Sistema de prova de conhecimento nulo baseado em isomorfismo de subgrafos

Alexandre Marques Albano da Silveira, Joseph Soares Alcântara e José Cláudio do Nascimento

***Resumo*— Sabe-se que o problema do isomorfismo de grafos (GI) tem um perfeito sistema de prova de conhecimento, mas uma análise de segurança para determinar se o problema de isomorfismo de subgrafo (SGI) satisfaz as três condições de sistema de prova de conhecimento nulo ainda não foi proposta. Se o problema GI fosse mais geral que o SGI, esta análise não seria necessária. Mas esse não é o caso. O problema SGI é mais geral do que o problema GI, requerendo uma investigação detalhada. Portanto, esse trabalho traz os detalhes da análise de segurança para o sistema de prova de conhecimento nulo construído a partir do problema de isomorfismo de subgrafos.**

***Palavras-Chave*— encriptação, problema de isomorfismo de subgrafo, prova de conhecimento nulo.**

***Abstract*— It is known that the graphs isomorphism problem (GI) has a perfect system of proof of knowledge, but a security analysis to determine if the subgraph isomorphism problem (SGI) meets the three conditions of zero-knowledge proof system was not proposed. If the GI problem was more general than the SGI, this analysis would not be necessary. But this is not the case. The SGI problem is more general than the GI problem, requiring a security analysis in the proposed of a zero-knowledge proof system. Therefore, this paper presents the details of safety analysis for the zero-knowledge proof system built from the subgraphs isomorphism problem.**

***Keywords*— encryption, subgraph isomorphism problem, zero knowledge proof.**

I. Introdução

Um algoritmo que é peça fundamental para muitos sistemas criptográficos é a prova de conhecimento nulo [1], [2], [3]. Uma prova de conhecimento é um método pelo qual uma das partes (o provador) pode provar a outra parte (o verificador) que uma declaração é verdadeira, sem transmitir qualquer informação adicional além do fato de que a afirmação é verdade. Através do trabalho [4] é bem conhecido que existem provas de conhecimento nulo para qualquer problema NP, desde que as funções alçapão existam. Ainda em [4] são mostrados sistemas de prova de conhecimento nulo que não necessitam de mensagens cifradas. Estes sistemas são baseados no problema de isomorfismo e não-isomorfismo de grafos. Como não se conheciam a mesma quantidade de propriedades nos problemas de isomorfismo de grafos que se conheciam nos problemas baseados em aritmética modular, com o passar do tempo poucos artigos foram surgindo e o problema de isomorfismo de grafos não chegou a ser empregado na prática.

Alexandre Marques Albano da Silveira, Joseph Soares Alcântara e José Cláudio do Nascimento. Programa de Pós Graduação em Engenharia Elétrica e de Computação - PPGEEC. Universidade Federal do Ceará, Sobral-CE, Brasil, E-mails: alexandrealbano@gtel.ufc.br, josephsoaresalcantara@gmail.com e claudio@deti.ufc.br. Este trabalho foi parcialmente financiado pela FUNCAP.

Outro motivo que favoreceu o não uso do problema de isomorfismo de grafos foi o desconhecimento da classe de complexidade computacional a que ele pertenceria, pois ele não estava na classe dos problemas polinomiais, mas também não estava na classe NP-Completo. Por isso foi criada uma classe IG para investigar somente esse problema [5], [6]. Depois surgiram algoritmos que reduzem significativamente a complexidade da solução, embora o problema ainda não possa ser resolvido em tempo polinomial [7], [8]. Assim, puseram sob suspeita os sistemas de prova de conhecimento nulo baseados no problema de isomorfismo de grafos.

