

Provas de Conhecimento Zero: Um Estudo sobre zk-SNARKs e sua Aplicação em Sistemas de Autenticação

Alex Davis Neuwiem da Silva

Ciências da Computação | INE | CTC



UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Introdução

1. Estudo Teórico

- ▶ Foi realizada uma análise aprofundada dos principais protocolos zk-SNARKs: foco em **Pinocchio** e **Groth16**.

2. Aplicação Prática

- ▶ Foi desenvolvido um sistema de autenticação biométrica facial que utiliza um dos protocolos estudados: **Groth16**.

O que é uma Prova de Conhecimento Zero?

É um protocolo computacional entre duas partes:

- ▶ **Prorador P :** A parte que alega conhecer uma informação.
- ▶ **Verificador V :** A parte que valida a alegação.

Satisfaz as seguintes propriedades:

- ▶ **Completeness:** Se P é honesto, V sempre aceitará a prova.
- ▶ **Solidez:** Se P é desonesto, V só aceitará a prova com uma probabilidade negligenciável.
- ▶ **Conhecimento Zero:** Garante que o verificador seja convencido sem revelar nenhuma informação adicional além da veracidade dessa afirmação.

Estrutura

zk-SNARKs

Groth16

Autenticação Biométrica

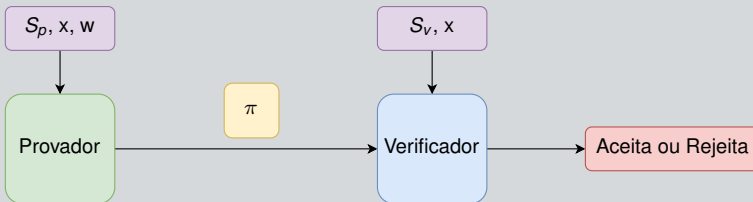
O que é um zk-SNARK?

Uma classe especial de Prova de Conhecimento Zero:

- ▶ **Succinct** (Sucinto): O tamanho da prova e tempo de verificação são constantes.
- ▶ **Non-interactive** (Não-interativo): O Provedor envia uma única mensagem para o Verificador, geralmente requer uma cerimônia inicial.
- ▶ **AR**gument of **K**nowledge (Argumento de Conhecimento): A solidez é *computacional*, baseada na dificuldade de certos problemas, e não estatística.

Processamento de uma prova

- ▶ Circuito $C(x, w) = 0$
- ▶ Pré-Processamento $S(C) \rightarrow (S_p, S_v)$



Estrutura

zk-SNARKs

Groth16

Autenticação Biométrica

Criptografia de Curvas Elípticas

Fundamentam o ambiente aritmético utilizado no protocolo:

- ▶ $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ são grupos de pontos de curva elíptica
- ▶ Emparelhamento $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
- ▶ g_1 e g_2 são geradores de \mathbb{G}_1 e \mathbb{G}_2
- ▶ $e(g_1, g_2)$ é o gerador de \mathbb{G}_T

Garantem propriedades:

- ▶ Logaritmo Discreto: Dado $s \cdot g_1$ é difícil encontrar s
- ▶ Coeficientes do emparelhamento podem ser reorganizados:
$$e(s_1 \cdot g_1, s_2 \cdot g_2) = e(g_1, g_2)^{(s_1 \cdot s_2)}$$
- ▶ Permite que o protocolo funcione para múltiplos Ps e Vs

Groth16: O Padrão de Eficiência

O Groth16 é uma otimização do protocolo Pinocchio, tornando-se o padrão da indústria.

Sua principal vantagem é a otimização:

- ▶ Reduz a prova a apenas **3 elementos**
- ▶ Reduz a verificação a apenas **3 emparelhamentos**

É dividido em 4 etapas:

- ▶ Aritmetização
- ▶ Cerimônia de confiança
- ▶ Geração da prova
- ▶ Verificação da prova

Aritmetização

Transforma a computação em uma representação matemática verificável:

$$\begin{array}{lcl} x^3 + x + 5 = y & \rightarrow & \begin{array}{l} v_1 = x * x \\ v_2 = v_1 * x \\ y = (v_2 + x + 5) * 1 \end{array}, \quad \underbrace{w = [1, y, x, v_1, v_2]}_{\text{Vetor testemunha}} \\ & & \underbrace{\hspace{10em}}_{\text{Restrições}} \end{array}$$

