160-223	2010
112	2048	224-255	2030
128	3078	256-383	Mais do que 2031

Fonte: National Institute of Standards and Technology

1.1 OBJETIVOS

O objetivo geral deste trabalho é propor uma implementação de uma unidade de criptografia utilizando curvas elípticas. Os objetivos específicos são:

- Revisão bibliográfica da criptografia por curvas elípticas;
- Estudar os algoritmos de implementação no intuito de utilizar o que apresentar os melhores resultados;
- Desenvolver um software capaz de realizar o algoritmo definido;
- Definir uma arquitetura capaz de produzir os mesmos resultados apresentados pelo software desenvolvido
- Implementar em linguagem de descrição de hardware a arquitetura definida.

2 REVISÃO MATEMÁTICA

Para o melhor entendimento da matemática por trás da criptografia de curvas elípticas, é necessário primeiro, entender alguns conceitos mais básicos, de Álgebra Linear, de forma que a definir o conjunto onde todas essas operações irão ocorrer, de tal forma que seja possível validar o seu funcionamento não apenas computacionalmente, mas matematicamente.

2.1 GRUPOS ABELIANOS

Primeiramente, definimos o significado de um Grupo Abeliano, que também é chamado de grupo comutativo. É um grupo G, onde existe um operador binário $*$, em que $a * b = b * a$ para quaisquer a e b pertencentes à G. Em outras palavras, a aplicação da operação binária não depende da ordem dos elementos do grupo. Se n for um número natural e x for um elemento de um grupo abeliano G, escrito aditivamente, então nx pode ser definido como $x + x + \cdots + x$ e $(-n)x = -(nx)$. [1]

2.2 REPRESENTAÇÃO DE MONTGOMERY

Se a e b são inteiros definidos no intervalo $[0, N - 1]$, a soma deles é definida no intervalo $[0,2N - 2]$ e a diferença $[- N + 1, N - 1]$, então a representação dessas operações no intervalo definido por $[0, N - 1]$ requer no máximo uma subtração ou adição, respectivamente, por N. Entretanto, o produto ab é definido para o intervalo $[0, N\text{^}2 - 2N + 1]$. Dessa forma, armazenar o produto intermediário ab requer o dobro de bits que os inteiros a e b, e a determinação da representação do mesmo no intervalo desejado $[0, N - 1]$ requer uma divisão.

Dados dois inteiros a e b e modulo N, a multiplicação modular calcula inicialmente o produto de precisão dupla $ab\ mod\ N$, e então, calcula a divisão, subtraindo múltiplos de N de forma a cancelar os bits mais significativos indesejados até que o produto resultante seja menor que N.

A redução de Montgomery, por outro lado, adiciona múltiplos de N para cancelar os bits menos significativos até que o resultado seja um múltiplo de uma constante $R > N$. Então, os bits menos significativos são descartados, produzindo um resultado menor que $2N$. Uma última subtração pode acontecer, de forma que o resultado seja menor que N. Esse processo evita a complexidade dos algoritmos clássicos de divisão. O resultado é o produto desejado dividido por R.

Para multiplicar a e b, estes números são inicialmente convertidos para uma representação de Montgomery $aR \ mod \ N$ e $bR \ mod \ N$. Quando multiplicados, esse produto $abR^2 \ mod \ N$, e a divisão final produz $abR \ mod \ N$, a forma de Montgomery do produto desejado.

Matematicamente, o inteiro entre 0 e $N - 1$ que é congruente à ab pode ser expressado utilizando o teorema de divisão Euclidiana:

$$ab = qN + r \tag{1}$$

Onde q é o quociente ab/N e r, o resto, no intervalo definido $[0, N - 1]$. O resto r é então, por definição $ab \ mod \ N$. A determinação de r ocorre calculando-se q, depois subtraindo-se qN de ab.

Ainda que as divisões ainda sejam necessárias, elas podem ser realizadas por um divisor diferente, nesse caso R. Esse divisor, pode ser escolhido de forma que seja uma potência de dois, que transformaria o hardware necessário apenas em operações de deslocamento para a direita, ou pode ser utilizado um valor constante, que resultará apenas em operações de soma e deslocamento. Essas divisões são bem mais rápidas, de forma que o custo computacional do produto modular utilizando a forma de Montgomery resume-se basicamente ao cálculo de um produto convencional.

3 CURVAS ELÍPTICAS

As curvas elípticas usadas em Criptografia são definidas tipicamente em dois tipos de campos finitos: campos de características ímpar p (F_p) e campos de característica par ($\mathbb{F}_{2^m}$). Quando a distinção não é importante nós denotamos ambos eles como F_q, onde $q = p$ ou $q = 2^m$. Em F_p os elementos são inteiros ($0 \leq x < p$) que são combinados usando a aritmética modular. No caso $\mathbb{F}_{2^m}$ é um pouco mais complicado: os elementos do conjunto têm representações diferentes dos elementos do campo, são representados como cadeia de bits para cada escolha do polinômio binário irredutível $f(x)$ do grau m. O conjunto de todos os pares das coordenadas $(x, y) \in \mathbb{F}_q$ em que se tem como resultado, o plano $\mathbb{F}_q \times \mathbb{F}_q$.

Uma curva elíptica é o conjunto dos pontos do plano cujas coordenadas satisfazem a uma determinada equação cúbica junto com um ponto na infinidade. No caso dos campos de característica ímpar, com $p > 3$ a equação definida de $E(F_p)$ pode ser escrita:

$$y^2 = x^3 + ax + b \tag{2}$$

onde $a \in F_p$ e $b \in F_p$ são constantes. No outro caso, os de característica par, a equação definida de $E(\mathbb{F}_{2^m})$ pode ser escrita:

$$y^2 + xy = x^3 + ax^2 + b \tag{3}$$

onde $a \in \mathbb{F}_{2^m}$ e $b \in \mathbb{F}_{2^m}$ são constantes e $b \neq 0$.

Embora o ponto no infinito O não tenha coordenadas, é conveniente representá-las, usando um par das coordenadas que não satisfazem à equação definida, por exemplo, $O = (0,0)$ se $b \neq 0$ e $O = (0,1)$. Nesse trabalho, trataremos apenas dos campos com características pares, devido a facilidade de desenvolvimento dos algoritmos utilizados para a implementação em hardware.

Os pontos de uma curva elíptica dão forma a um grupo abeliano $(E(F), +)$ com O o ponto distinto na infinidade. E nos campos de característica ímpar, o negativo de um ponto $P = (x, y)$ possui valor, $-P = (x, x + y)$ para $P \in E(\mathbb{F}_{2^m})$. Daí, pode-se definir a operação de adição da seguinte forma:

- Se $Q = O$ então $P + Q = P$

Se $Q = -P$ então $P + Q = O$

- Se $Q \neq P$, então $P + Q = R$, onde:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$

$$y_R = \lambda(x_p - x_R) - y_P$$

$$\lambda = (y_P + y_Q)/(x_P + x_Q)$$

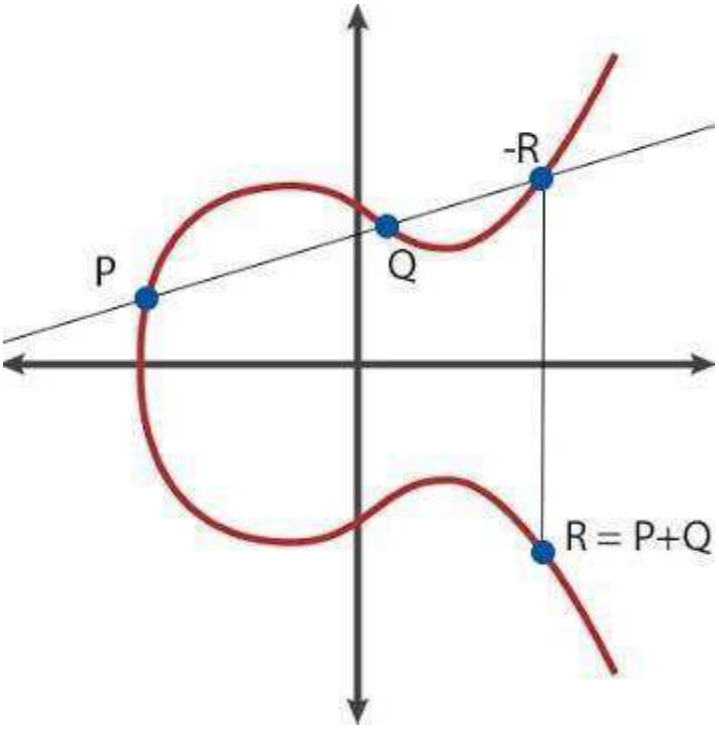
- Se $Q = P$ então $P + Q = R$, onde:

$$x_R = \lambda^2 + \lambda + a$$

$$y_R = x_p^2 + (\lambda + 1)x_R$$

$$\lambda = x_p + y_P/x_P$$

Figura 2 – Esquema simplificado da relação entre pontos nas curvas elípticas.