Uma vantagem que os problemas com grafos têm sobre os problemas com aritmética modular é o tempo de execução. Isso desperta o interesse para aplicações em hardware mais simples [9]. No artigo [10] foi relatado o uso de um algoritmo de prova de conhecimento nulo baseado em isomorfismo de subgrafos. Este é um problema NP-Completo e o tempo de espera para encontrar o isomorfismo oculto é alto no caso árduo [12], [13], [14]. No entanto, não foi fundamentada a segurança nem o método de geração de grafos. O que exige uma pesquisa que melhor fundamente a sua aplicação. Embora seja conhecido que o problema SGI apresente um excelente método para assinatura de mensagens que pode ser implementado em microcontroladores [9], nem todo problema de isomorfismo de subgrafos é NP-Completo. Existem problemas em P como podemos encontrar em [15], [16], [17], [18]. Se o problema GI fosse mais geral que o SGI, esta análise não seria necessária, pois seria apenas um caso particular a aplicação do problema. Mas isso não é verdade. O SGI é um problema mais geral que o problema GI.

Neste trabalho é mostrado uma análise de segurança do perfeito sistema de prova de conhecimento nulo para o problema de isomorfismo de subgrafos. A análise de segurança feita nesse trabalho segue o caminho apresentado em [4] para o isomorfismo de grafos.

Esse trabalho está dividido da seguinte maneira: Na seção II temos uma breve revisão sobre linguagem NP-Completo e a sua relação com o perfeito sistema de prova de conhecimento nulo. Na seção III apresentamos um sistema de prova de conhecimento nulo usando o problema de isomorfismo de subgrafos, bem como uma análise das três propriedades que são necessárias para que o algoritmo seja um perfeito sistema de prova de conhecimento nulo. Por fim, na seção IV apresentamos a conclusão.

II. Complexidade de problemas e prova de conhecimento nulo

Para medir a eficiência de um algoritmo é necessário usar o tempo teórico que o programa leva para encontrar uma resposta em função dos dados de entrada. Se a dependência do tempo com relação aos dados de entrada for polinomial com o comprimento da entrada p($|x|$), o programa será considerado rápido. Caso a dependência do tempo seja exponencial com o comprimento da entrada ($a^{|x|}$ para a > 0), então o programa é considerado lento.

Stephen Cook [19] observou um fato simples: se um problema pode ser resolvido em tempo polinomial, então, pode-se também verificar se uma possível solução é correta em tempo polinomial. Uma maneira mais formal de expressar essa ideia é através da descrição do problema de decisão. Um problema de decisão pode ser visto como um problema de reconhecimento de linguagem. Considere $\sum^*$ como sendo o conjunto de todas as possíveis entradas para um problema de decisão. Considere $L \subseteq \sum^*$ como sendo o conjunto de todas as entradas cuja resposta é sim. Esse conjunto é chamado de linguagem correspondente ao problema. Em [19], Cook definiu classes de problemas quanto à complexidade destes e percebeu que alguns problemas não admitiam uma simplificação polinomial no seu tempo de execução, mas podiam ser certificados em tempo polinomial. Então, Cook definiu a classe dos problemas NP ou classe de complexidade NP como aquela para a qual apenas pode-se verificar, em tempo polinomial, se uma possível solução é correta.

Algoritmos eficientes para transportar ("traduzir") de uma linguagem L a outra linguagem L' podem ser criados. Assim, diz-se que L é polinomialmente redutível para uma linguagem L'. Nesse caso, simplesmente expressamos por $L \subseteq_p L'$ (lê-se L é p-redutível a L'). Leonid Levin [20] e Stephen Cook [19] observaram que dentre os problemas NP existem alguns que são mais difíceis do que outros, no sentido de que, resolvendo um desses problemas em tempo polinomial, então, todos os problemas em NP também podem ser resolvidos em tempo polinomial. Assim, a classe dos problemas NP-completos é o subconjunto dos mais difíceis problemas não-determinísticos polinomiais.

A assimetria entre a complexidade da tarefa de verificação e a complexidade da tarefa do ato de provar faz com que a classe NP seja vista como um sistema de prova. Basta notar que o processo de verificação é fácil, enquanto que chegar até a prova é uma tarefa difícil [21]. Duas propriedades de um sistema de prova são a sua validade e a completude. A propriedade de validade afirma que *o verificador deve possuir a habilidade de se proteger contra argumentos falsos para não ser convencido por eles*. Por outro lado, a propriedade de completude afirma à *capacidade de um provador convencer sempre com sentenças verdadeiras*.

