MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

ATHOS COTTA COUTO
NAUM AZEREDO FERNANDES BARREIRA

DESENVOLVIMENTO DE SISTEMA CRIPTOGRÁFICO PARA CRIAÇÃO DE CONTRATOS EM REDES PÚBLICAS

Rio de Janeiro
2017

INSTITUTO MILITAR DE ENGENHARIA

ATHOS COTTA COUTO
NAUM AZEREDO FERNANDES BARREIRA

DESENVOLVIMENTO DE SISTEMA CRIPTOGRÁFICO PARA CRIAÇÃO DE CONTRATOS EM REDES PÚBLICAS

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Jose Antônio Moreira Xexeo - D.Sc.

Rio de Janeiro
2017

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270

004.65 Couto, Athos Cotta
C871d Desenvolvimento de Sistema Criptográfico para Criação de Contratos em Redes Públicas / Athos Cotta Couto, Naum Azeredo Fernandes Barreira, orientado por Jose Antônio Moreira Xexeo - Rio de Janeiro: Instituto Militar de Engenharia, 2017.

45p.: il.

Projeto de Fim de Curso (PROFIC) - Instituto Militar de Engenharia, Rio de Janeiro, 2017.

1. Curso de Engenharia de Computação - Projeto de Fim de Curso. 1. Criptografia. I. Barreira, Naum Azeredo Fernandes . II. Xexeo, Jose Antônio Moreira . II. Título. III. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

ATHOS COTTA COUTO
NAUM AZEREDO FERNANDES BARREIRA

DESENVOLVIMENTO DE SISTEMA CRIPTOGRÁFICO PARA CRIAÇÃO DE CONTRATOS EM REDES PÚBLICAS

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: José Antônio Moreira Xexéo - D.Sc.

Aprovado em 05 de Outubro de 2017 pela seguinte Banca Examinadora:

José Antônio Moreira Xexéo - D.Sc. do IME - Presidente

Anderson Fernandes Pereira dos Santos - D.Sc. do IME

Ronaldo Ribeiro Goldschmidt - D.Sc. do IME

Rio de Janeiro
2017

Aos amigos, familiares e companheiros, fundamentais para nosso crescimento não somente profissional como também pessoal.

AGRADECIMENTOS

Agradeço a todas as pessoas que me incentivaram, apoiaram e possibilitaram esta oportunidade de ampliar meus horizontes. Meus familiares, amigos e mestres.

Em especial ao meu Professor Orientador Dr. José Antônio Moreira Xexéo, por suas disponibilidades e atenções.

"...if you aren't, at any given time, scandalized by code you wrote five or even three years ago, you're not learning anywhere near enough "

NICK BLACK

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

ABREVIATURAS

ITU	-	International Telecommunication Union
NIST	-	National Institute of Standards and Technology
OECD	-	Organisation for Economic Co-operation and Development

RESUMO

Confiança é um atributo central nas sociedades modernas. Como a computação é a ciência que lida com o armazenamento, processamento e transporte de dados, a preocupação com a confiança se torna mais crítica. Sistemas modernos de confiança dependem da figura do terceiro confiável. Essa dependência pode gerar problemas, já que o terceiro confiável pode ser visto como um ponto único de falha. Sistemas criptográficos podem substituir a figura do terceiro confiável. Esse trabalho tem como objetivo projetar e implementar um sistema capaz de realizar tal substituição, de maneira que se possa gerar transações e provar posse de ativos na rede sem a presença do terceiro confiável.

ABSTRACT

Trust is a central attribute in modern societies. As computing is the science that studies storage, processing and data transport, the importance of trust is critical. Modern systems of truste depend on trusted third parties. This dependence can create problems, since the trusted third party can generate a single point of failure for the systems. Cryptographic systems are able to replace the trusted third parties in trust systems. This work has as goals to project and implement a system capable of achieve such replacement, in a way that proof of ownership of and transactions of assets can be generated without the presence of a trusted third party.

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

No primeiro semestre de 2014, o faturamento dos cartórios brasileiros ultrapassou a marca de um bilhão por mês (O GLOBO, 2014). Contratos, títulos de propriedades e similares são feitos, autenticados e armazenados por fóruns e cartórios, que controlam todo o processo de criação de tais documentos. Aos olhos do governo, tais instituições são consideradas a parte confiável em qualquer negociação, sendo assim capazes de atribuir fé pública aos documentos por elas autenticados. Ainda assim, diversos serviços disponibilizados por tais instituições são altamente ineficientes e caros, como por exemplo a emissão de certidões e autenticação de documentos.

Nakamoto (2008) demonstrou que é possível eliminar a necessidade de um terceiro confiável em transações monetárias digitais, sem a ocorrência do problema do gasto duplo. A técnica por ele descrita, chamada de *blockchain*, tem amplas aplicações, não se limitando apenas a transações monetárias. Ela pode ser utilizada para se gerar contratos em uma rede distribuída, de maneira que todos os contratos sejam armazenados na rede e possam ser auditados por qualquer usuário.

1.2 OBJETIVOS

Esse trabalho tem como objetivo projetar, desenvolver e analisar um sistema criptográfico que permita que:

- Criar e publicar transações de ativos ou bens em um banco de dados público, através de contratos.
- Emitir provas de identidade que confirmem a posse do ativo, sem que tais provas emitam informações sobre o proprietário. Ou seja, o proprietário deve ser capaz e emitir provas de conhecimento-zero.

Para que tais objetivos sejam alcançados, será realizado um estudo dos algoritmos criptográficos presentes na literatura, de maneira a se comparar tais tecnologias e escolher os métodos que integrarão o sistema.

O projeto do banco de dados com as capacidades necessárias para se manter o sistema ficará fora do escopo do projeto. No caso do *Bitcoin*, por exemplo, usa-se a *blockchain* para garantir que uma transação do passado seja permanente, ou seja, não se possa alterar informações relativas a tal transação. Assim sendo, irá se admitir a existência de um banco de dados distribuídos com tal capacidade, que será simulado por um banco de dados comum encapsulado pela aplicação a ser desenvolvida.

1.3 JUSTIFICATIVA

Esse trabalho pode ser justificado com base nos seguintes pontos:

- Crescente popularização do *blockchain*, que hoje funciona como principal tecnologia de moedas como *Bitcoin* e o *Litecoin*.
- Uso da *blockchain* no mercado imobiliário sueco (FORBES, 2017), o que mostra que a tecnologia já é confiável o suficiente para seu uso ser regulamentado.
- Dependência do *blockchain* de várias tecnologias criptográficas, que serão avaliadas e comparadas durante o projeto da solução do problema proposto.

Além disso, esse trabalho abre caminhos para trabalhos relacionados, que podem tratar de:

- Implementação de uma *blockchain* para servir como o banco de dados distribuído para o sistema de contratos.
- Expansão das capacidades do sistema, permitindo contratos envolvendo múltiplas partes e transações de bens que exigem a autorização de múltiplos usuários.

1.4 VIABILIDADE

Para avaliar a viabilidade do projeto, o primeiro ponto a ser checado é se os requisitos do projeto podem ser supridos pelas tecnologias disponíveis. (KOSBA et al., 2016) apresenta um sistema com requisitos similares e que foi implementado com tecnologias criptográficas. Além disso, há presença na literatura de textos referenciando o uso de tecnologias criptográficas para a criação e publicação de contratos em redes de *blockchains*, como em (SWAN, 2015).

1.5 METODOLOGIA

A metodologia adotada para realizar os objetivos propostos foi composta de:

- Levantamento dos Requisitos: foram levantados quais os requisitos necessários ao projeto e quais as características desejáveis do um esquema criptográfico a ser projetado.
- Fundamentação Teórica: foi levantada a literatura relacionada e buscou-se entender quais as principais tecnologias que permitiam as transações de ativos online de maneira descentralizada.
- Projeto do Software: baseado nas tecnologias disponíveis, foi projetado um esquema criptográfico que preservasse a maior quantidade de características desejáveis e fosse passível de ser implementado em tempo hábil. Como o foco era o sistema criptográfico, propôs-se utilizar um banco local para simular um banco de dados distribuído e abstrair os problemas de concorrência e consistência de dados.
- Implementação e Testes: o sistema foi implementado utilizando e testado incrementalmente. Foram utilizadas bibliotecas externas para as operações com curvas elípticas e os esquemas propostos no projeto de software foram totalmente implementados usando apenas tais dependências.