Cada restrição i é reescrita na combinação $w \cdot A_i \times w \cdot B_i = w \cdot C_i$:

$$\begin{array}{ccccc} w \cdot \begin{bmatrix} 0, 0, 1, 0, 0 \\ 0, 0, 0, 1, 0 \\ 5, 0, 1, 0, 1 \end{bmatrix} & \times & w \cdot \begin{bmatrix} 0, 0, 1, 0, 0 \\ 0, 0, 1, 0, 0 \\ 1, 0, 0, 0, 0 \end{bmatrix} & = & w \cdot \begin{bmatrix} 0, 0, 0, 1, 0 \\ 0, 0, 0, 0, 1 \\ 0, 1, 0, 0, 0 \end{bmatrix} \\ & & \begin{matrix} A & & B & & C \end{matrix} \end{array}$$

Aritmetização

As matrizes que representam a progressão da computação são transformadas em matrizes de polinômios:

$$\begin{array}{c} \begin{bmatrix} 1 & y & x & v_1 & v_2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\ C \end{array} \xrightarrow{\text{Interpolação dos pontos } (1, C_1) \dots (i, C_i)} \begin{array}{c} \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1.5 & 0.5 \\ 0 & 0 & 0 \\ 3 & -2.5 & 0.5 \\ -3 & 4 & -1 \end{bmatrix} \\ O(x) \end{array}$$

Analogamente para as matrizes A e B , resultando em $L(x)$ e $R(x)$, respectivamente.

Aritmetização

Os polinômios resultantes:

$L(x)$

$$\begin{bmatrix} 5 & -7.5 & 2.5 \\ 0 & 0 & 0 \\ 4 & -4 & 1 \\ -3 & 4 & -1 \\ 1 & -1.5 & 0.5 \end{bmatrix}$$

$R(x)$

$$\begin{bmatrix} 1 & -1.5 & 0.5 \\ 0 & 0 & 0 \\ 0 & 1.5 & -0.5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$O(x)$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -1.5 & 0.5 \\ 0 & 0 & 0 \\ 3 & -2.5 & 0.5 \\ -3 & 4 & -1 \end{bmatrix}$$

Obedecem à equação:

$$w \cdot L(x) \times w \cdot R(x) = w \cdot O(x)$$

Aritmetização

A computação original foi rescrita como uma equação polinomial:

$$P(x) = w \cdot L(x) \times w \cdot R(x) - w \cdot O(x) = H(x) \times T(x)$$

Teorema Fundamental da Álgebra: Se $P(x)$ possui raízes $\{1, \dots, n\}$, ele deve ser divisível por cada fator $(x - 1) \dots (x - n)$. Em nosso exemplo, temos:

$$T(x) = (x - 1)(x - 2)(x - 3)$$

Note que $L(x)$, $R(x)$, $O(x)$ e $T(x)$ são públicos e apenas um provador honesto pode gerar $P(x)$ e $H(x)$.

A Otimização Chave do Groth16

A equação é reestruturada usando os valores aleatórios α e β :

$$\begin{aligned} &(\alpha + w \cdot L(x)) \times (\beta + w \cdot R(x)) = \\ &\alpha \cdot \beta + \alpha \cdot w \cdot R(x) + \beta \cdot w \cdot L(x) + w \cdot O(\tau) + H(\tau) \cdot T(\tau) \end{aligned}$$

Verificação no ponto τ , pelo lema de Schwartz-Zippel:

$$\begin{aligned} &\underbrace{e((\alpha + w \cdot L(\tau)) \cdot g_1)}_A, \underbrace{(\beta + w \cdot R(\tau)) \cdot g_2}_B = \\ &\underbrace{e(\alpha \cdot g_1, \beta \cdot g_2)}_D \cdot \\ &\underbrace{e((\alpha \cdot w \cdot R(\tau) + \beta \cdot w \cdot L(\tau) + w \cdot O(\tau) + H(\tau) \cdot T(\tau)), g_2)}_C \end{aligned}$$

Cerimônia de Confiança

O Groth16 realiza uma cerimônia de confiança **específica do circuito** para gerar parâmetros públicos.

- ▶ Gera valores aleatórios secretos: $\alpha, \beta, \tau, \gamma, \delta$.
- ▶ Estes valores devem ser descartados. Se guardados, podem criar provas falsas.
- ▶ A cerimônia gera a Chave de Prova (S_p) e a Chave de Verificação (S_v).