Definição 1: Sistema de Prova Interativo. *Um par de máquinas interativas (**P,V**) é chamado um sistema de prova interativo para uma linguagem L, se a máquina **V** tem eficiência de tempo polinomial e satisfaz as seguintes condições:*

1) (*Completude*) *Para todo* $x \in L$,
$$Pr[(\boldsymbol{P},\boldsymbol{V})(x) = 1] > \frac{2}{3};$$
2) (*Validade*) *Para todo* $x \notin L$ *e um provador interativo* $\boldsymbol{P'}$, então
$$Pr[(\boldsymbol{P'},\boldsymbol{V})(x) = 1] < \frac{1}{3}.$$

É evidente que a definição dos limites para a satisfação das propriedades de completude e validade podem ser modificados. De forma geral, o sistema de prova interativo tem completude limitada por probabilidade p_c e tem validade limitada por p_v.

Definição 2: Perfeito Conhecimento Nulo. *Seja (**P,V**) um sistema de prova interativa para alguma linguagem L. Diz-se que (**P,V**) ou de fato a máquina do provador **P** tem perfeito conhecimento nulo se para toda máquina interativa probabilística de tempo polinomial **V**, existe um algoritmo S, tal que para toda entrada $x \in L$, as seguintes duas variáveis aleatórias são identicamente distribuídas:*

1) (***P,V***)(*x*)*, ou seja, a saída da máquina interativa **V** após interagir com a máquina **P** na entrada x;*
2) *S*(*x*)*, ou seja, a saída da máquina S na entrada x. A máquina S é chamada de simulador da interação de **V** com **P**.*

O real ganho de conhecimento de habilidade computacional é caracterizado quando uma máquina, após interagir com outra parte, é capaz de computar algo que antes ela não era capaz. No caso de nenhum ganho de conhecimento computacional, tem-se que, com o algoritmo $S(x)$, após a interação entre as máquinas ***V*** e ***P***, a máquina ***V*** não é capaz de fazer computações além do que ela era capaz de fazer sozinha antes da interação.

III. Sistema interativo de prova de conhecimento nulo para subgrafos isomorfos

Agora suponha que Paulo, um provador, que denotaremos por **P** conheça o isomorfismo entre um grafo G_0 com n vértices e um subgrafo, G_1, com m vértices, tal que $n > m$. Assim, temos uma função de correspondência de G_0 a G_1, $\sigma : G_0 \rightarrow G_1$. Sabendo que o isomorfismo de subgrafos é um problema NP-Completo, considere n e m suficientemente grandes, tal que, qualquer outra máquina probabilística de tempo polinomial não encontre o isomorfismo entre G_0 e G_1 em tempo hábil. Assim, Paulo guardará como segredo a transformação $\sigma : G_0 \rightarrow G_1$ e dirá a Victor, um verificador, denotado pela letra **V**, que conhece esse segredo. Para provar a **V** de que ela está falando a verdade sem revelar o segredo, ela terá que provar em tempo hábil desafios lançados por **V** como descrito no seguinte algoritmo:

Protocolo 1 (Sistema de prova de conhecimento nulo para isomorfismos de subgrafos) - O seguintes passos são repetidos n vezes:

1) **P** gera a função de correspondência $\lambda : G_0 \rightarrow H$ e envia H para **V** (λ é uma permutação e H é um grafo isomorfo a G_0, que é, consequentemente, subgrafo isomorfo a G_1);

2) $\mathbf{V}$ gera aleatoriamente um bit b e envia o bit a $\mathbf{P}$;
3) $\mathbf{P}$ envia a função de correspondência $\xi_b = \sigma^b \circ \lambda$ para $\mathbf{V}$;
4) Se o valor de $b = 0$, então o verificador usa a função de correspondência $\xi_0 = \lambda$ para testar se H é isomorfo a G_0. Se $b = 1$, então o verificador usa a função de correspondência $\xi_1 = \sigma \circ \lambda$ para testar se H é um subgrafo isomorfo de G_1.