1.6 ESTRUTURA

O texto segue a estrutura delineada a seguir:

- Capítulo 2: apresenta a fundamentação teórica necessária para entender e se projetar o sistema proposto.
- Capítulo 3: utiliza os conhecimentos do capítulo anterior para propor os esquemas criptográficos utilizados no sistema.
- Capítulo 4: apresenta a modelagem utilizada no sistema e decisões de implementação.
- Capítulo 5: conclui a monografia com comentários sobre o projeto e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SEGURANÇA DA INFORMAÇÃO

A Segurança da Informação é prática de prevenção de acesso não autorizado, uso, modificação, inspeção ou destruição, entre outros, da informação. Com o aumento significativo do número de usuários com acesso à Internet, mais de 3,4 bilhões de usuários segundo os dados da International Telecommunication Union (ITU), a importância em proteger ou verificar a autenticidade dos dados se torna cada vez maior.

2.1.1 DEFINIÇÃO

Os conceitos básicos da Segurança da Informação são conhecidos como *Tríade C.I.D.*: Confidencialidade, Integridade e Disponibilidade. Durante os anos princípios mais específicos foram sendo propostos, como em 1992, e revisado em 2002, pela *OECD* (ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT, 2002), foram propostos nove princípios, e em 2004, pela *NIST* (GARY STONEBURNER, 2004), foram propostos 33 princípios. Pela simplicidade e relevância no estudo serão apresentados somente os conceitos da *Tríade C.I.D.*.

- Confidencialidade: Garante que a informação não será conhecida por pessoas que não estejam autorizadas para o tal. A codificação dos dados é o método mais comum para garantir a confidencialidade, outras opções incluem verificação biométrica e *tokens* de segurança.

- Integridade: Garante a consistência, precisão e confiabilidade dos dados em todo seu ciclo de vida. A informação não deve ser alterada em trânsito e medidas devem tomadas para garantir que não possa haver alteração por pessoas não autorizadas. Dentre essas medidas estão permissões dos arquivos e controle de acesso do usuário. Os dados também podem conter *checksum* para verificação da integridade.

- Disponibilidade: Garante que a informação possa ser obtida sempre que for necessário. Isso significa que os sistemas computacionais usados para armazenar e processar os dados, os controles de segurança usados para proteger a informação e os canais de comunicação usados para acessar a informação devem estar funcionando corretamente.

2.2 CRIPTOGRAFIA

Criptografia estuda técnicas matemáticas que permitem comunicação na presença de partes não confiáveis. A comunicação é dita segura quando alcança os seguintes objetivos:

- Confidencialidade: somente partes autorizadas devem ser capazes de ler mensagens enviadas.
- Integridade: partes autorizadas devem ser capazes de identificar alterações sobre a mensagem enviadas.
- Autenticidade: partes autorizadas devem ser capazes de identificar o emissor de cada mensagem.
- Não-repúdio: o emissor não deve ser capaz de negar que enviou uma mensagem.

Vale ressaltar que nem todas as aplicações necessitam das quatro propriedades acima. Portanto existem esquemas de criptografia que possuem diferentes coberturas sobre tais propriedades.

Para atingir tais objetivos os esquemas criptográficos podem ser definidos a partir de duas funções E_e e D_d. E_e recebe uma mensagem m e retorna uma cifra c. D_d recebe uma cifra c e retorna uma mensagem m. Tal esquema deve ser projetado de maneira que existam chaves e e d cujas funções D_d e E_e sejam tais que $D_d(E_e(m)) = m$, ou seja E_e e D_d são funções inversas.

Os esquemas de criptografia podem ser divididos basicamente em dois tipos:

- Chave simétrica: esquemas de chave simétrica são aqueles em que dadas as funções inversas E_e e D_d, pode-se calcular d facilmente caso possua-se e e vice-versa. Nesses esquemas as partes devem trocar chaves que são autenticas e secretas.
- Chave pública: esquemas de chave pública ou de chave assimétrica são aqueles em que dadas as funções inversas E_e e D_d, não é computacionalmente viável de se encontrar d dado e. A chave e é dita a **chave pública**, enquanto a chave d é chamada de **chave privada**.

Esquemas de criptografia simétrica são normalmente mais eficazes computacionalmente. Apesar dessa vantagem, eles não são sempre empregados por causa dos problemas de distribuição e armazenamento de chave. Para se estabelecer um canal seguro de criptografia simétrica, as partes precisam trocar a chave de comunicação, o que não pode ser

feito seguramente sobre um canal inseguro com criptografia simétrica. Além disso, em um cenário onde existem n partes que desejam se comunicar, cada uma teria que guardar a chave das outras $n - 1$ partes, problema que se agrava quando n cresce.

Além disso, esquemas de criptografia assimétricos podem ser usados para implementar esquemas de não-repúdio. Tais sistemas podem ser implementados exigindo que qualquer mensagem transmitida seja assinada pela chave privada do emissor. A parte receptora - e qualquer outra com acesso a mensagem -, que tem acesso a chave pública do emissor, pode então verificar se a mensagem foi de fato enviada pelo emissor.

2.3 ASSINATURA DIGITAL

A assinatura digital é um método de autenticação de mensagens digitais ou documentos, inicialmente usado como um substituto para a assinatura física. O esquema de assinatura digital usual é demonstrado na figura 2.1.

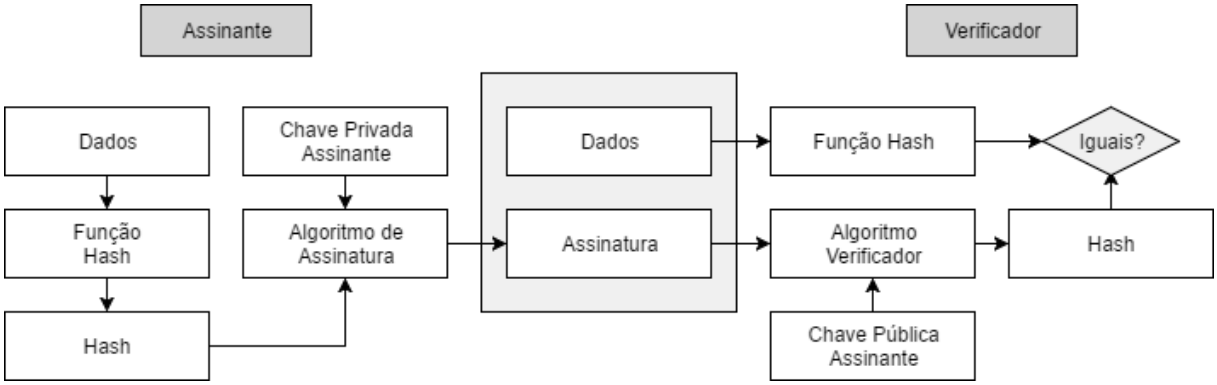

FIG. 2.1: Assinatura digital

2.3.1 DEFINIÇÃO

Uma assinatura válida indica ao destinatário que a mensagem foi confeccionada por um remetente conhecido (autenticação), que a mensagem não foi alterada em trânsito (integridade) e que o remetente não pode negar que enviou a mensagem (não-repúdio).

A assinatura digital não garante que a mensagem seja autêntica, íntegra ou que o suposto remetente não tenha assinado tal mensagem, porém a probabilidade de falsificação de uma assinatura digital é extremamente baixa, com tal probabilidade variando entre os algoritmos de assinatura digital.

2.4 PROVA DE CONHECIMENTO ZERO

Provas de conhecimento zero são métodos que possibilitam que uma parte provadora mostre a validade de uma afirmação sem revelar qualquer outra informação à parte verificadora. No caso em que provar a afirmação exija uma informação secreta só conhecida pela parte provadora, a parte verificadora não será capaz de provar a validade da afirmação a mais ninguém, já que nenhum conhecimento foi compartilhado durante a prova.

Qualquer prova de conhecimento zero deve satisfazer três propriedades:

a) Integralidade: se a afirmação é verdadeira, um verificador honesto será convencido desse fato.

b) Corretude: se a afirmação é falsa, não existe probabilidade significativa do provador convencer o verificador da afirmação.

c) Conhecimento-zero: se a afirmação é verdadeira, nenhum verificador desonesto aprende nada além desse fato.

Vale ressaltar que provas de conhecimento zero não são provas no sentido estrito, pois existe sempre uma pequena probabilidade, o erro de corretude, do verificador ser enganado por um provador desonesto.

2.4.1 PROVA DE IDENTIDADE USANDO O PROBLEMA DO LOGARITMO DISCRETO

Pode-se usar o esquema de identificação de *Schnorr* com o problema do logaritmo discreto para realizar prova de identidade de conhecimento zero.

Para explicar o algoritmo, considere que o usuário Alice queria provar sua identidade para o usuário Bob. Alice escolhe:

- p: um número primo grande.
- q: um número primo grande que divida $p - 1$.
- g: um gerador do grupo $\mathbb{Z}_p$.
- x: um número aleatório no intervalo $[1, q - 1]$.