Geração da Prova

Para garantir o **conhecimento zero**, o provador mascara os componentes da prova, gerando dois escalares aleatórios, r e s .

$$A = \alpha \cdot g_1 + \sum_{i=0}^{m-1} w_i \cdot L_i(\tau) \cdot g_1 + r \cdot \delta \cdot g_1$$

$$B' = \beta \cdot g_1 + \sum_{i=0}^{m-1} w_i \cdot R_i(\tau) \cdot g_1 + s \cdot \delta \cdot g_1$$

$$B = \beta \cdot g_2 + \sum_{i=0}^{m-1} w_i \cdot R_i(\tau) \cdot g_2 + s \cdot \delta \cdot g_2$$

$$C = \sum_{i=1}^{m-1} w_i \cdot \sigma_i + H(\tau) \cdot T(\tau) + s \cdot A + r \cdot B' - r \cdot s \cdot \delta \cdot g_1$$

Verificação da Prova

Primeiramente, o verificador utiliza a parte pública da testemunha e $\sigma_0, \dots, \sigma_{l-1}$ para calcular W :

$$W = \sum_{i=0}^{l-1} w_i \cdot \sigma_i$$

O verificador então insere a prova π , sua parte calculada W , e os elementos da chave de verificação na equação:

$$e(A, B) = e(\alpha \cdot g_1, \beta \cdot g_2) \cdot e(W, \gamma \cdot g_2) \cdot e(C, \delta \cdot g_2)$$

Estrutura

zk-SNARKs

Groth16

Autenticação Biométrica

Reconhecimento Facial com Similaridade de Cossenos

O reconhecimento facial pode ser modelado como uma tarefa de comparação entre vetores:

- ▶ Cada rosto é representado por um vetor de características (*embedding*)
- ▶ Vetores são gerados por redes neurais treinadas para extrair feições únicas

A comparação é feita utilizando a **similaridade de cossenos**.

O que é a Similaridade de Cossenos?

A **similaridade de cossenos** mede o ângulo entre dois vetores:

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|}$$

- ▶ Varia entre -1 (opostos) e 1 (iguais)
- ▶ Se $\cos(\theta) \approx 1$, vetores são semelhantes \rightarrow rostos parecidos

Autenticação com Similaridade de Cossenos

1. A imagem de entrada é convertida em vetor \vec{A}
2. Vetor \vec{B} é previamente armazenado durante a etapa de registro
3. Se $\cos(\vec{A}, \vec{B}) > \tau$, a autenticação é aceita

Nota: τ é um limiar definido com base no modelo de IA (ex: 0.7)

Problemas em Armazenar *Embeddings* Sem Proteção

Embeddings faciais são representações vetoriais únicas do rosto de uma pessoa. Armazená-las sem proteção apresenta riscos sérios:

- ▶ **Embeddings são identificadores biométricos:** um atacante pode usar *embeddings* roubadas para reconstruir um rosto e se autenticar como outra pessoa
- ▶ **Vazamentos são irreversíveis:** diferente de senhas, as representações vetoriais são insubstituíveis

Problemas em Armazenar *Embeddings* Sem Proteção

Embeddings faciais são representações vetoriais únicas do rosto de uma pessoa. Armazená-las sem proteção apresenta riscos sérios:

- ▶ **Embeddings são identificadores biométricos:** um atacante pode usar *embeddings* roubadas para reconstruir um rosto e se autenticar como outra pessoa
- ▶ **Vazamentos são irreversíveis:** diferente de senhas, as representações vetoriais são insubstituíveis

Solução: usar **provas de conhecimento zero** para provar correspondência sem expor o vetor.

Integração com Provas de Conhecimento Zero

O cálculo da similaridade de cosseno pode ser embutido no circuito de prova de conhecimento zero:

- ▶ $\cos(\vec{A}, \vec{B}) > \tau$ é o circuito a ser verificado
- ▶ O limiar de similaridade τ é o parâmetro público
- ▶ As *embeddings* faciais representam os valores privados

Isso permite autenticação facial **sem revelar** os vetores faciais.

Vantagens do Método Proposto

O sistema de autenticação com provas de conhecimento zero traz benefícios significativos:

- ▶ **Privacidade Total:** o verificador não sabe quem é o usuário, apenas verifica se a prova é válida.
- ▶ **Resistência a Vazamentos:** nenhum dado biométrico é armazenado no sistema, por isso não há o que ser vazado ou roubado.