Quando se olha o protocolo acima como um sistema de prova interativo, percebe-se que a entrada desse sistema é o par de grafos isomorfos G_0 e G_1. Então, a sentença dessa linguagem é $(G_0, G_1) \in SGI$. Notamos que a verificação dessa sentença sempre é feita indiretamente através de $\sigma^b \circ \lambda : H \rightarrow G_b$. Portanto, a função de correspondência σ sempre será um segredo mantido pelo provador.

Quanto a eficiência de execução do Protocolo 1, o programa do provador é executado por uma máquina probabilística de tempo polinomial, ou seja, apenas uma permutação escolhida aleatoriamente é realizada no Passo 1. O programa do verificador pode ser executado em tempo polinomial determinístico no passo 4. Denotaremos esse programa por

$$V(H, G_b, \xi) = \begin{cases} True & \text{se} \quad \xi : H \rightarrow G_b \\ False & \text{se} \quad \xi : H \rightarrow \tilde{H} \neq G_b \end{cases}$$

A escolha do bit de desafio é uma simples computação probabilística no Passo 2 e a verificação é $O(n^2)$ independente do bit escolhido. Assim, queremos verificar se a linguagem SGI tem um perfeito sistema de prova de conhecimento nulo.

Proposição 1 A linguagem SGI tem um perfeito sistema de prova interativa com conhecimento nulo. Para verificadores limitados por máquinas de tempo polinomial (determinística ou probabilística), o Protocolo 1 satisfaz as seguintes afirmativas:

1) Se G_1 é um grafo isomorfo a um subgrafo contido em G_0, então o verificador sempre aceita quando interage com $\mathbf{P}$;
2) Se G_1 não é um grafo isomorfo a um subgrafo contido em G_0, então a entrada será rejeitada com probabilidade menor do que $\frac{1}{2}$;
3) $\mathbf{P}$ realiza provas com perfeito conhecimento nulo.

Quando n interações entre o verificador e o provador são realizadas a probabilidade de erro para a validade é limitada por $\frac{1}{2^n}$. Deve ser enfatizado que todas as computações probabilísticas são completamente independentes para cada iteração das máquinas probabilísticas, consequentemente, isto é válido para interações entre as máquinas. Portanto, segue a prova da Proposição 1:

1) **Prova:** Claramente, se o grafo G_1 é um grafo isomorfo a um subgrafo de G_0, então o grafo H construído por $\mathbf{P}$ no passo 1 sempre retornará True na no passo 4, $V(H, G_0, \xi_0) = V(H, G_1, \xi_1) = True$. Consequentemente se cada parte segue o que está prescrito no protocolo, então $\mathbf{V}$ sempre aceita os argumentos do provador.
2) **Prova:** Já sabemos que um provador desonesto não pode fazer provas. Mas o verificador pode vir a ser um futuro provador desonesto após uma real interação com o provador? A resposta é não se o verificador, sem qualquer interação com o provador, conseguir apenas tuplas que podem ser obtidas através de uma simulação com baixo esforço computacional. O baixo esforço computacional a que nos referimos é um algoritmo de tempo polinomial em máquina probabilística. Neste caso, considerando um verificador desonesto, o que deve ser demonstrado é que o provador realiza provas sem que o verificador tenha qualquer aprendizado (ganho de conhecimento) sobre suas habilidades computacionais [22].
3) **Prova:** Já sabemos que um provador desonesto não pode fazer provas. Mas o verificador pode vir a ser um futuro provador desonesto após uma real interação com o provador? A resposta é não, se o verificador, sem qualquer interação com o provador, conseguir apenas tuplas que podem ser obtidas através de uma simulação com baixo esforço computacional. O baixo esforço computacional a que nos referimos é um algoritmo de tempo polinomial em máquina probabilística. Neste caso, considerando um verificador desonesto, o que deve ser demonstrado é que o provador realiza provas sem que o verificador tenha qualquer aprendizado (ganho de conhecimento) sobre suas habilidades computacionais [22]. Assim, de posse de um simulador, este terá que produzir n tuplas (H, b, ξ) onde H é gerado uniformemente (pois o provador é honesto) e b é gerado de acordo com $\mathbf{V}$. Então uma possível simulação pode ser realizada da seguinte forma:
   a) O simulador escolhe $b \in \{0, 1\}$ de forma uniformemente aleatória;
   b) O simulador escolhe uma permutação $\xi : G_b \rightarrow H$ de maneira uniformemente aleatória. Se $b = 0$, então ξ será uma permutação sobre n vértices. Caso contrário, ξ será uma permutação sobre m vértices;
   c) O simulador aplica um algoritmo probabilístico de tempo polinomial, $V_1(H, w_1)$, em que $w_1 = \epsilon$ significa sem símbolo inicial, e verifica se o bit c calculado por V_1 é igual a b;
      i) Caso $c = b$, então a simulação foi feita com sucesso e a tupla nesta interação deve ser (H, c, ξ). A computação auxiliar feita por $\mathbf{V}$ com o fim de ser utilizada em interações futuras é gravada em w_1;
      ii) Se $c \neq b$, então o simulador volta para o Passo (a).