Alice calcula $X = g^x \mod p$ e publica sua chave pública (X, g, p). Para Bob checar a identidade de Alice:

a) Bob verifica a chave pública de Alice e pede que ela inicie o protocolo.

b) Alice escolhe um número v uniformemente distribuído em $[0, q-1]$ e calcula $V = g^v \mod p$. Alice envia V para Bob.

c) Bob escolhe um número desafio c. c é uniformemente distribuído em $[0, 2^t - 1]$, onde t é o tamanho do desafio. Bob envia c a Alice.

d) Alice calcula $b = v - x * c \mod q$ e envia a Bob.

e) Bob verifica se $V = g^b * X^c \mod p$.

Caso a verificação de Bob seja correta, então Bob aceita a afirmação de Alice sobre sua identidade.

De acordo com (HAO, 2017), o protocolo de *Schnorr* é conhecido por possuir as seguintes propriedades:

a) Integralidade: se a afirmação é verdadeira, um verificador honesto será convencido desse fato.

b) Corretude: um adversário que não sabe o logaritmo discreto de X possui uma probabilidade irrisória (em torno de 2^{-t}) de passar no teste do verificador.

c) Conhecimento-zero: o provador honesto só revela um bit de informação: se ele sabe ou não o logaritmo discreto de X.

Portanto, ele pode ser classificado como um algoritmo de conhecimento zero.

2.5 CURVAS ELÍPTICAS

Curvas elípticas já vem sendo estudadas durante centenas de anos e possuem uma ampla gama de aplicações. Sua aplicação em criptografia de chave pública foi proposta recentemente, apenas em 1985.

2.5.1 *ECDSA*

O *ECDSA* (do inglês, *Elliptic Curve Digital Signature Algorithm*) é uma variante do *DSA* (do inglês, *Digital Signature Algorithm*), que se utiliza da dificuldade de resolver o problema do logaritmo discreto no grupo finitos das curvas elípticas.

Tal algoritmo engloba a geração e verificação da assinatura digital de um documento.

Antes do algoritmo ser executado, deve-se acordar um protocolo de comunicação que engloba os parâmetros $(\mathbb{Z}_p, \mathbb{C}, G, n)$, onde $\mathbb{Z}_p$ denota o corpo onde os parâmetros da curva estão definidos, $\mathbb{C}$ denota os parâmetros da curva, G um gerador do corpo e n é um primo, a ordem do gerador.

2.5.1.1 ALGORITMO DE GERAÇÃO DE ASSINATURA

Para assinar uma mensagem m, utilizando uma chave privada d, deve-se:

1. Calcular z, os $b(n)$ primeiros bits de um *hash* da mensagem m.
2. Tomar um número aleatório no intervalo $[1, n-1]$.
3. Calcular $(x, y) = kG$.
4. Calcular $r = x \pmod{n}$. Caso $r = 0$, retornar ao passo 2..
5. Calcular $s = (z + rd)k^{-1} \pmod{n}$. Caso $s = 0$, retornar ao passo 2..
6. A assinatura é dada por (r, s).

2.5.1.2 ALGORITMO DE VERIFICAÇÃO DE ASSINATURA

Para verificar a assinatura (r, s) de uma mensagem m, utilizando uma chave pública $Q = kG$, deve-se:

1. Verificar se $r, s \in [1, n-1]$.
2. Calcular z, os $b(n)$ primeiros bits de um *hash* da mensagem m.
3. Calcular $w = s^{-1} \pmod{n}$.
4. Calcular $u_1 = zw \pmod{n}$ e $u_2 = rw \pmod{n}$.
5. Calcular $(x, y) = u_1G + u_2Q$.
6. A assinatura é válida se $r \equiv x \pmod{n}$.

2.5.2 PROVA DA CORRETUDE DO ALGORITMO

Para provar a corretude do algoritmo, deve-se verificar se:

$$u_1G + u_2Q = kG$$

Para isso, note que:

$$u_1G + u_2Q = zwG + rwdG = (z + rd)wG = k\frac{(z + rd)}{(z + rd)}G = kG$$

2.5.3 ATAQUES CONHECIDOS

Um dos ataques mais famosos acontece quando k se repete em duas assinaturas. Supondo que k seja igual, haverá duas assinaturas (r, s) e (r, s') para os *hashes* z e z'. Então, pode-se escrever:

$$s - s' = (z + rd)k^{-1} - (z' + rd)k^{-1} \pmod{n} = (z - z')k^{-1}$$

$$\Rightarrow k = (z - z')(s - s')-1$$

Só que:

$$s = (z + rd)k^{-1} \Rightarrow d = (sk - z)r^{-1}$$

$$\Rightarrow d = \left(s\frac{(z - z')}{(s - s')} - z\right)r^{-1}$$

Portanto, é possível obter a chave privada de uma parte que assinou duas mensagens distintas com o mesmo valor do parâmetro k.

2.6 ASSINATURAS EM ANEL

Assinaturas em anel permitem que um usuário assine uma mensagem de maneira que um grupo (ou anel) de usuários seja identificado pela assinatura, de maneira que não seja revelado qual o membro do anel foi o responsável por realizar a assinatura. Tais esquemas de assinatura são extremamente flexíveis, e permitem a formação de anéis *ad-hoc* que sejam capaz de regular o nível de anonimidade que o signatário deseja ter.

Elas se diferem das assinaturas de grupo, pois não há nenhuma entidade centralizada que possa agir como gerente do grupo, que seja capaz de identificar o usuário responsável pela assinatura e até mesmo revocar a permissão de assinar.

2.6.1 DEFINIÇÃO

Formalmente, um sistema de assinatura em anel pode ser visto como uma tripla de funções $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, que gera as chaves para um usuário, assina a mensagem e verifica as assinaturas, respectivamente. Seja $R = (P_1, \ldots, P_n)$ uma lista de chaves públicas, a qual chama-se de anel. Admite-se que não existam chaves públicas duplicadas no anel. Pode-se descrever a tripla de funções como:

1. $\mathcal{G}()$ gera uma chave pública P e uma chave privada p.
2. $\mathcal{S}(m, R, p)$ gera a assinatura σ da mensagem m usando o anel R e a chave privada p. Admite-se que a chave pública correspondente a p está contida em R.
3. $\mathcal{V}(m, \sigma, R)$ verifica se a assinatura σ foi gerada por algum $P \in R$ a partir da mensagem m.

2.6.2 ALGORITMO

O esquema de assinatura em anel pode ser adaptado para não permitir que mais de uma assinatura seja gerada pela mesma chave privada. Tal adaptação é descrita em van Saberhagen (2014) e utiliza uma curva elíptica com gerador G e e ordem q. Também são utilizadas duas funções de *hash* $\mathcal{H}_s$ e $\mathcal{H}_p$, onde a primeira retorna um inteiro e a segunda retorna um ponto da curva elíptica.

A seguir, encontra-se a descrição do algoritmo:

1. $\mathcal{G}()$ gera p, um número aleatório em $\mathbb{Z}_q - 0$, calcula $P = pG$ e retorna o par (p, P).
2. $\mathcal{S}$ recebe uma mensagem m, um anel $R = (P_1, \ldots, P_n)$ contendo n chave públicas e uma chave privada p correspondente a chave pública $P_s \in R$. Para calcular a assinatura σ, gera-se $\{q_i | i \in 1, \ldots, n\}$ e $\{w_i | i \in 1, \ldots, n\}$, números aleatórios em $\mathbb{Z}_q - 0$. Calcula-se a imagem da assinatura I que é dada por $I = p\mathcal{H}_p(P)$. Com tais valores, as seguintes transformações são aplicadas:

$$L_i = \begin{cases} q_s G \pmod q, & \text{caso } i = s. \\ q_i G + w_i P_i \pmod q, & \text{caso contrário.} \end{cases}$$

$$R_i = \begin{cases} q_s H_p(P) \pmod q, & \text{caso } i = s. \\ q_i H_p(P_i) + w_i I \pmod q, & \text{caso contrário.} \end{cases}$$

Com isso calcula-se:

$$c = H_s(m, L_1, \ldots, L_n, R_1, \ldots, R_n)$$

E com isso obtém-se a resposta do signatário:

$$c_i = \begin{cases} c - \sum_{j \neq s} w_i \pmod q, & \text{caso } i = s. \\ w_i \pmod q, & \text{caso contrário.} \end{cases}$$

$$r_i = \begin{cases} q_s - c_s p \pmod q, & \text{caso } i = s. \\ q_i \pmod q, & \text{caso contrário.} \end{cases}$$

Que é dada por $\sigma = (I, c_1, \ldots, c_n, r_1, \ldots, r_n)$.