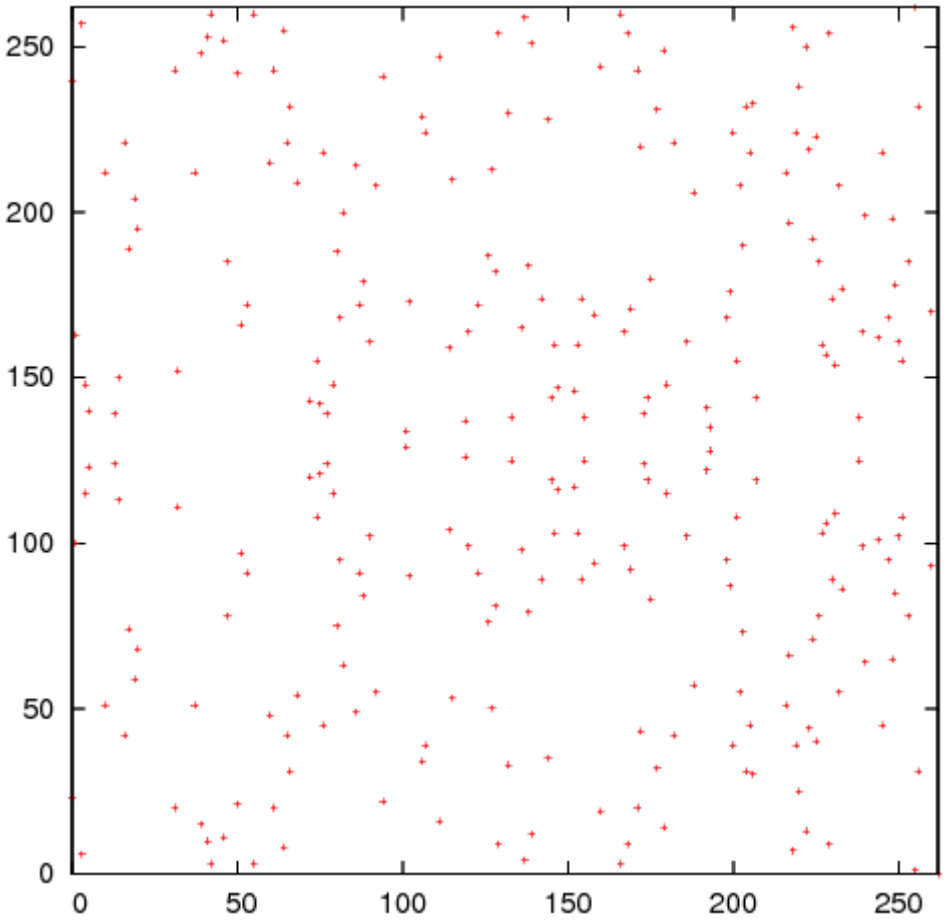


Fonte: Youtube.com – Autor: Robert Pierce.

O ponto mais importante do grupo abeliano formado pelos pontos de uma curva elíptica, consiste é a propriedade de multiplicação escalar, uma operação $k.P$, onde k é um inteiro positivo e P é um ponto na curva elíptica. Calcular $k.P$ significa adicionar o ponto P exatamente $k - 1$ vezes a ele mesmo, que resulta em um outro ponto Q da curva elíptica. A segurança do sistema, se determina pelo problema matemático que é recuperar k, quando P e Q são conhecidos, este problema é conhecido como o Problema Discreto do Logaritmo da Curva Elíptica, e a dificuldade suposta de diversos problemas

relacionados ao logaritmo discreto no subgrupo $E(\mathbb{F}_q)$ permite o uso da Criptografia de Curva Elíptica.

Figura 3 – Exemplo de pontos obtidos a partir da curva



Fonte: ECC Tutorial, Certicom

4 ALGORITMOS

Utilizando-se da matemática definida por Montgomery, definiu-se a série de algoritmos que serão utilizados para a implementação do hardware desejado. As operações aritméticas no campo binário necessárias para implementação em hardware são: adição, multiplicação e inversão definidas no domínio F_{2^m}. A representação padrão base que é utilizada em nossa implementação é a redução polinomial:

$$F(x) = x^m + G(x) = x^m + \sum_{i=0}^{m-1} g_i x^i \tag{4}$$

Onde, $g_i \in \{0,1\}$ para $i = 1, \ldots, m-1$ e $g_0 = 1$. Seja α uma raiz de $F(x)$, então representamos $A \in F_{2^m}$ em base polinomial como sendo:

$$A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i \, , a_i \in F_{2^m} \tag{5}$$

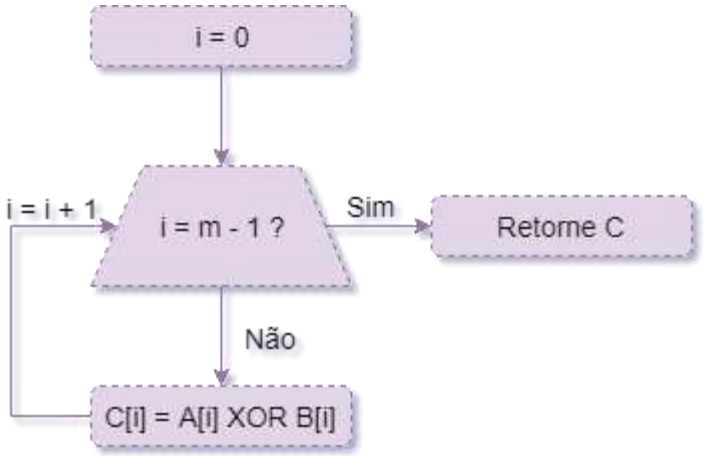
4.1 OPERAÇÕES MATEMÁTICAS

4.1.1 ADIÇÃO

Adição no domínio F_{2^m} é a operação mais simples entre as outras que serão definidas, uma vez que a adição de bits no domínio F_2 é mapeada simplesmente em operações XOR, tanto em software como em hardware.

$$C \equiv A + B \; mod \; F(\alpha) \equiv (a_{m-1} \oplus b_{m-1}) \alpha^{m-1} + \cdots + (a_0 \oplus b_0) \tag{6}$$