   Após a primeira iteração $(k > 1)$, no Passo (c), o simulador aplica $V_k(H, w_k)$, um algoritmo probabilístico de tempo polinomial para escolha do bit c de desafio na k-ésima iteração que usa dados auxiliares $w_k \in \{0, 1\}^*$ obtidos em computações anteriores, e verifica se o bit c calculado por V_k é igual a b;
      i) Caso $c = b$, então a simulação foi feita com sucesso e a tupla nesta interação deve ser (H, c, ξ). A computação auxiliar feita por $\mathbf{V}$

com o fim de ser utilizada em interações futuras é gravada em w_{k+1};

ii) Se $c \neq b$, então o simulador volta para o Passo (a).

A principal diferença no algoritmo de simulação para gerar as tuplas do protocolo de prova de conhecimento nulo baseado SGI para o de GI em [22] é o passo b). A escolha do bit b no passo a) altera a entrada do gerador aleatório da permutação, definindo o tamanho sobre qual conjunto de vértices a permutação ocorrerá. Deixando-o dependente do passo anterior. Essa alteração não aumenta a complexidade da simulação. No demais, as consequências são as mesmas. O verificador não ganha nenhum conhecimento adicional e o provador realiza um perfeita prova de conhecimento nulo. O objetivo de V_k e w_k é escolher o bit c de forma a minimizar as chances de fracasso no passo ii). Podemos imaginar que o algoritmo V_k sempre terá mais chances do que V_{k-1} e que o dado w_k sempre será mais útil do que w_{k-1}. Mas para ter sucesso nessa simulação, não serão necessários tantos melhoramentos nesses algoritmos e dados. Pois no pior caso, em que os melhoramentos de V_k e w_k em cada iteração não acontecem, a probabilidade de $c = b$ é $\frac{1}{2}$, dado que b foi escolhido com distribuição de probabilidade uniforme e c poderá ser escolhido da mesma maneira. Mesmo nesta situação, o simulador terá sucesso, em média, após duas tentativas. Para finalizar, basta verificar que a sequência de tuplas (H, c, ξ) gerada pelo simulador tem exatamente a mesma distribuição que as tuplas produzidas por uma interação real com o provador. Por esta razão, estas sequências são computacionalmente indistinguíveis.