3. Para se checar se a assinatura está correta, a função $\mathcal{V}$ recebe a assinatura $\sigma = (I, c_1, \ldots, c_n, r_1, \ldots, r_n)$, a mensagem m e o anel R. Confere-se se I não foi utilizado em nenhuma transação publicada anteriormente e calcula-se:

$$\begin{cases} L'_i = r_i G + c_i P_i & \pmod q \\ R'_i = r_i H_p(P_i) + c_i I & \pmod q \end{cases}$$

Checa-se se $\sum_{i=1}^{n} c_i \stackrel{?}{=} H_s(I, L'_1, \ldots, L'_n, R'_1, \ldots, R'_n)$. Caso a igualdade não seja válida ou I já tenha sido utilizada em transações anteriores, a assinatura não é aceita. Caso contrário, aceita-se a assinatura.

2.6.3 PROVA DA CORRETUDE DO ALGORITMO

Veja que a condição de verificação é $\sum_{i=1}^{n} c_i \stackrel{?}{=} H_s(m, L'_1, \ldots, L'_n, R'_1, \ldots, R'_n)$. Supondo $L'_i = L_i$ e $R'_i = R_i$, checar tal igualdade é equivalente a $\sum_{i=1}^{n} c_i \stackrel{?}{=} H_s(m, L_1, \ldots, L_n, R_1, \ldots, R_n) = c$. Só que:

$$\sum_{i=1}^{n} c_i = c \Leftrightarrow c_s = c - \sum_{i \neq s}^{n} c_i = c - \sum_{i \neq s}^{n} w_i$$

Que é a definição de c_s. Portanto, tal condição sempre vale. Falta mostrar que $L'_i = L_i$ e $R'_i = R_i$.

Note que a definição de L'_i e de R'_i são as mesmas de L_i e de R_i para todo $i \neq s$. Logo, se a assinatura σ estiver correta, tais valores serão idênticos. Para mostrar que $L'_s = L_s$ e que $R'_s = R_s$ note que:

$$L'_s = r_s G + c_s P = (q_s - c_s p)G + c_s pG = q_s G = L_s$$
$$R'_s = (q_s - c_s p)H_p(P) + c_s I = q_s H_p(P) - c_s pH_p(P) + c_s I = q_s H_p(P) = R_s$$

O que completa a demonstração.

2.6.4 CONSIDERAÇÕES DE SEGURANÇA E PERFORMANCE

Como foi proposta originalmente em (RIVEST et al., 2001), a assinatura em anel tinha o objetivo de conseguir a maior privacidade possível. Como o esquema de assinatura em anel permite a qualquer usuário verificar que um dos donos de alguma das chaves públicas contidas no anel foi o responsável pela assinatura, mas não qual, temos que quanto maior o tamanho do anel, menos exposta fica a identidade do signatário. Ou seja, no contexto original quanto maior o tamanho da assinatura, melhor.

No contexto desse trabalho há vantagens e desvantagens a serem consideradas. A assinatura em anel faz com que a origem dos pagamentos não seja rastreável, pois cada pagamento não inclui sua origem e sim um anel de possíveis origens. Quanto maior o anel, mais difícil é de ser rastrear a origem ou de fazer alguma análise da topologia da rede. Em contra partida, a quantidade de informação necessária para se armazenar na rede por transação também aumenta com o tamanho do anel. Como as transações são armazenadas em um banco de dados público, o aumento no tamanho das transações pode ser bastante custoso, portanto há um preço para se obter níveis maiores de privacidade.

Ao se escolher o tamanho do anel deve-se então levar em conta o nível de privacidade desejado e as restrições de memória. Como este trabalho foca apenas nos aspectos criptográficos do sistema, a memória não foi levada em consideração na escolha do tamanho dos anéis. Julgou-se que 10 elementos por anel seria uma quantidade suficiente para impedir qualquer tentativa de rastreamento nas transações.

3 ESQUEMA CRIPTOGRÁFICO PROPOSTO

O esquema criptográfico proposto para o projeto será especificado durante este capítulo. Tal esquema é baseado no trabalho de van Saberhagen (2014) e foi projetado de maneira de que usuários possam fazer transações de ativos arbitrários em uma rede pública. Esse trabalho se difere de van Saberhagen (2014) pois não limita a rede a transações de apenas um ativo. Suas vantagens sobre o trabalho de Nakamoto (2008) são as herdadas de van Saberhagen (2014), que conferem a rede a capacidade de ofuscar a origem dos ativos e o destinatário de cada transação, impedindo que informações das movimentações dos usuários sejam expostas na rede.

O esquema utiliza assinaturas ECDSA, mas ao contrário de atribuir a cada usuário um par de chaves a, $P = aG$, propõe-se que a cada identidade seja relacionado um par de chaves privadas (a, b) e um par de chaves públicas $(A, B) = (aG, bG)$, onde G é o gerador da curva escolhida.

3.1 ESQUEMA DE TRANSAÇÃO

A descrição do esquema será feita com base nas ações de dois agentes, Alice e Bob, que representam usuários do sistema. Caso Alice deseje enviar algum ativo a Bob, ela deve criar uma transação. Uma transação envia ativos a pelo menos, mas não se limitando, a um destinatário. Cada um desses envios caracteriza uma saída da transação, ou *TXO* (do inglês *Transaction Output*).

Para criar a transação, Alice define T, o tipo do ativo que será transacionado e gera um número aleatório r, contido em $\mathbb{Z}_q$, corpo dos inteiros módulo q, onde q é o inteiro que representa a ordem da curva elíptica escolhida. Tal número deve ser mantido em sigilo, pois é a chave privada da transação. A partir de r, Alice gera $R = rG$, chave pública da transação. Para cada destinatário com chave pública (A, B), Alice define a quantidade Q do ativo T que tal destinatário receberá e calcula $P = \mathcal{H}_p(rA) + B$ endereço público desse *TXO*. Um esquema da transação pode ser visto na figura 3.1.

Como o endereço da transação não é determinístico, Bob não sabe a priori onde buscar ativos que lhe foram enviados. Porém dada a chave pública da transação, Bob é capaz de usar sua chave privada para checar se a transação foi destinada a ele. Para isso ele calcula $P' = H_s(Ra) + B$. Caso $P = P'$, Bob sabe que a transação foi destinada a ele. Portanto,

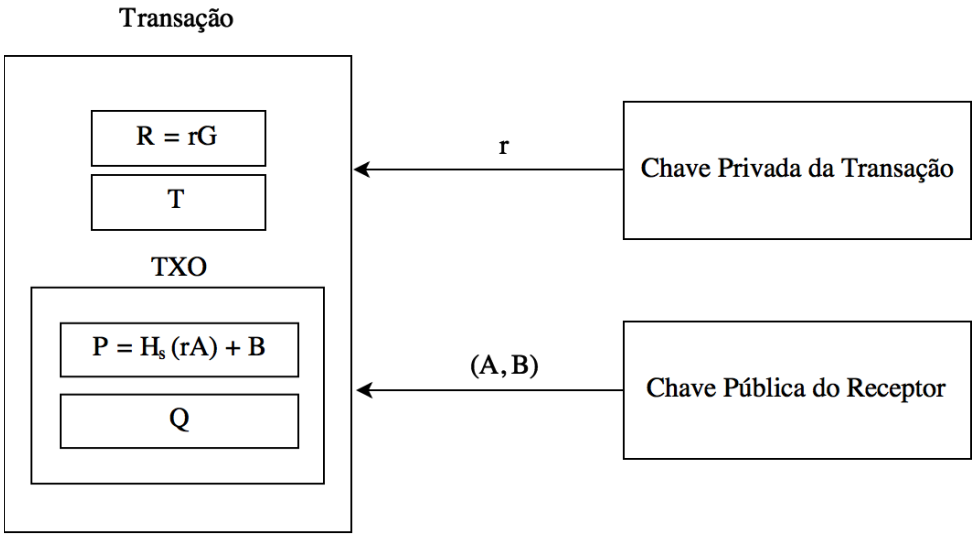

FIG. 3.1: Esquema criptográfico da transação

Bob deve iterar em todas as TXOs não vistas no sistema e checar se alguma delas lhe foi destinada. Bob também pode calcular a chave privada do TXO, que é dada por:

$$p = H_s(Ra) + b$$

Tal valor é de fato a chave privada pois $pG = H_s(Ra)G + bG = H_s(rA)G + B = P$.

Bob não precisa usar toda sua chave privada para encontrar quais TXOs lhe foram destinados. Ele apenas precisa do par (a, B) para calcular P'. Entretanto, para gerar o chave privada dos TXOs - e ser capaz de utilizar os recursos contidos nesse TXO - Bob precisa de (a, b). Caso Bob não tenha recursos computacionais para checar quais TXOs lhe foram direcionados, ele pode ceder seu par (a, B) para um terceiro que tenha e que pode agregar as transações relevantes para Bob, sem ser capaz de gastar os ativos que ele recebe.