Fase de registro

1. O Usuário U inicia o processo de registro enviando sua face f_{reg} para o Modelo de IA M :

$$M \xleftarrow{f_{reg}} U$$

2. M extrai a *embedding* e_{reg} de f_{reg} ao calcular:

$$e_{reg} := Emb(f_{reg})$$

3. M envia a *embedding* e_{reg} para U :

$$M \xrightarrow{e_{reg}} U$$

4. U gera uma chave simétrica k e criptografa sua *embedding*:

$$c_{reg} := Enc_k(e_{reg})$$

Fase de registro

5. U envia a *embedding* criptografada c_{reg} para o Servidor S :

$$U \xrightarrow{c_{reg}} S$$

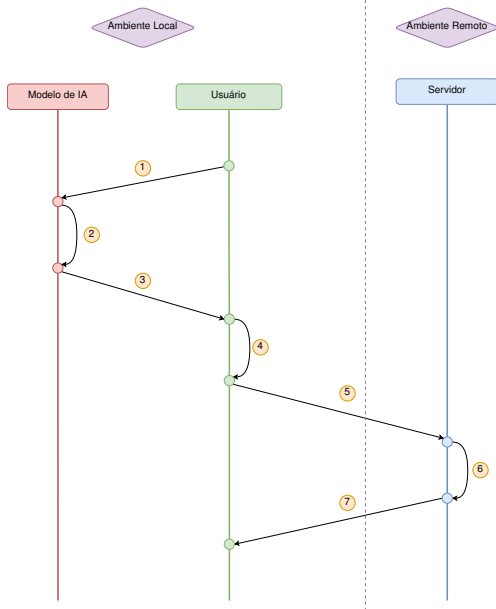
6. S armazena a *embedding* criptografada c_{reg} e gera um identificador único id :

$$id := GenId()$$

7. S envia o identificador id para U :

$$U \xleftarrow{id} S$$

Fase de registro



Fase de autenticação

1. O Usuário U inicia o processo de autenticação ao enviar o identificador id ao Servidor S :

$$U \xrightarrow{id} S$$

2. S envia a chave de prova S_p e uma cópia da *embedding* criptografada c_{reg} , que é correspondente ao id recebido, a U :

$$U \xleftarrow{S_p, c_{reg}} S$$

3. U descriptografa a *embedding* e_{reg} recebida ao calcular:

$$e_{reg} := Dec_k(c_{reg})$$

Fase de autenticação

4. U solicita a prova do protocolo zk-SNARK ao enviar a chave de prova S_p , a *embedding* recém descriptografada e_{reg} e uma nova captura de sua face f_{cur} para o Modelo de IA M :

$$M \xleftarrow{S_p, e_{reg}, f_{cur}} U$$

5. M extrai a *embedding* e_{cur} a partir da face f_{cur} e gera a prova zk-SNARK π ao computar:

$$e_{cur} := Emb(f_{cur})$$

$$\pi := Prove(S_p, x = \tau, w = (e_{cur}, e_{reg}))$$

6. M envia a prova zk-SNARK π a U :

$$M \xrightarrow{\pi} U$$

Fase de autenticação

7. U envia a prova zk-SNARK π a S :

$$U \xrightarrow{\pi} S$$

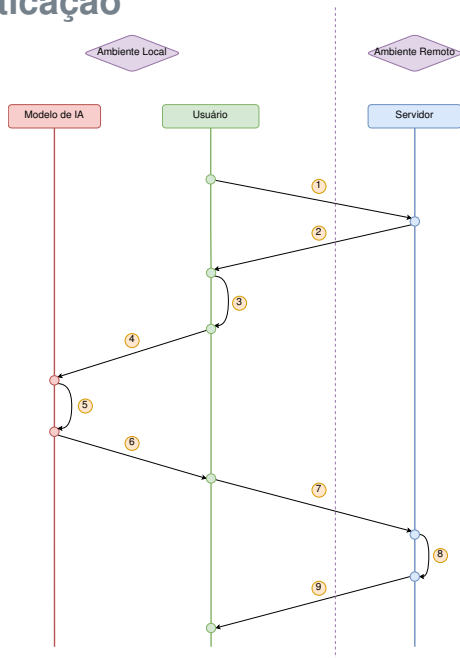
8. S verifica a validade da prova π ao calcular:

$$r := \text{Verify}(S_v, x = \tau, \pi)$$

9. S envia o veredito da autenticação r para U :

$$U \xleftarrow{r} S$$

Fase de autenticação



Tempo médio das operações do sistema

Métrica	Tempo Médio (s)	Responsável
Geração de <i>embeddings</i>	0.31	Modelo de IA
Geração da FRC	19.74	Servidor
Geração da prova	1.42	Modelo de IA
Verificação da prova	0.52	Servidor
Etapa de registro	0.39	Usuário
Etapa de autenticação	2.27	Usuário

Muito obrigado!



UNIVERSIDADE FEDERAL
DE SANTA CATARINA