IV. Conclusões

Verificamos que sistemas de prova de conhecimento nulo baseado no problema de isomorfismo de subgrafos garante três importantes pontos de segurança: o verificador sempre aceita os argumentos do provador quando interage com ele, falsos isomorfismos não serão aceitos após n interações e que $\mathbf{P}$ realiza provas com perfeito conhecimento nulo. Além disso, temos um sistema de prova de conhecimento nulo baseado num problema NP-Completo sem a necessidade de cifra segura. Isso viabiliza a proposta [10], cuja motivação foi a facilidade de implementação desse algoritmo em máquinas de baixo poder computacional. Verificamos que a impossibilidade de transferência de prova ou perfeito conhecimento nulo é uma propriedade muito bem preservada com problemas de isomorfismo de subgrafos, pois as modificações que fizemos no simulador não alteraram a sua complexidade. Além disso, fato de um problema ser verificável em tempo polinomial, mas cuja demonstração exige a solução de um problema NP-Completo, fortalece a completude e a validade dos argumentos apresentados.

Referências

[1] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology - CRYPTO'86*, vol. 163, pp. 186–197, 1987.

[2] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology - EUROCRYPT'97*. Springer-Verlag, 1997, pp. 103–118.

[3] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Proceedings on Advances in cryptology*, ser. CRYPTO '88. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 319–327. [Online]. Available: http://dl.acm.org/citation.cfm?id=88314.88969

[4] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, pp. 690–728, 1991.

[5] K. S. Booth and C. S. Colbourn, "Problems polynomially equivalent to graph isomorphism," Computer Science Department, University of Waterloo, Waterloo, Ontario, Tech. Rep. CS-77-04, 1977.

[6] J. Köbler, U. Schöning, and J. Torán, *The graph isomorphism problem: its structural complexity*. Basel, Switzerland, Switzerland: Birkhauser Verlag, 1993.

[7] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Performance evaluation of the vf graph matching algorithm," in *Proceedings of the 10th International Conference on Image Analysis and Processing*, ser. ICIAP '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1172–. [Online]. Available: http://dl.acm.org/citation.cfm?id=839281.840896

[8] B. D. McKay, "Practical Graph Isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

[9] L. Szöllösi, T. Marosits, G. Fehér, and A. Recski, "Fast digital signature algorithm based on subgraph isomorphism," in *Proceedings of the 6th international conference on Cryptology and network security*, ser. CANS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 34–46. [Online]. Available: http://dl.acm.org/citation.cfm?id=1778554.1778558

[10] L. B. e. E. M. J. M. Kizza, "Using Subgraph Isomorphism as a Zero Knowledge Proof Authentication in Timed Wireless Mobile Networks," *International Journal of Computing and ICT Research*, vol. 4, 2010.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979, gT48.

[12] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976. [Online]. Available: http://doi.acm.org/10.1145/321921.321925

[13] ——, "Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism," *J. Exp. Algorithmics*, vol. 15, pp. 1.6:1.1–1.6:1.64, Feb. 2011. [Online]. Available: http://doi.acm.org/10.1145/1671970.1921702

[14] C. Solnon, "Alldifferent-based filtering for subgraph isomorphism," *Artificial Intelligence*, vol. 174, no. 12-13, pp. 850–864, Aug. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370210000718

[15] D. Raviv, R. Kimmel, and A. M. Bruckstein, "Graph isomorphisms and automorphisms via spectral signatures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1985–1993, 2013.

[16] C. Puech and R. Reischuk, Eds., *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1046. Springer, 1996.

[17] A. Gupta and N. Nishimura, "Characterizing the complexity of subgraph isomorphism for graphs of bounded path-width." in *STACS*, ser. Lecture Notes in Computer Science, C. Puech and R. Reischuk, Eds., vol. 1046. Springer, 1996, pp. 453–464.

[18] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," *J. Graph Algorithms & Applications*, vol. 3, pp. 1–27, 1999.

[19] S. A. Cook, "The complexity of theorem-proving procedures," in *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1971, pp. 151–158.

[20] L. A. Levin, "Universal sequential search problems," *Jour Probl. Peredachi Inf.*, vol. 9, pp. 115–116, 1973.

[21] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001, vol. 1.

[22] P. Mateus, "Análise de sistemas de prova de conhecimento nulo," *Portuguese IBM Scientific Prize*, 2005.