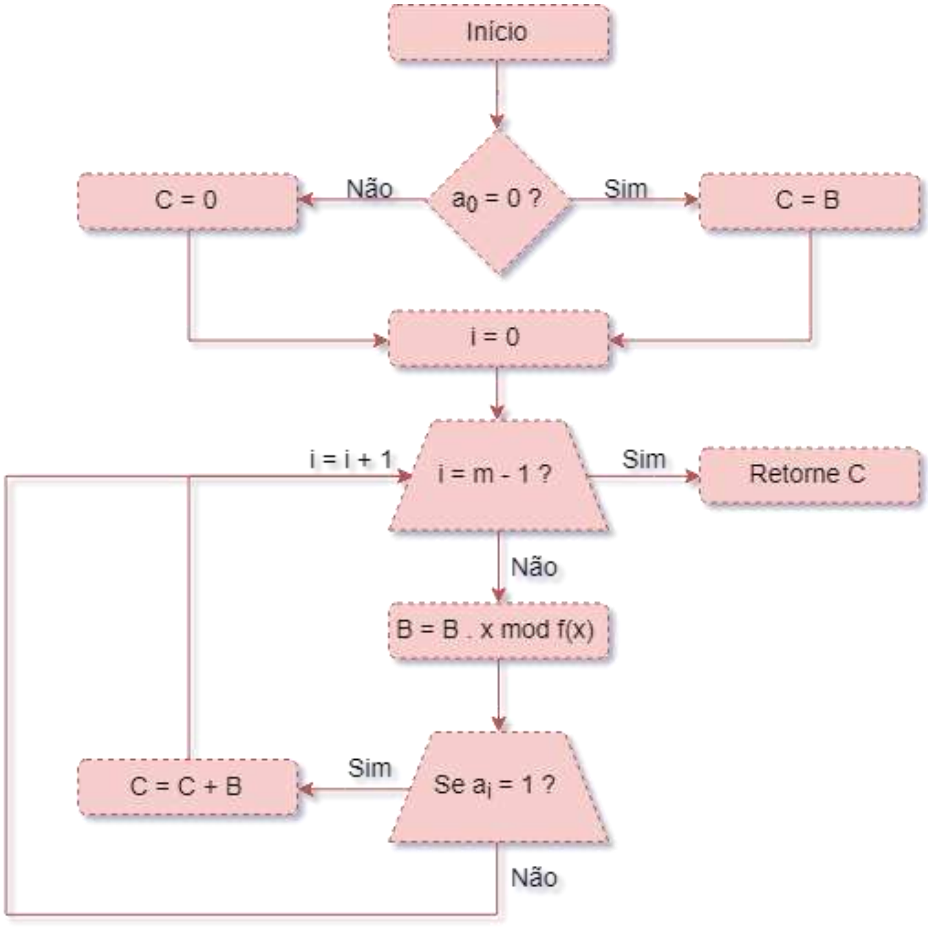
Figura 4 - Algoritmo de soma modular.



Fonte: O próprio autor.

4.1.2 MULTIPLICAÇÃO

Figura 5 – Algoritmo de multiplicação modular.



Fonte: O próprio autor.

A multiplicação dos elementos A e B, $A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i$ e $B(\alpha) = \sum_{i=0}^{m-1} b_i \alpha^i$, sendo $A, B \in \mathbb{F}_{2^m}$. É dada por:

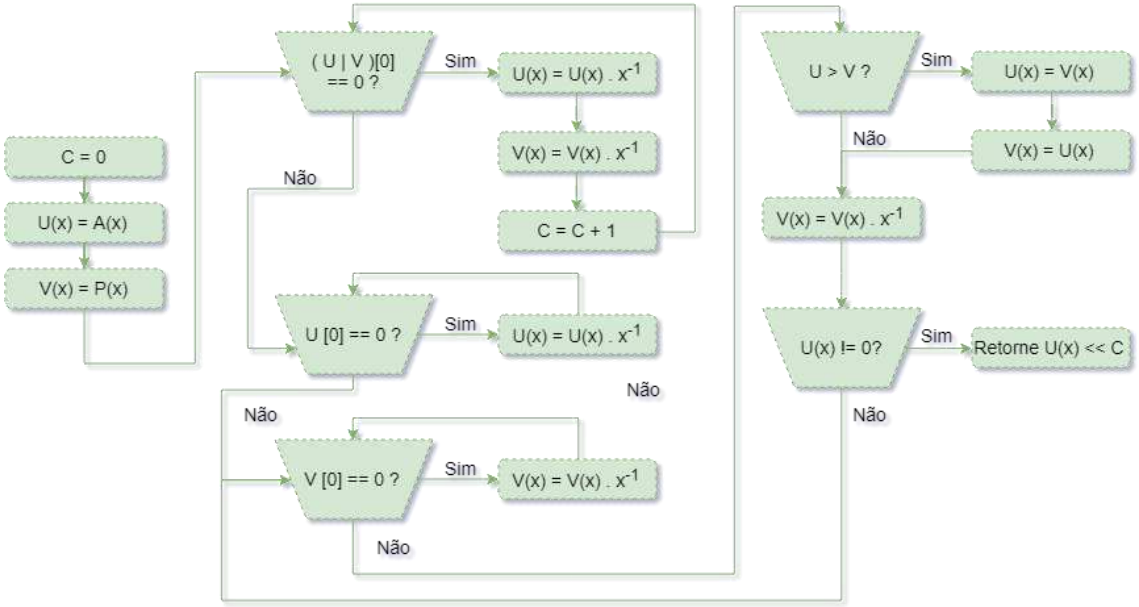
$$C(\alpha) = \sum_{i=0}^{2m-2} c_i \alpha^i \equiv A(\alpha) \cdot B(\alpha) \; mod \; F(\alpha) \tag{7}$$

4.1.3 INVERSÃO

Para implementar a divisão modular necessária no algoritmo de multiplicação, visto anteriormente, no tópico 2.2, utilizaremos uma otimização do Algoritmo Estendido Euclidiano Binário, que utiliza apenas operações de soma, subtração e deslocamento para realização de inversões modulares.

$$x = A^{-1} \; mod \; F(a), \tag{8}$$

Figura 6 – Algoritmo de inversão modular.



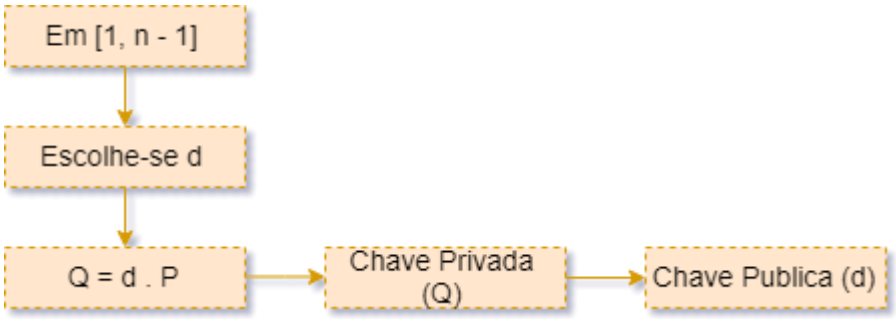
Fonte: O próprio autor.

4.2 PROTOCOLOS

Dessa matemática definida por Montgomery, define-se então os algoritmos que serão responsáveis por realizar as operações básicas da criptografia assimétrica, a assinatura e a validação da assinatura.

Porém, antes de partir para os mais complexos, define-se inicialmente, como é o processo de definição das chaves pública e privada utilizadas nos algoritmos a seguir.

Figura 7 – Algoritmo de definição das chaves.