Para Alice criar a transação, ela precisa fornecer como entradas da chave privada do endereço dos ativos que ela enviou a Bob. Com tal chave, Alice pode assinar a transação, garantindo que ela tem direito a transacionar seus ativos. Para diminuir a quantidade de informações expostas nas transações da rede, usa-se o esquema de assinatura em anel descrito na seção 2.6 para assinar as transações.

Para que a assinatura em anel funcione, Alice deve encontrar outras transações na rede que utilizem o mesmo ativo que ela está enviando e que enviem a mesma quantidade do ativo. Caso não seja possível encontrar as mesmas quantidades, pode-se quebrar as transações em várias menores, utilizando quantidades já transacionadas na rede.

3.2 ESQUEMAS DE CONFIRMAÇÃO

3.2.1 CONFIRMAÇÃO DE TRANSAÇÃO

Para que Alice confirme a Bob que foi ela quem criou a transação que Bob recebeu, Alice deve mostrar que sabe r. Só que Alice não deseja revelar r a Bob, pois se alguém além de Alice tiver o conhecimento de r, não faz sentido Alice utilizar tal valor para confirmar a origem da transação.

Para que Alice mostre que sabe o valor r sem revelar de fato tal valor, utiliza-se um esquema de prova de conhecimento zero. O esquema descrito na seção 2.4.1 pode ser adaptado para curvas elípticas de maneira a não depender de iteratividade, como exposto em Chatzigiannakis (2011). Tal esquema segue o seguinte algoritmo:

1. Alice gera um inteiro aleatório $k \in \mathbb{Z}_q$.
2. Alice calcula $c = \mathcal{H}_s(rA, kA, kG)$, onde A é a primeira metade da chave pública de Alice.
3. Alice calcula $s = k + cr \pmod{q}$.
4. Alice envia a mensagem (s, rA, kA, kG) a Bob.
5. Bob calcula $c = \mathcal{H}_s(rA, kA, kG)$.
6. Bob obtm os valores de A da chave pública de Alice e R da transação.
7. Bob checa se $sG = kG + cR$.
8. Bob checa se $sA = kA + crA$.

Se as checagens de Bob forem positivas, Bob aceita que Alice sabe r e portanto foi ela quem enviou a transação. Como argumentado em (CHATZIGIANNAKIS, 2011), para Alice enganar Bob, ela teria que achar s tal que $sG = kG + cR$, o que em si é uma instância do problema do logaritmo discreto.

Além disso, Bob não é capaz de tomar tal mensagem para assumir a identidade de Alice, dado que um outro verificador usaria a chave pública de Bob para fazer a última checagem. Portanto tal esquema é correto, integro e de zero conhecimento, pois não deixa vazar informações sobre r.

3.2.2 CONFIRMAÇÃO DE POSSE

Para que Alice confirme a Bob que ela possui determinado *TXO*, Alice deve mostrar que sabe p e que $I = p\mathcal{H}_p(P)$ não é uma imagem que já foi utilizada. Só que Alice não deseja revelar p a Bob, pois se alguém além de Alice tiver o conhecimento de p, então essa pessoa pode utilizar o *TXO* como se fosse Alice.

Alice e Bob então utilizarão um esquema criptográfico similar ao criado para confirmar transações.

1. Alice gera um inteiro aleatório $k \in \mathbb{Z}_q$.
2. Alice calcula $c = \mathcal{H}_s(I, k\mathcal{H}_p(P), kG)$.
3. Alice calcula $s = k + cp \pmod{q}$.
4. Alice envia a mensagem $(s, I, k\mathcal{H}_p(P), kG)$ a Bob.
5. Bob calcula $c = \mathcal{H}_s(I, k\mathcal{H}_p(P), kG)$.
6. Bob obtém o P do *TXO*.
7. Bob checa se $sG = kG + cP$.
8. Bob checa se $s\mathcal{H}_p(P) = k\mathcal{H}_p(P) + cI$.
9. Bob checa se a imagem I já foi utilizada em alguma transação.

Se as checagens de Bob forem positivas, Bob aceita que Alice sabe p, que a imagem $I = p\mathcal{H}_p(P)$ não foi utilizada e, portanto, que Alice possui o ativo contido no *TXO* de chave pública P.

Ao contrário do esquema anterior, o esquema de confirmação de posse não é deconhecimento zero. Ele é correto e íntegro, mas faz com que Bob saiba o valor de I. Portanto Bob saberá que Alice ainda possui o ativo enquanto ela não fizer uma transação contendo o *TXO* correspondente.

Uma solução alternativa envolveria Alice enviar a Bob um múltiplo de I, cI por exemplo, mas para que Bob checasse que Alice de fato não gastou tal *TXO* ele teria que checar todas as imagens de transações já realizadas, pegando suas imagens e multiplicando por c. Tal abordagem, apesar de ser de zero conhecimento torna a complexidade computacional do esquema grande demais.

3.3 VANTAGENS

Diferentemente do proposto por Nakamoto (2008), um endereço não é caracterizado pela chave pública de um usuário da rede. O endereço de destino de cada transação é único e é gerado a partir da chave da transação e da chave pública do destinatário. Dessa maneira não é possível identificar o receptor à partir do destino da transação, o que aumenta o anonimato na rede e diminui a possibilidade de uma análise dos dados.

Utilizando as assinaturas em anéis ainda é possível evitar que um ativo seja duplicado por um usuário malicioso, enquanto mantém-se em sigilo se qualquer ativo da rede foi trocado ou não. Por exemplo, se Alice envia x ações da empresa y para Bob através de uma transação T, Alice não tem como saber se Bob ainda tem ou não tais ações, pois, como assinaturas em anel são utilizadas para realizar as transações, mesmo que a chave pública da transação T apareça nos anéis de alguma transação, não há como Alice saber se tais ações estão de fato contidas em T.

Em Noether (2014a), Noether (2014b) e em van Saberhagen (2014) pode-se encontrar análises da segurança do esquema proposto por (VAN SABERHAGEN, 2014). O sistema implementado nesse trabalho é uma modificação do proposto por (VAN SABERHAGEN, 2014), onde existem vários tipos de ativos e em que a possibilidade de se fazer provas de conhecimento zero sobre posse de ativos e execução de transações. Portanto, espera-se que o sistema desse trabalho seja tão robusto quanto o proposto em (VAN SABERHAGEN, 2014). Em especial, vale ressaltar a segurança de tal esquema é dependente do problema do logaritmo discreto no corpo das curvas elípticas. Tal problema já foi amplamente estudado, o que corrobora com a ideia de ser um esquema seguro de criptografia.

3.4 LIMITAÇÕES

Tal esquema não permite esconder a informação sobre o histórico de transações de ativos únicos (carros, títulos de terrenos e imóveis, por exemplo). Tal limitação existe devido à utilização da assinatura em anel no esquema proposto. Tal assinatura esconde qual ativo está sendo transferido, expondo uma lista de ativos idênticos na qual tal transação pode ser aplicada. É garantido que quem realiza a transação tem acesso a um dos ativos, mas não sabe-se qual ativo em particular foi enviado.

No caso de ativos únicos, não é possível utilizar tal esquema pois não existem outros ativos equivalentes para se realizar tal mascaramento. Portanto, quando o ativo é único, a rede vaza informações sobre seu histórico de transações. Apesar disso ser negativo (o dono de um imóvel pode querer manter em sigilo o número de vezes que tal imóvel foi

vendido nos últimos anos), tal deficiência não é crítica o suficiente para abrir mão dos outros benefícios do esquema proposto.

Idealmente, para evitar que qualquer tipo de informação na rede, deve-se evitar que o tipo do ativo e a quantidade dos ativos transacionados apareça nas transações, mas que se garanta que a soma das saídas e das entradas de cada transação seja sempre nula. O esquema proposto por (POELSTRA, 2014) possui tais propriedades, mas sua compatibilidade com o sistema projetado nesse trabalho não chegou a ser estudada e pode ser proposto como trabalho futuro.

(NOETHER, 2016) sugere as *Ring Confidential Transactions*, assinaturas em anel que permitem que os valores transacionados sejam ofuscados. Não foi analisado a compatibilidade de tais resultados com múltiplos ativos, mas caso seja possível adaptar as *Ring Confidential Transactions* para trabalhar com múltiplos ativos, podem existir ganhos significativos na privacidade fornecida pela rede.

Na operação de confirmação de posse ainda há o vazamento de informação sobre a imagem do *TXO*. Tal vazamento é indesejável, pois quando-se confirma posse deseja-se fazê-lo somente imediatamente, não dando a quem recebeu a confirmação um mecanismo para saber por quanto tempo o dono ainda possuirá o ativo.

4 SISTEMA

Em relação as suas capacidades, o sistema é simples. Como foi descrito nos objetivos, o sistema deve ser capaz de:

- Criar e publicar transações de ativos ou bens em um banco de dados público, através de contratos.
- Emitir provas de identidade que confirmem a posse do ativo, sem que tais provas emitam informações sobre o proprietário.
- Emitir provas de que confirmem uma transação efetuada, sem que tais provas emitam informações sobre o proprietário.

As provas devem ser de conhecimento-zero para que nenhuma informações dos proprietários sejam emitidas durantes as provas.

Para dar suporte a tais objetivos, o sistema também deve ser capaz de:

- Criar um tipo de ativo no banco de dados público, que será representado por um identificador único, e emitir a quantidade de tal ativo que estará disponível para troca.
- Associar o identificador do ativo criado e armazenado no banco de dados com um ativo real.

Note que tais ativos e transações serão registrados em um banco de dados público onde admite-se que não seja possível modificar registros anteriores, somente adicionar novas transações. Para evitar que os usuários possam ter informações privadas divulgadas é necessário que tais transações não possam ser ligadas a pessoa que as realizou. Ou seja, qualquer análise dos dados disponíveis não pode ser capaz extrair informações de usuários da rede.

Por exemplo, análises não podem ser capaz de ligar um usuário a um ativo, ou até mesmo constatar que dois ativo pertencem a um mesmo usuário, mesmo sem ser capaz de identificar quem seja tal usuário. Uma outra necessidade é de impedir que informações sobre as transações de um ativo sejam reveladas, não deve ser possível identificar por quantas transações um ativo passou ou quais foram as datas de tais transações.

Como o foco do trabalho está no modelo criptográfico proposto, supõe-se que existe um identificador físico para cada ativo. Tal hipótese não é difícil de ser realizada dependendo o tipo de ativo que se deseja registrar. No exemplo dos imóveis, pode-se usar o endereço do imóvel como o identificador físico e é razoável considerar que o *SHA-256* de tal identificador pode ser considerado o identificador único do banco de dados. Um segundo exemplo de ativo que pode ser transacionado na rede são ações. Ações podem ser unicamente identificadas e consistem em um grande número de ativos idênticos.

4.1 CASOS DE USO

Os casos de uso do sistema são demonstrados na figura 4.1.

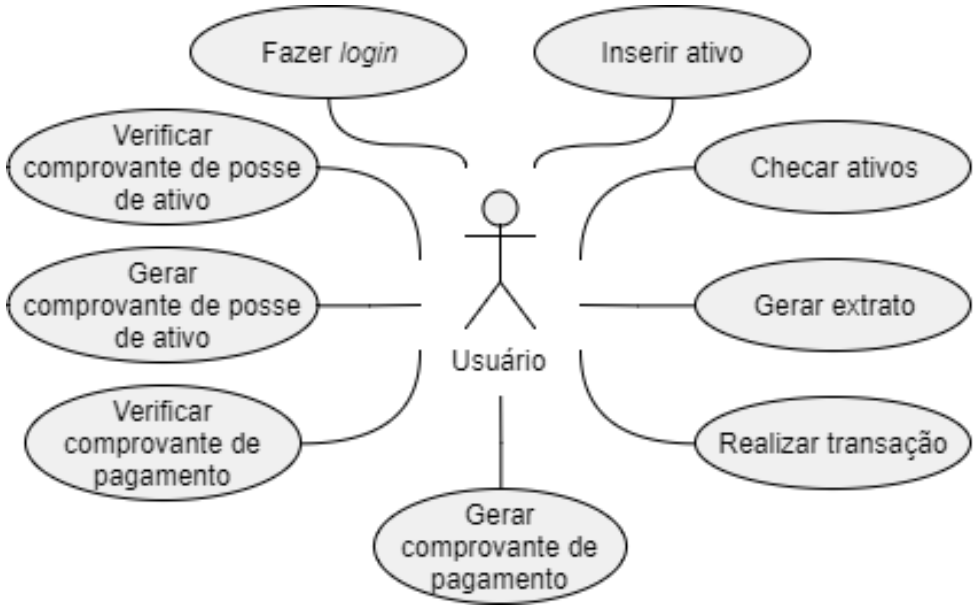

FIG. 4.1: Casos de uso

Ao fazer *login*, o usuário confirma suas informações privadas. Essa ação é necessária para evitar que somente o usuário possa ter acesso à suas informações privadas e que nenhum usuário malicioso que tenha acesso ao aparelho do usuário com acesso ao sistema faça transações que o usuário não queira.

Com acesso ao sistema o usuário pode:

- Inserir ativos: inserindo um novo ativo no sistema
- Checar ativos: listando todos os ativos que o usuário possui no momento
- Gerar extrato: listando todas as transações realizadas pelo usuário

- Realizar transação: criando uma transação de algum ativo que o usuário possui no momento e transferindo para outro usuário
- Gerar comprovante de pagamento: gera um comprovante para confirmar que a transação foi efetuada pelo usuário
- Verificar comprovante de pagamento: verifica se o comprovante de pagamento é válido
- Gerar comprovante de posse de ativo: gera um comprovante para confirmar que o ativo pertence ao usuário
- Verificar comprovante de posse de ativo: verifica se o comprovante de posse do ativo é válido

Os casos que contém comprovante são necessários devido às propriedades criptográficas do sistema, pois não é possível saber que usuário realizou as transações.

4.2 DIAGRAMA DE CLASSES

A modelagem do sistema é demonstrada na figura 4.2.

O tipo de dado *Point* é dependência externa do sistema, sendo um ponto geométrico pertencente à curva elíptica. O tipo de dado *JSON*, *JavaScript Object Notation*, é a notação utilizada no sistema para representar os dados independente da plataforma e linguagem, sendo usado para serializar os objetos das classes e armazenar no banco de dados.

As classes *User*, *PrivateKey* e *PublicKey* representam o usuário, a chave privada e a chave pública do mesmo, respectivamente.

A classe *Transaction* contém as informações de uma transação, sendo R seu identificador único, T o identificador do ativo que a transação está transacionando, *inputs* o conjunto de assinaturas de anel que contém as saídas de transações dos ativos sendo transferidos e *outputs* o conjunto de saídas de transações que são resultado da transação.

A classe *TXO* representa as saídas de transação, como explicado no capítulo 3, contendo P e *amount*, sendo o primeiro o endereço de destino da saída de transação e o segundo a quantidade transferida.

A classe *Signature* representa uma assinatura em anel, sendo *Ring* uma classe auxiliar que armazenar as informações do anel.