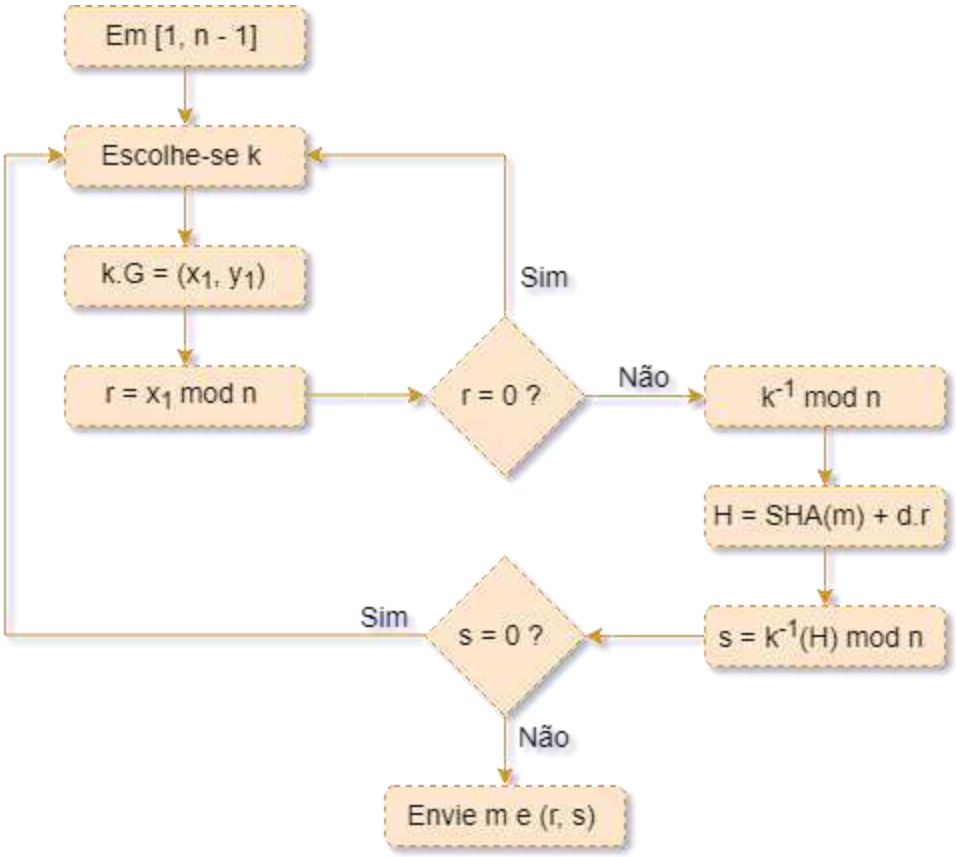


Fonte: O próprio autor.

4.2.1 ECDSA

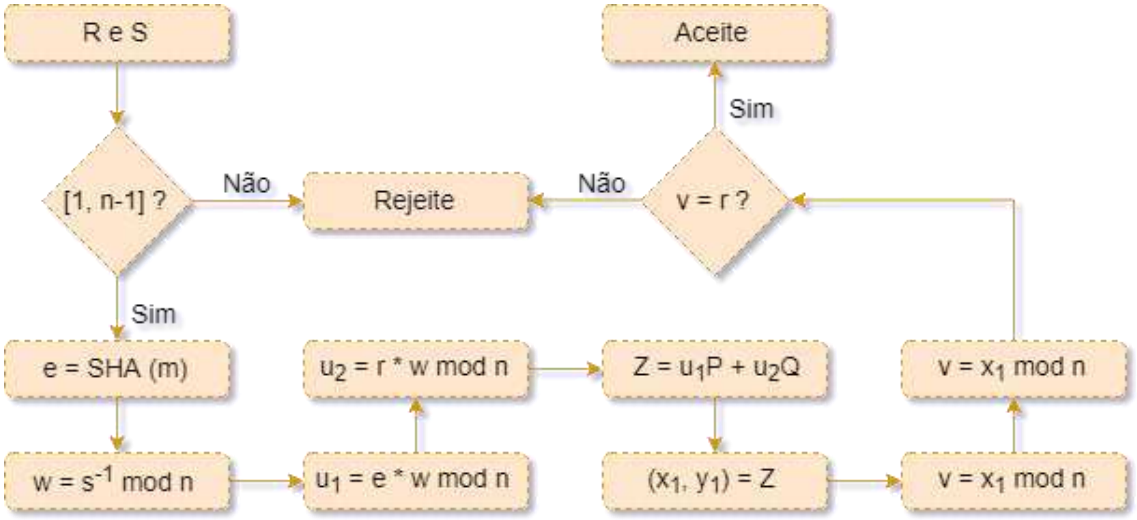
O algoritmo de assinatura digital por curvas elípticas, ou ECDSA (Elliptic Curve Digital Signature Algorithm) é o algoritmo utilizado para as duas operações realizadas pelo sistema de criptografia de curvas elípticas. Desta forma, o mesmo é responsável pela descrição de como ocorre a assinatura e a verificação de assinatura do sistema. Ambas as operações, assim como o processo de definição das chaves, são definidas abaixo.

Figura 8 – Algoritmo de assinatura.



Fonte: O próprio autor.

Figura 9 – Algoritmo de verificação de assinatura.

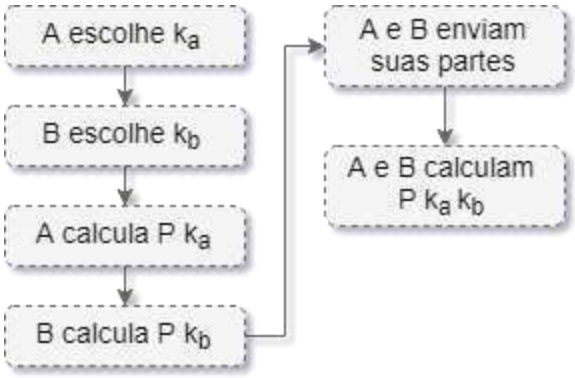


Fonte: O próprio autor.

4.2.2 DIFFIE-HELLMAN

O algoritmo do protocolo Diffie-Hellman, que é um protocolo de criptografia assimétrica proposto para definir uma chave secreta compartilhada entre duas partes. Se dois usuários, A e B, inicialmente definem que irão utilizar uma curva específica e um campo específico, é possível que os mesmos compartilhem de uma chave secreta.

Figura 10 – Algoritmo Diffie-Hellman.



Fonte: O próprio autor.

Utilizando-se deste algoritmo, é possível que as duas partes definam uma chave secreta a ser utilizada em um sistema de criptografia simétrica, em que ambas as partes compartilham da mesma chave, utilizando um canal inseguro e sem comprometer a segurança do sistema.

5 VALIDAÇÃO EM SOFTWARE

A fim de validar o funcionamento dos algoritmos explanados no capítulo anterior, e preparar um ambiente de verificação para a verificação do hardware a ser desenvolvido, foram definidos e executados diversos códigos em linguagens de programação de alto nível, nesse caso *Python*, por sua facilidade de implementação, ampla bibliografia e comunidade engajada.

A biblioteca utilizada define alguns vetores de testes definidos pelo NIST, e foi utilizada como forma de conhecer-se valores intermediários das operações básicas, a fim de facilitar a depuração das mesmas no hardware a ser implementado. Foi definido então, uma operação modelo, dada como:

$$R = a \,/\, b \bmod P \;\equiv a * b^{-1} \bmod P \tag{9}$$

Uma vez definidos valores de entrada, saída e intermediários para cada um dos módulos a serem validados, somador, multiplicador e inversor, construiu-se a

Tabela 2.

Tabela 2 – Resultados para operações matemáticas do software

a	$0x07B6882CAAEFA84F9554FF8428BD88E246D2782AE2$
b	$0x0369979697AB43897789566789567F787A7876A654$
P	$0x800000000000000000000000000000000000000C9$
b^{-1}	$0x6350B655FA1E950BA49F0675D07B9330BB1EE5DC7$
R	$0x3EC51A40015303 6A56826715448DB78D9AE999425$

Fonte: O próprio autor.

Após a documentação dos resultados obtidos nas operações matemáticas, partiu-se então para a definição das chaves, pública e privada, *d* e *Q* respectivamente, que serão utilizadas nos algoritmos de assinatura e validação, os resultados podem ser observados na Tabela 3.

Tabela 3 – Definição das chaves pública e privada

d	$0x3$
Q	$0x23CC6C3C701CA9C669C03369C037E696F69631013$

Fonte: O próprio autor.

Finalmente, para as operações intermediárias dos algoritmos criptográficos, preenchendo-se a Tabela 4, com os valores de *k*, *R*, *S*, *m*.

Tabela 4 - Valores das assinaturas da mensagem codificada

k	$0x1B80C371A8127F626B515B6C735A3E0DAF2EA3591$
m	$0x21217373656E6B726164206F6C6C6548$
R	$0x18838334B3F4FA62DF16D1EAB6B1D0EFE5B8843C3$
S	$0x5B7771B08C0B9AAD0F7654AD4065D7C98B3E2853F$

Fonte: O próprio autor.

O método utilizado para verificação do funcionamento do software foi justamente utilizar os algoritmos de validação de assinatura para garantir que todos os resultados foram congruentes.

5.1 DEFINIÇÃO DE PARÂMETROS DAS CURVAS

A curva elíptica E definida sobre o conjunto F_{2^m} é especificada de acordo com os seus coeficientes $a, b \in F_{2^m}$ da equação $y^2 + xy = x^3 + ax^2 + b$. O número de pontos de E definida sobre o conjunto é dado por nh, onde n é um número primo, e h é chamado de cofator.

De forma a melhorar a criptografia, e obter as informações de entrada e saída de vetores de teste, foram utilizados curvas pré-definidas para cada modo de operação, a definição de cada parâmetro da curva, assim como seu polinômio irredutível são apresentados nas Tabelas Tabela 5 e Tabela 6.

Tabela 5 – Parâmetros da Curva B-163

$f(x)$	$x^{163} + x^7 + x^6 + x^3 + 1$
A	$0x07\ B6882CAA\ EFA84F95\ 54FF8428\ BD88E246\ D2782AE2$
B	$0x07\ 13612DCD\ DCB40AAB\ 946BDA29\ CA91F73A\ F958AFD9$
G_x	$0x0003\ 69979697\ AB438977\ 89566789\ 567F787A\ 7876A654$
G_y	$0x0000\ D51FBC6C\ 71A0094F\ A2CDD545\ B11C5C0C\ 797324F1$
n	$0x03\ FFFFFFFF\ FFFFFFFF\ FFFF48AA\ B689C29C\ A710279B$
h	$0x2$

Fonte: O próprio autor.

Tabela 6 – Parâmetros da Curva B-233

$f(x)$	$x^{233} + x^{74} + 1$
A	$0x0000\ 00000000\ 00000000\ 00000000\ 00000000\ 000000000$

	0000000 00000001
B	$0x$0066 647$EDE6C$ 332$C7F8C$ 0923BB58 213$B333B$ 20$E9CE$42 81FE115F 7$D8$F90AD
G_x	$0x000000FA$ $C9DFCBAC$ 8313BB21 39$F1BB$75 5$FEF65BC$ 391$F8B36$ $F8F8EB73$ 71$FD558B$
G_y	$0x$00000100 6$A08A$419 03350678 $E58528BE$ $BF8A0BEF$ $F867A7CA$ 36716$F7E$ 01F81052
n	$0x$0100 00000000 00000000 00000000 0013E974 $E72F8A$692 2031D26 03$CFE0D$7
h	$0x2$

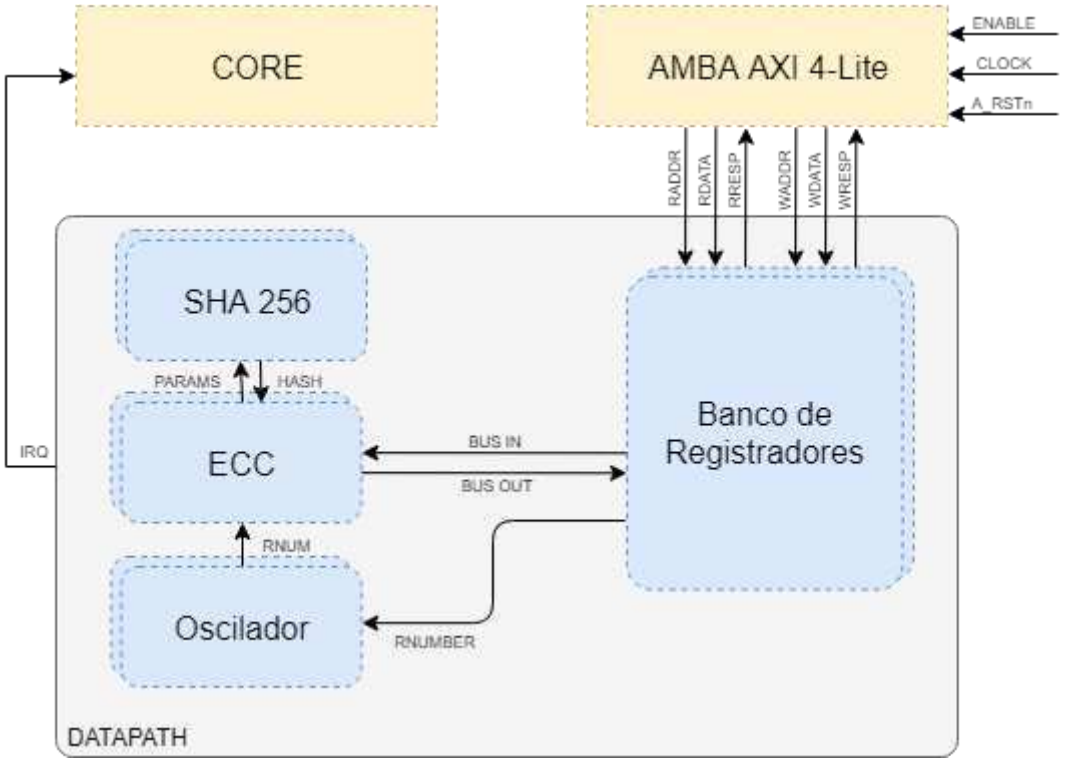
Fonte: O próprio autor.

6 ARQUITETURA PROPOSTA

A propriedade intelectual aqui apresentada, utiliza uma interface de comunicação *AMBA AXI-4Lite*, propriedade *ARM* e licenciado para utilização em diversos IPs. A escolha do barramento deve-se a ampla variedade de aplicações ao qual o IP pode ser utilizado, uma vez que o protocolo mantido pela *ARM* é amplamente utilizado em diversos *SoCs* fabricados, e inclusive com suporte à algumas *FPGAs*. Permitindo assim, uma validação de seu funcionamento.

Ainda, de forma a garantir melhores resultados de aleatoriedade às chaves geradas pelo IP, é oferecido ao usuário a utilização de um cristal senoidal como fonte do número aleatório utilizado para o algoritmo. Essa modificação, foi acrescentada após estudo de falhas envolvendo pseudoaleatoriedade nos algoritmos de chave assimétrica, onde um caso bastante famoso foi descoberto no *PlayStation 3*, resultando no desbloqueio e possibilidade de venda de jogos não-originais do famoso console da *Sony*. [2], [3]

Figura 11 – Arquitetura proposta do IP

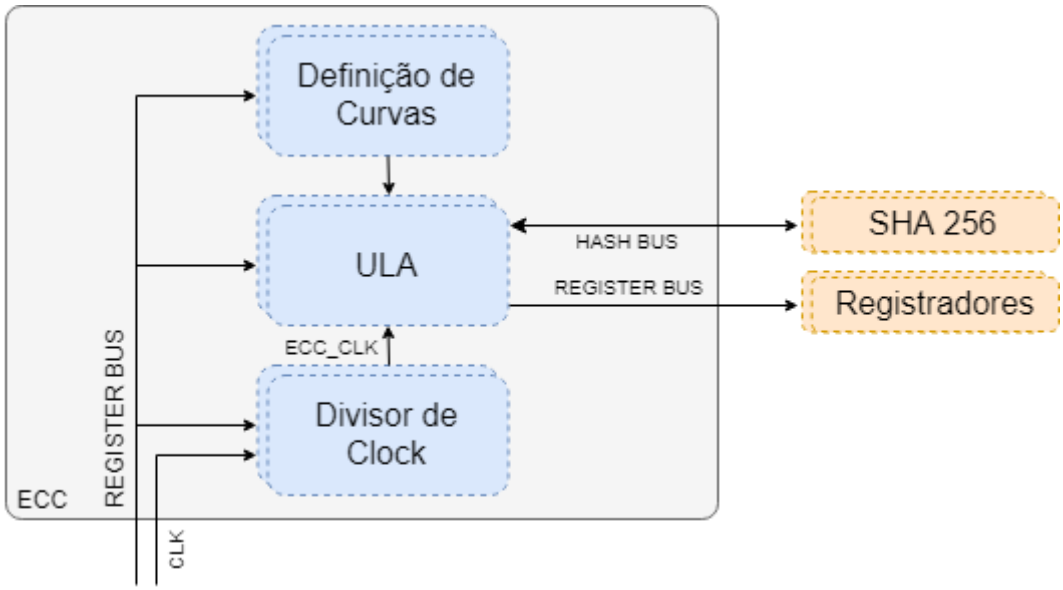


Fonte: O próprio autor.

O banco de registradores possibilita ainda, a configuração de como funcionará o sinal de interrupção do IP, possibilitando que o IP se adapte a diferentes processadores,

utilizando interrupção por borda e nível. Ambos os tipos de interrupção, nível baixo ou nível alto ativos, são saídas do IP, o que permite a fácil integração com o processador.

Figura 12 – Arquitetura do Módulo ECC



Fonte: O próprio autor.

Foi previsto um divisor de *clock* interno ao IP de forma que seja possível ao usuário final, e ao projetista do *SoC*, integrar o IP de forma completa ao barramento, podendo o mesmo operar com o *clock* do barramento apenas para escritas e leituras nos registradores, e utilizar um *clock* mais lento, para a propagação dos sinais internos. Desta forma, é possível que mesmo com grandes lógicas combinacionais sem a utilização de registradores intermediários, garantir que a propagação dos sinais ocorra sem que os registradores sejam afetados.

O módulo de definição de curvas, permite ao usuário a escolha entre 4 curvas elípticas com parâmetros predefinidos pelo NIST, são elas, B-163, B-233. Dessa forma, o usuário pode gerar suas chaves públicas e privadas com maior confiança, uma vez que o oscilador interno garante uma alta pseudoaleatoriedade, e a possibilidade de escolher entre várias curvas, aumenta a dificuldade de que a segurança seja comprometida, pois o usuário pode sempre trocar a curva utilizada para geração da chave.

6.1 MAPA DE MEMÓRIA

Os registradores, utilizados para acesso as informações do IP, são apresentados na Tabela 7, todos apresentam 32 bits, e suas palavras apresentam comportamento *little-*

endian, ou seja, a parte mais significativa da palavra, está no menor endereço de memória, desta forma, para uma mensagem com 16 caracteres ASCII como: *"Hello darkness!!"* deverá ser escrita como: *0x21217373*, *0x656E6B72*, *0x6164206F*, *0x6C6C6548*, preenchendo os endereços de *0x30* à *0x3C*, respectivamente.

O acesso aos endereços da chave, fornece por padrão a chave pública, sendo necessário alterar um bit no registrador de comando para que seja necessário à leitura da chave privada.

Tabela 7 – Mapa de memória do IP.

Endereço	Pseudônimo	Uso	Tipo de Acesso
0x00	Command	Registrador de comando	Escrita/Leitura
0x04	Config	Registrador de configuração	Escrita/Leitura
0x08	Status	Registrador de estado	Leitura
0x0C	RandomNum	Registrador de número aleatório	Escrita/Leitura
0x10	Key[0]	Primeira parte da Chave	Escrita/Leitura
0x14	Key[1]	Segunda parte da Chave	Escrita/Leitura
0x18	Key[2]	Terceira parte da Chave	Escrita/Leitura
0x1C	Key[3]	Quarta parte da Chave	Escrita/Leitura
0x20	Key[4]	Quinta parte da Chave	Escrita/Leitura
0x24	Key[5]	Sexta parte da Chave	Escrita/Leitura
0x28	Key[6]	Sétima parte da Chave	Escrita/Leitura
0x2C	Key[7]	Oitava parte da Chave	Escrita/Leitura
0x30	Msg[0]	Primeira parte da Mensagem	Escrita/Leitura
0x34	Msg[1]	Segunda parte da Mensagem	Escrita/Leitura
0x38	Msg[2]	Terceira parte da Mensagem	Escrita/Leitura
0x3C	Msg[3]	Quarta parte da Mensagem	Escrita/Leitura

Fonte: O próprio autor.

6.1.1 REGISTRADOR DE COMANDO

A Tabela 8 apresenta os *bits* registrador de comando, que possui a função de iniciar as operações no IP, ou enviar um reset síncrono para o mesmo.

Tabela 8 – Registrador de comando

Bit	**[31]**	**[30-3]**	**[2]**	**[1]**	**[0]**
----	Reset	Reservado	Validar assinatura	Assinar	Gerar Chave

Fonte: O próprio autor.

Escritas realizadas no registrador de comando são automaticamente limpas, sendo o valor de leitura retornado sempre 0x00000000.

6.1.1.1 RESET

Escrever nesse bit, 31, gerará automaticamente uma borda de reset síncrono com o barramento, sendo possível reiniciar todo o IP ao seu estado inicial independente de um reset externo ser realizado.

6.1.1.2 VALIDAR ASSINATURA

Escrita nesse bit, 2, irá enviar um sinal de início de operação de validação de mensagem para o módulo de validação e assinatura, será considerada a mensagem contida nos registradores de mensagem.

6.1.1.3 ASSINAR

Escrita nesse bit, irá enviar um sinal de início de operação de assinatura de mensagem para o módulo de validação e assinatura, será considerada a mensagem contida nos registradores de mensagem. Como o hardware necessário para as operações de validar assinatura e assinar é compartilhado, a preferência de operação nesse caso é da assinatura, sendo que se for escrito 1 em ambos os bits, a operação de validação será ignorada.

6.1.1.4 CRIAR CHAVES

Escrever nesse bit, irá enviar um sinal de início de operação para o módulo de geração das chaves pública e privada. Como os registros de chave pública e privada são utilizados para as operações de assinatura e validação, se escrito 1 neste bit, as demais operações disponíveis serão ignoradas.

6.1.2 REGISTRADOR DE CONFIGURAÇÕES

A Tabela 9 apresenta os *bits* registrador de configurações, que permite ao usuário configurar alguns parâmetros do IP

Tabela 9 – Registrador de configurações

Bit	**[31-16]**	**[15-8]**	**[7-3]**	**[2]**	**[1]**	**[0]**
---	Reservado	*Clock* Div	Reservado	Rand	Pub_Privn	Curva

Fonte: O próprio autor.

6.1.2.1 CURVA DE OPERAÇÃO

Esta parte do registro define a curva de operação que será utilizada para as operações do IP.

- 0: B-163
- 1: B-233.

6.1.2.2 DEFINIÇÃO DE CHAVE LIDA

De forma a aumentar a segurança da chave privada gerada nos algoritmos, e economia na quantidade de memória ocupada no barramento AMBA AXI, é possível alterar entre qual chave será lida nos registradores da Chave, endereços $0x10$ até $0x4C$, sendo que quando o valor deste bit for 0, será possível a leitura da chave pública, e 1 para leitura da chave privada.

6.1.2.3 DEFINIÇÃO DA ALEATORIEDADE

Como o algoritmo de geração das chaves pública e privadas, bem como a segurança inerente à essas chaves é altamente dependente do número aleatório utilizado, é fornecido ao usuário utilizar apenas o oscilador senoidal interno ao IP, ou ainda, que o mesmo forneça um número inteiro que será utilizado em conjunto com a saída do oscilador como definição do k.

6.1.2.4 DIVISOR DE *CLOCK*

Como alternativa de forma a garantir o funcionamento do IP independentemente do *clock* do barramento ao qual o mesmo é inserido, o *clock* que será utilizado internamente passa primeiramente por um divisor, sendo possível realizar a redução da velocidade do relógio por um fator de até 256. O *clock* interno, segue funcionamento de acordo com a equação abaixo:

$$Clk_{int} = Clk_{AMBA}/(1 + Div_{clk}) \quad (10)$$

6.1.3 REGISTRADOR DE ESTADO

No registrador de estado, o usuário tem as informações do momento atual do IP, e o resultado da operação de validação.

Tabela 10 - Registrador de estado

Bit	[31-6]	[3]	[2-0]
---	Reservado	OK	Operando

Fonte: O próprio autor.

6.1.3.1 REALIZANDO OPERAÇÃO

Cada bit de operação corresponde a um estado de operação do IP. A ordem de exibição segue a mesma lógica do registrador de comando, sendo então:

- 1: Gerando chaves; 2: Assinando; 4: Validando assinatura.

6.1.3.2 ASSINATURA CONFERIDA (OK)

O bit de assinatura conferida indica que a operação de validação de assinatura terminou de ser realizada e que a mensagem contida nos registradores de mensagem confere com a assinatura enviada. Caso a operação seja finalizada e não seja escrito 1 neste bit, é porque a chave indicada não é válida.

6.1.4 DEMAIS REGISTROS

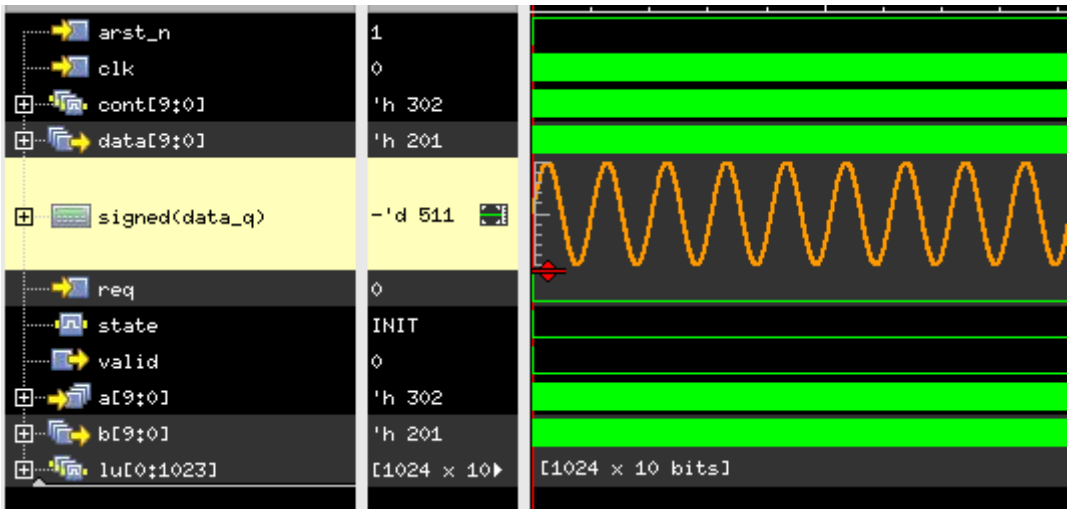
Durante as operações do IP, fica inviabilizada a escrita em todos os registradores do IP, sendo que apenas o registrador de comando fica habilitado, de forma que seja possível ao usuário enviar um sinal de reset síncrono caso ele deseje abortar uma operação. Este reset, também limpará o conteúdo de todos os registrados, de tal forma, que será necessário redefinir as configurações desejadas, e os parâmetros utilizados para realizar a operação desejada.

7 SIMULAÇÕES

Para obtenção das formas de onda apresentadas nesse tópico, foram realizadas simulações no software *Incisive Enterprise Simulator*. O simulador em questão, foi utilizado para verificação de erros de design do código escrito em *SystemVerilog* responsável pela descrição em hardware da arquitetura apresentada no tópico anterior.

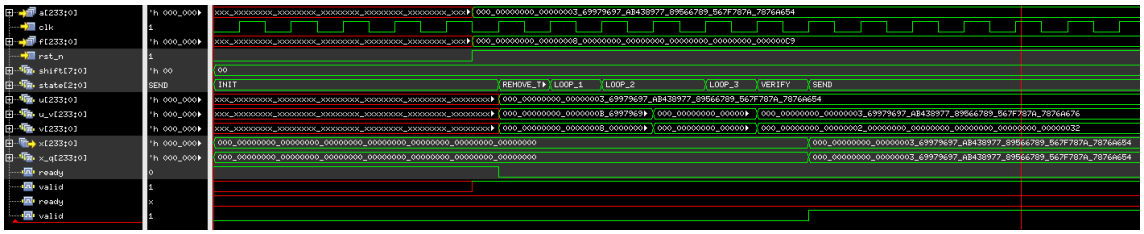
Exemplos de operação de cada um dos módulos internos vistos na Arquitetura do IP, podem ser verificados nas Figuras Figura 13, Figura 14, Figura 15, Figura 16, Figura 17 e Figura 18.

Figura 13 - Funcionamento do oscilador interno



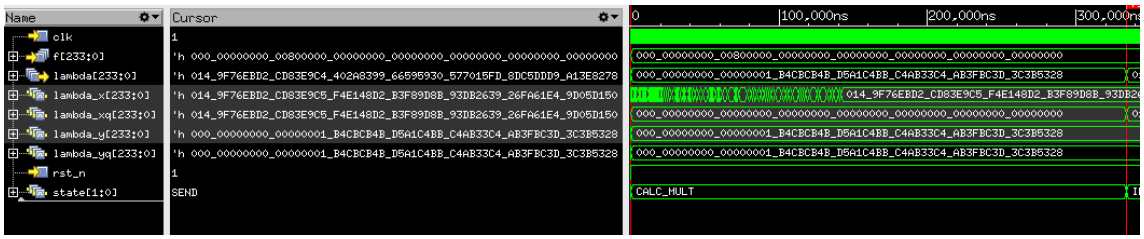
Fonte: O próprio autor.

Figura 14 - Exemplo de cálculo de inversão modular



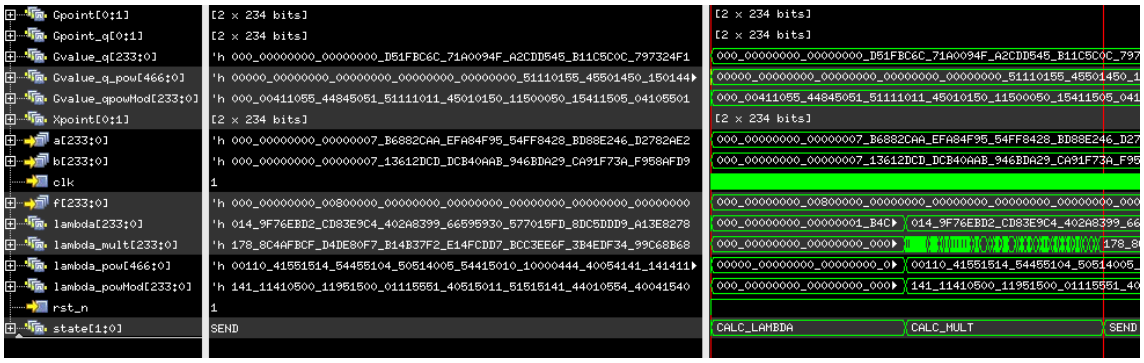
Fonte: O próprio autor.

Figura 15 - Exemplo de cálculo do lambda



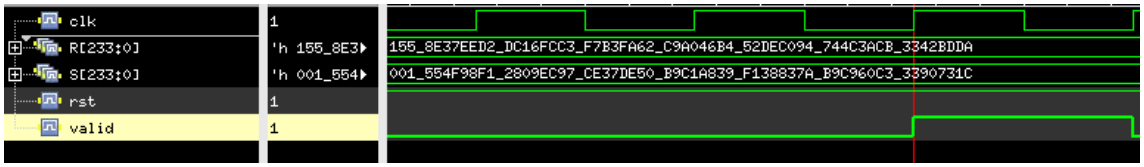
Fonte: O próprio autor.

Figura 16 - Exemplo de um ponto $Y = k.G$



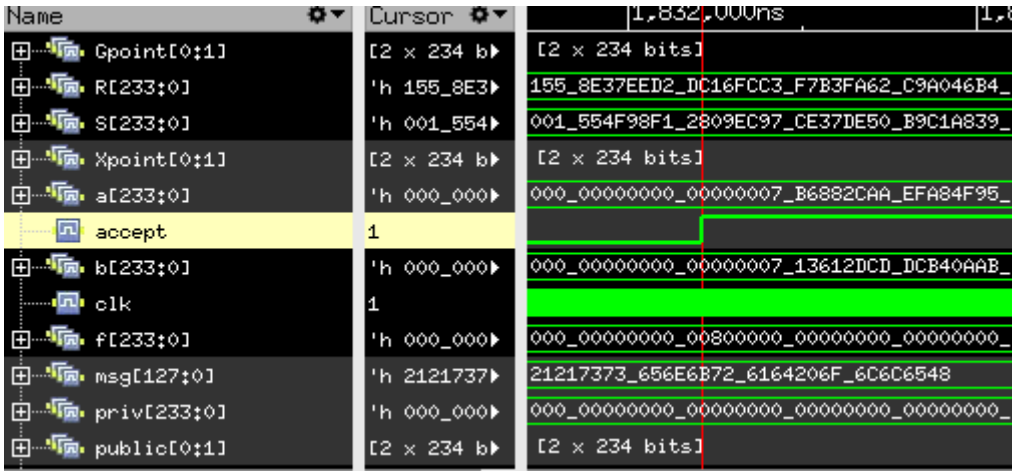
Fonte: O próprio autor.

Figura 17 - Exemplo de assinatura válida para uma mensagem M



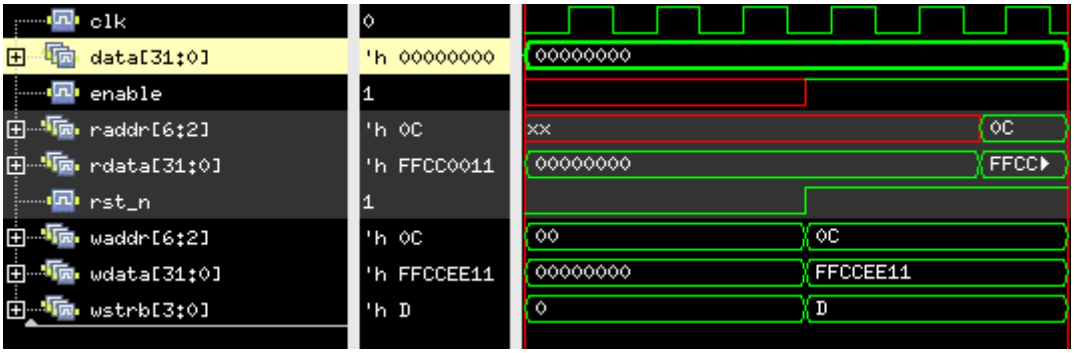
Fonte: O próprio autor.

Figura 18 - Exemplo de aceitação de uma assinatura R, S para uma mensagem M



Fonte: O próprio autor.

Figura 19 - Exemplo de leitura e escrita nos registradores internos



Fonte: O próprio autor.

Após a verificação do funcionamento dos módulos internos, cada um individualmente, e integração dos mesmos em um módulo superior, foi adicionada a interface *AMBA AXI-4 Lite* de forma a permitir o acesso ao banco de registradores do IP, conforme pode ser visto no exemplo da Figura 19.

Finalmente, com o IP integrado, foi realizada a verificação de seu funcionamento, utilizando os vetores definidos no código em *Python*, verificando-se após as simulações que os resultados foram exatamente os mesmos apontados pela versão em software, confirmando o bom funcionamento do IP implementado.

8 CONCLUSÃO E TRABALHOS FUTUROS

A utilização de um IP de criptografia assimétrica é de grande valor para SoCs que utilizam processadores mais simples, sem aceleração de hardware para as várias operações realizadas em sequência para garantir a segurança do algoritmo. Pensando em um contexto mais amplo para a utilização do mesmo, é possível sugerir como melhoria ao IP já implementado, a utilização de barramentos de transferência de informação mais robustos, como AMBA AXI-4 Full, ou o AMBA AHB, ambos também bastante utilizados na indústria.

Isso posto como melhorias ao já existente RTL, é de suma importância que haja uma verificação funcional de forma completa do IP, onde sejam explorados pontos chaves, afim de verificar não só o funcionamento em todos os casos do RTL, bem como sua capacidade de permanecer operacional, recebendo diferentes estímulos.

Por fim, uma validação do hardware com síntese lógica, ou verificação em gate-level, de forma a verificar que a trilhas e lógicas combinacionais não acrescentam atraso capaz de comprometer o funcionamento do IP.

Neste trabalho, foi discutido a importância da utilização de algoritmos de criptografia de chave assimétrica dentre eles, em especial o de curvas elípticas, de forma a justificar a utilização deste algoritmo em detrimento dos demais. Isto posto, foram realizados testes em software, de forma a validar os algoritmos obtidos de trabalhos anteriores e a partir destes propor e implementar uma arquitetura capaz de funcionalmente, obter os mesmos resultados.

O IP em suas simulações obteve os mesmos resultados que o banco de testes fornecidos pelas aplicações anteriores e foi bem-sucedido em testes simples de reset assíncrono, comportando-se conforme o esperado para todos os vetores de testes definidos e executados.

9 REFERÊNCIAS BIBLIOGRÁFICAS

1. ARNOLD, D. M.; RANGASWAMY, K. M. **Abelian Groups and Modules**. New York, 1996.

2. BORZA, M. The Sony PlayStation 3 hack deciphered: what consumer-electronics designers can learn from the failure to protect a billion-dollar product ecosystem. **EDN**, 19 Maio 2011. Disponivel em: <https://www.edn.com/design/consumer/4368066/The-Sony-PlayStation-3-hack-deciphered-what-consumer-electronics-designers-can-learn-from-the-failure-to-protect-a-billion-dollar-product-ecosystem>.

3. FILDES, J. iPhone hacker publishes secret Sony PlayStation 3 key. **BBC News**, 6 Janeiro 2011. Disponivel em: <https://www.bbc.co.uk/news/technology-12116051>.

4. TAVERNE, J.; FAZ-HERNÁNDEZ, A.; ARANHA, D. F. Software Implementation of Binary Elliptic Curves: Impact of the Carry-Less Multiplier on Scalar Multiplication. **Cryptographic Hardware and Embedded Systems – CHES 2011**, p. 108-123, 2011.

5. SAVAS, E.; KOC, C. K. The Montgomery modular inverse-revisited. **IEEE Transactions on Computers**, v. 49, n. 7, p. 763-766, 2000.

6. PETER, S.; LANGENDORFER, P.; PIOTROWSKI, K. Flexible Hardware Reduction for Elliptic Curve Cryptography in GF(2m). **2007 Design, Automation & Test in Europe Conference & Exhibition**, 2007.

7. OKEYA, K.; KURUMATANI, H.; SAKURAI, K. Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications. **Public Key Cryptography Lecture Notes in Computer Science,** p. 238-257, 2000.

8. MENEZES, A. J.; C., V. O. P.; VANSTONE, S. A. **Handbook of applied cryptography**. [S.l.]: [s.n.], 2001.

9. LÓRENCZ, R.; HLAVÁČ, J. Subtraction-free Almost Montgomery Inverse algorithm. **Information Processing Letters**, v. 94, n. 1, p. 11-14, 2005.

10. BROWN, M.; HANKERSON, D.; LÓPEZ, J. Software Implementation of the NIST Elliptic Curves Over Prime Fields. **Topics in Cryptology — CT-RSA**, p. 250–265, 2001.

11. AMARA, M. A. S. A. Hardware Implementation of Elliptic Curve Point Multiplication over GF(2m) for ECC protocols. **Internation Journal for Information Security**, p. 106-112, 2012.

12. ALFRED J. MENEZES, P. C. V. O. A. S. A. V. **Handbook of Applied Cryptography**. [S.l.]: CRC Press, 2001. Disponivel em: <http://cacr.uwaterloo.ca/hac/>.

13. ELLIPTIC Curve Cryptography Tutorial. Disponivel em: <https://www.johannes-bauer.com/compsci/ecc/?menuid=4>>.