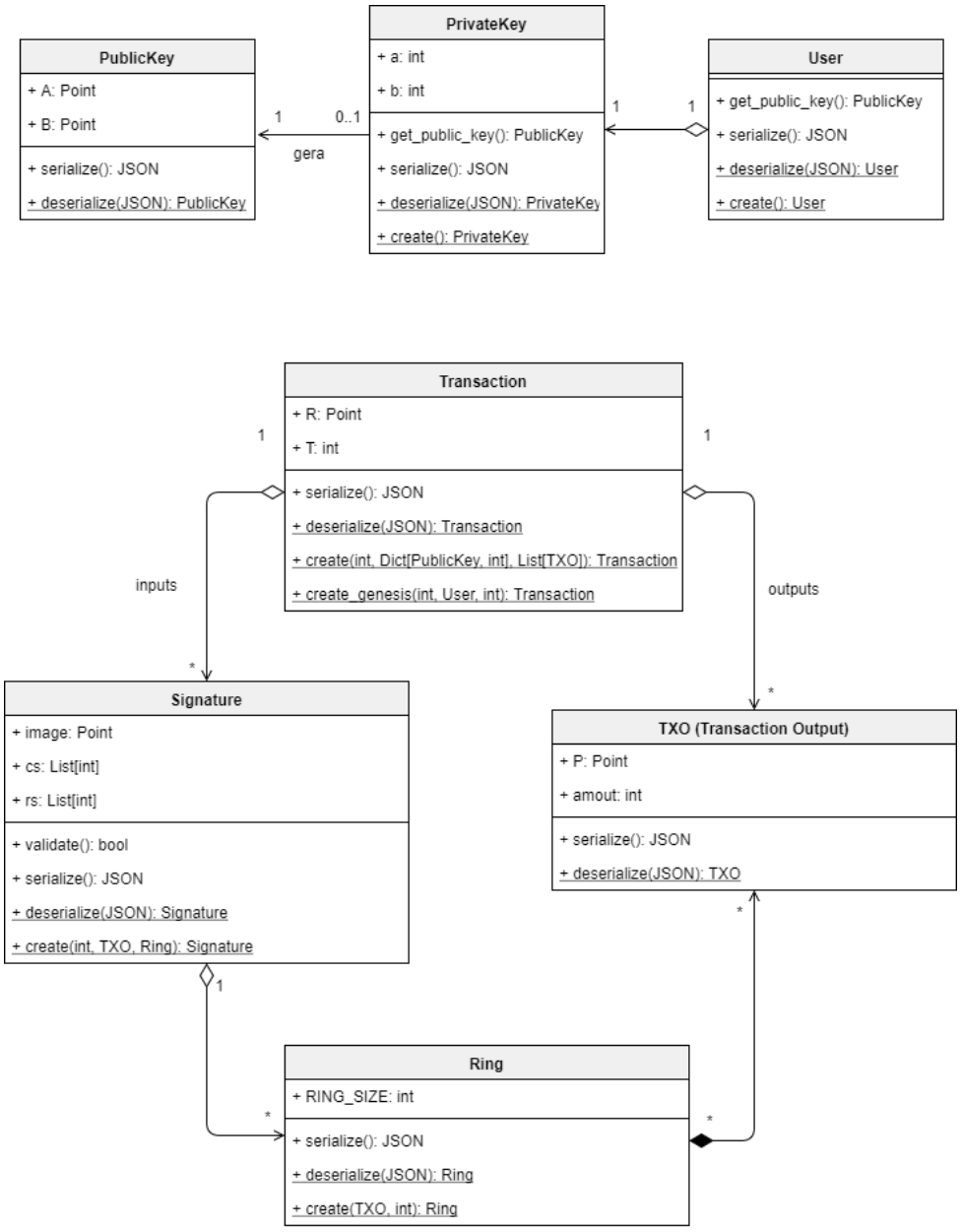

FIG. 4.2: Diagrama de classes

Todas as classes contém métodos de serialização e desserialização para que possam ser armazenadas e recuperadas do banco de dados, pois os objetos contém números grandes que não costumam ser facilmente armazenados em bancos de dados além de pontos em curvas elípticas.

4.3 IMPLEMENTAÇÃO DO SISTEMA

O sistema foi implementado na linguagem de programação *Python* por ser uma linguagem com vasto uso na área de criptografia e segurança da informação, facilitando a busca por

bibliotecas confiáveis que implementassem criptografia com curvas elípticas.

A biblioteca utilizada para criptografia com curvas elípticas foi o pacote *fastecdsa* (KUELTZ, 2017), que implementa geração de chaves públicas e privadas além de aritmética em curvas elípticas, que foram necessárias para a implementação da assinatura em anel do sistema.

Para testar as operações no sistema se utilizou o pacote *pytest* (HOLGER KREKEL, 2004-2017) e foi implementado testes unitários para garantir que os métodos implementados estavam funcionando como esperado.

Além disso, foi utilizado o banco de dados *MongoDB* (MONGODB INC., 2009-2017) para simular o banco de dados público que poderia ser utilizado em uma aplicação real do sistema.

O uso da *API* é apresentado no apêndice 7.1.

5 CONCLUSÃO

O trabalho teve como objetivo projetar e desenvolver um sistema criptográfico que possibilite o armazenamento de informações sobre posse de ativos em um banco de dados público. Tal sistema também deveria possibilitar aos seus usuários a capacidade de transferir os ativos e emitir provas de conhecimento zero sobre a execução de transferências e sobre a posse de um determinado ativo.

Após um estudo da literatura disponível sobre provas de conhecimento zero e sistemas de *criptomoedas* e *blockchains*, foi possível conhecer as várias tecnologias que possibilitam a criação de tal sistema, suas vantagens e limitações. Decidiu-se usar o trabalho de van Saberhagen (2014) como base para desenvolver o esquema criptográfico pois ele utiliza técnicas criptográficas para manter privacidade e a anonimidade dos usuários.

Como o foco do trabalho é o sistema criptográfico, a implementação foi feita usando como base um banco de dados local, sem levar em conta os problemas de concorrência e consistência que devem ser levados em conta no projeto de um sistema distribuído do tipo *peer to peer*.

Dos objetivos iniciais apenas o de emissão de prova de conhecimento zero sobre posse de ativos precisou ser relaxado, todos os demais foram incluídos no projeto e na implementação do sistema. O projeto permitiu um aprendizado amplo na área de criptografia e traz consigo a possibilidade de trabalhos futuros relacionados, tanto em relação a melhorias na área criptográfica - possibilitar o uso de criptografia em anel mesmo em transações de quantidades e ativos distintos -, quanto em relação a tornar o sistema distribuído por meio de uma *blockchain* ou tecnologia similar.

6 REFERÊNCIAS BIBLIOGRÁFICAS

CHATZIGIANNAKIS, IOANNIS AND PYRGELIS, A. P. Y. C. Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices. In: MOBILE ADHOC AND SENSOR SYSTEMS (MASS), 2011 IEEE 8TH INTERNATIONAL CONFERENCE ON, 1., 2011. **Anais**... [S.l.: s.n.], 2011, p. 715–720.

FORBES. The First Government To Secure Land Titles On The Bitcoin Blockchain Expands Project. Disponível em: <https://www.forbes.com/sites/laurashin/2017/02/07/the-first-government-to-secure-land-titles-on-the-bitcoin-blockchain-expands-project/>. Acesso em: 5 mai. de 2017.

GARY STONEBURNER, CLARK HAYDEN, A. F. **Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A**. [S.l.]: National Institute of Standards and Technology, 2004.

HAO, F. **Schnorr NIZK Proof: Non-interactive Zero Knowledge Proof for Discrete Logarithm**. [S.l.: s.n.], 2017. (Relatório Técnico, draft-hao-schnorr-06).

HOLGER KREKEL, ET AL. pytest. Disponível em: <https://docs.pytest.org>. Acesso em: 28 set. de 2017.

KOSBA, A.; MILLER, A.; SHI, E.; WEN, Z. ; PAPAMANTHOU, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: SECURITY AND PRIVACY (SP), 2016 IEEE SYMPOSIUM ON, 1., 2016. **Anais**... [S.l.: s.n.], 2016, p. 839–858.

ANTON KUELTZ. fastecdsa. Disponível em: <https://github.com/AntonKueltz/fastecdsa>. Acesso em: 28 set. de 2017.

MONGODB INC. MongoDB. Disponível em: <https://github.com/mongodb/mongo>. Acesso em: 28 set. de 2017.

NAKAMOTO, SATOSHI. Bitcoin: A peer-to-peer electronic cash system. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: 5 mai. de 2017.

NOETHER, SARANG NOETHER SURAE AND MACKENZIE, A. A note on chain reactions in traceability in cryptonote 2.0. **Research Bulletin MRL-0001. Monero Research Lab**, v. 1, p. 1–8, 2014.

NOETHER, SHEN AND MACKENZIE, A. Ring confidential transactions. **Ledger**, v. 1, p. 1–18, 2016.

NOETHER, SHEN AND NOETHER, S. Monero is not that mysterious. **Research Bulletin MRL-0003. Monero Research Lab**, v. 1, p. 1–10, 2014.

O GLOBO. Cartórios faturam R$ 1 bilhão por mês no Brasil. Disponível em: <https://oglobo.globo.com/brasil/cartorios-faturam-1-bilhao-por-mes-no-brasil-11337663>. Acesso em: 5 mai. de 2017.

ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT. **OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security**. [S.l.]: OECD Publishing, 2002.

POELSTRA, ANDREW AND BACK, ADAM AND FRIEDENBACH, MARK AND MAXWELL, GREGORY AND WUILLE, PIETER. Confidential Assets. Disponível em: <https://blockstream.com/bitcoin17-final41.pdf>. Acesso em: 15 jul. de 2017.

RIVEST, R.; SHAMIR, A. ; TAUMAN, Y. How to leak a secret. **Advances in Cryptology—ASIACRYPT 2001**, v. 0, p. 552–565, 2001.

SWAN, M. **Blockchain: Blueprint for a new economy**. [S.l.]: "O'Reilly Media, Inc.", 2015.

VAN SABERHAGEN, NICOLAS. Cryptonote v 2.0, 2013. Disponível em: <https://cryptonote.org/whitepaper.pdf>. Acesso em: 01 jul. de 2017.

7 APÊNDICES

APÊNDICE 1: USO DA *API* DO SISTEMA

A inicialização da *API*, criação de usuários e obtenção das as chaves públicas é demonstrado na figura 7.1.

```
In [1]: # Imports
        from cryptolib_api import *
        from uuid import uuid4

In [2]: # Clear DB to remove inconsistencies and crate users
        clear_data()

        user1 = create_user('user1', 'pass1')
        user2 = create_user('user2', 'pass2')
        user3 = create_user('user3', 'pass3')

        pk1 = user1.get_public_key()
        pk2 = user2.get_public_key()
        pk3 = user3.get_public_key()
```

FIG. 7.1: Inicialização, criação de usuários e obtenção das chaves públicas

A criação de um ativo é demonstrada na figura 7.2.

```
In [3]: # Define asset id and amount to be created
        asset_id = uuid4().int
        asset_amount = 1000

In [4]: # Attempt to make operations while not logged in
        create_asset(asset_id, asset_amount)

        User not logged in.
```

FIG. 7.2: Criação de ativo

O *login* e listagem dos ativos é demonstrado na figura 7.3.

A realização de transações é demonstrada na figura 7.4.

A checagem de que saída de transação pertence ao usuário atual é demonstrado na figura 7.5 e 7.7.

A verificação se é possível realizar um transação é verificada e caso não seja possível uma mensagem de erro é exibida, como demonstrado na figura 7.6.

Na figura 7.8 é demonstrado como se gera comprovantes das transação realizadas pelo usuário atual. Na figura 7.9 é demonstrado como se verifica se tais comprovantes são válidos.

```
In [5]: # Log in and check user assets
        login('user1', 'pass1')
        get_unspent_assets()

Out[5]: []

In [6]: # Repeat operations while logged in
        create_asset(asset_id, asset_amount)

        Asset created.

In [7]: # List all assets user have
        list_assets()

        Asset 22665361622589451140840687556011467500 2: 1000

In [8]: # Check user assets
        assets = get_unspent_assets()

        for asset in assets:
            print("Asset(ID : {}, Amount : {})".format(asset.T, asset.amount))

        Asset(ID : 22665361622589451140840687556011467500 2, Amount : 1000)
```

FIG. 7.3: *Login* e listagem de ativos

```
In [9]: # Make first transaction sending 50 units of the asset to user2 and 100 uni
        ts to user3
        transaction = make_transaction(asset_id, {pk2 : 200, pk3 : 300})

        Transfer successful.

In [10]: # Check if transaction subtracted user assets
         list_assets()

         Asset 22665361622589451140840687556011467500 2: 500
```

FIG. 7.4:

```
In [11]: # List TXO in transaction and check which ones bellongs to user1
         images = image_pool.get_list()
         text = " This TXO belongs to currently logged in user."

         for txo in transaction.outputs:
             print("Sent {} of asset {}.{}".format(txo.amount, txo.T, text if check_
         txo_ownership(txo, images) else ""))

         Sent 300 of asset 22665361622589451140840687556011467500 2.
         Sent 200 of asset 22665361622589451140840687556011467500 2.
         Sent 500 of asset 22665361622589451140840687556011467500 2. This TXO belongs
         to currently logged in user.
```

FIG. 7.5:

Na figura 7.10 e 7.11 é demonstrado como gerar comprovantes de posse de ativo e verificar se tais comprovantes são válidos.

A figura 7.12 demonstra o tamanho do anel gerado durante uma transação.

```
In [12]: # Try to transfer more than user have
         make_transaction(asset_id, {pk2 : 500, pk3 : 500})

         Not enough to transfer.

In [13]: # Checks user1 funds after transaction
         list_assets()

         Asset 22665361622589451140840687556011467500 2: 500

In [14]: # Transfer all funds to not existent account
         random_key = PrivateKey.create()
         lost_transaction = make_transaction(asset_id, {random_key.get_public_key()
         : 500})

         Transfer successful.

In [15]: # Checks that after user1 transfers all its funds it has no assets
         list_assets()

         No assets!
```

FIG. 7.6:

```
In [16]: # After transaction the user do not have ownership of the TXO anymore
         images = image_pool.get_list()
         text = " This TXO belongs to current logged in user."

         for txo in transaction.outputs:
             print("Sent {} of asset {}.{}".format(txo.amount, txo.T, text if check_
         txo_ownership(txo, images) else ""))

         Sent 300 of asset 22665361622589451140840687556011467500 2.
         Sent 200 of asset 22665361622589451140840687556011467500 2.
         Sent 500 of asset 22665361622589451140840687556011467500 2.
```

FIG. 7.7:

```
In [17]: # Storing TXOs in convinient way to check receipts
         def get_owner_pk(txo):
             for user in [user1, user2, user3]:
                 if check_txo_ownership(txo, [], owner = user):
                     return user.get_public_key()

         txos = {get_owner_pk(txo) : txo for txo in transaction.outputs}
         receipts = {pk : generate_asset_receipt(txo) for pk, txo in txos.items()}
```

FIG. 7.8:

As figuras 7.13, 7.14 e 7.15 mostram as estruturas internas das classes implementadas. As classes de assinatura, anel e transação possuem estruturas internas muito grandes, portanto não são apresentadas.

```
In [18]: # Verify that TXO that user1 received was in the transaction signed by user
         1
         login('user1', 'pass1')
         verify_asset_receipt(receipts[pk1], pk1, txos[pk1])

Out[18]: True

In [19]: # Verify that TXO that user2 received was in the transaction signed by user
         1
         login('user2', 'pass2')
         list_assets()
         verify_asset_receipt(receipts[pk2], pk1, txos[pk2])

         Asset 22665361622589451140840687556011467500 2: 200

Out[19]: True

In [20]: # Verify that TXO that user3 received was in the transaction signed by user
         1
         login('user3', 'pass3')
         list_assets()
         verify_asset_receipt(receipts[pk3], pk1, txos[pk3])

         Asset 22665361622589451140840687556011467500 2: 300

Out[20]: True
```

FIG. 7.9:

```
In [20]: # Verify that TXO that user3 received was in the transaction signed by user
         1
         login('user3', 'pass3')
         list_assets()
         verify_asset_receipt(receipts[pk3], pk1, txos[pk3])

         Asset 22665361622589451140840687556011467500 2: 300

Out[20]: True

In [21]: # Create a ownership receipt for the TXO received by user3
         receipt = generate_ownership_receipt(txos[pk3])

In [22]: # user1 verifies that receipt is indeed valid
         login('user1', 'pass1')
         verify_ownership_receipt(receipt)

Out[22]: True

In [23]: # user3 spends half of the TXO that create the receipt
         login('user3', 'pass3')
         new_transaction = make_transaction(asset_id, {pk2 : 50})

         Transfer successful.
```

FIG. 7.10:

```
In [24]: # Checks that ownership receipt is no longer valid
         login('user1', 'pass1')
         verify_ownership_receipt(receipt)

Out[24]: False
```

FIG. 7.11:

```
In [25]: # Create several transactions with same value

         login('user2', 'pass2')
         for i in range(3):
             make_transaction(asset_id, {pk1 : 10, pk3: 10})

         login('user3', 'pass3')
         for i in range(3):
             make_transaction(asset_id, {pk1 : 10, pk2: 10})

         Transfer successful.
         Transfer successful.
         Transfer successful.
         Transfer successful.
         Transfer successful.
         Transfer successful.

In [26]: login('user1', 'pass1')
         transaction = make_transaction(asset_id, {pk2 : 10})

         for txo in transaction.inputs:
             print("Ring size is {}.".format(len(txo.ring.txos)))

         Transfer successful.
         Ring size is 5.
```

FIG. 7.12:

```
In [27]: # PublicKey structure
         pk1.serialize()

Out[27]: {'A': {'x': '514765608328217036785547267605031052544969405328228199615264292
         258986341258936',
           'y': '8391384428375304573911128460162994189997083239088047360812800719125
         3436531510'},
          'B': {'x': '5800521627321569265964827024653153369763591491114767463153222 4
         370397481557725',
           'y': '4242185693484375466215968605892158831419203116045635530140103351126
         3399281180'}}
```

FIG. 7.13:

```
In [28]: # PublicKey structure
         user1.private_key.serialize()

Out[28]: {'a': '5659767697043019406276681938822065319917315439427578454567053729311 1
         782311440',
          'b': '3944379844778704985218448332063638364808671866195611741425875225359 6
         314985542'}
```

FIG. 7.14:

```
In [29]: # TXO structure
         transaction.outputs[0].serialize()
```

```
Out[29]: {'P': {'x': '916459794674123456031482434110005320351672420493460724676019084
         39456521108306',
           'y': '4026256701401989002147293041227068527464199705606631962567877780851
         3039084215'},
          'R': {'x': '1055632201285034331040283212875814324901954138418865903391704 2
         3580329634446951',
           'y': '9741424808861952612972432047984564194090921639738447048120207817953
         040530895'},
          'T': '226653616225894511408406875560114675002',
          'amount': '10',
          'public_key': {'A': {'x': '852901373363785735670382405466314428439029296700
         1141787739042056123134287 1574',
            'y': '8680574969174966698722683562159528680469396755326383192251801614 95
         67251686307'},
           'B': {'x': '2644839078581571418753692289254429400811804451316290370225791
         7876016644793822',
            'y': '2767908110045175464676046115122232032618944558471010968859488325 06
         20504229759'}}}
```

FIG. 7.15: