Utilização de Animações para Comparação de Hashes em Votação Eletrônica via Internet

Jessica Yumi Nakano Sato[1], **Jorge Miguel Ribeiro**[1],
Roberto Samarone Araujo[2], **José Coelho de Pina**[1], **Daniel Macêdo Batista**[1]

[1]Departamento de Ciência da Computação – Universidade de São Paulo (USP)

[2]Faculdade de Computação – Universidade Federal do Pará (UFPA)

jyns1703@usp.br, jorgemiguelribeiro92@gmail.com

rsa@ufpa.br,{coelho,batista}@ime.usp.br

Abstract. *Secure electronic voting systems via the Internet require data comparison at various stages of the voting process. Some of this data is hashes, usually represented by strings. Comparing these strings by humans tends to be tedious, which means that it is not always performed correctly. This paper reports preliminary results of a software system that advances state of the art in electronic voting via the Internet by comparing hashes represented by animations. The basis of the system are the free software hashify, responsible for the animations, and the free software Helios, responsible for the votes. Results show that it is possible to compare Helios hashes using hashify animations.*

Resumo. *Sistemas seguros de votação eletrônica via Internet requerem comparação de dados em várias etapas do processo de votação. Alguns desses dados são hashes, representados geralmente por strings. A comparação dessas strings por seres humanos tende a ser tediosa, o que faz com que a mesma nem sempre seja realizada corretamente. Este artigo relata resultados preliminares de um sistema de software que avança o estado da arte em votação eletrônica via Internet por meio da comparação de hashes representados por animações. A base do sistema são o software livre hashify, responsável pelas animações, e o software livre Helios, responsável pelas votações. Resultados mostram que é possível comparar os hashes do Helios usando as animações do hashify.*

1. Introdução

Nos últimos anos sistemas de votação via Internet têm sido desenvolvidos a partir de modernos protocolos criptográficos. Uma das propriedades empregadas nesses protocolos é garantir que cada votante possa certificar-se de que seu voto foi corretamente armazenado. Outra propriedade é possibilitar que qualquer observador(a) externo(a) seja capaz de verificar que todos os votos armazenados foram corretamente contados [de Marneffe et al. 2009]. Tudo isso deve ocorrer sem comprometer o sigilo do voto.

Um dos sistemas que objetiva satisfazer tais propriedades é o `Helios Voting` [Adida 2008]. Esse sistema está disponível como *software livre* (sob licenças Apache 2.0 e GNU GPL 3+ [Helios Voting 2022]) e tem sido amplamente utilizado em eleições reais, como aquelas realizadas pela Sociedade Brasileira

de Computação (https://www.sbc.org.br/institucional-3/eleicoes/82-plano-de-gestao-2013-2015) e pela Universidade de São Paulo (https://votacao.usp.br/info).

Dentre as várias etapas do processo de votação do Helios, existe a etapa de auditoria. Nessa etapa, cada votante pode verificar o *hash* da eleição para confirmar que participou da eleição correta e o *hash* do texto cifrado recebido ao votar. As verificações são realizadas por meio de comparação de *strings*. Entretanto, essa comparação pode facilmente confundir votantes levando-os a cometer erros. Como o Helios é proposto para ser um sistema de votação online para o público geral, é importante que a auditoria não seja confusa, principalmente para pessoas que não tenham um bom conhecimento teórico a respeito de criptografia. Dessa forma, convertendo os *hashes* em imagens, procura-se aumentar a segurança e confiança no sistema ao incentivar que cada votante participe ativamente da auditoria.

Com o objetivo de permitir a comparação de *hashes* criptográficos presentes em certificados digitais do protocolo HTTPS de um modo mais agradável, habitual e menos propenso a erros, o software hashify foi desenvolvido [Ribeiro et al. 2020] e está disponível como software livre [Ribeiro and de Pina 2020][1]. Ele funciona recebendo um *hash* como entrada e devolvendo uma animação como saída. Essa animação é muito mais fácil de ser comparada do que a *string* referente ao *hash*. Recentemente, o desempenho do hashify foi avaliado e concluiu-se que ele pode vir a ser integrado em aplicativos móveis [Carvalho et al. 2021].

Considerando os bons resultados alcançados pelo hashify na geração de animações, levantou-se a hipótese de que os algoritmos do hashify podem ser usados para gerar animações dos *hashes* gerados pelo Helios Voting. Em outras palavras, a hipótese é verificar se os algoritmos do hashify facilitam o processo de auditoria. Nesse contexto, este artigo traz como principal contribuição resultados preliminares alcançados no projeto e desenvolvimento de um sistema de software voltado para provar essa hipótese.

O restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta os conceitos preliminares para a compreensão do que será apresentado e os trabalhos relacionados. A Seção 3 descreve a metodologia utilizada no projeto e desenvolvimento do sistema de software proposto para permitir a comparação dos *hashes* do Helios Voting por meio de animações do hashify. A Seção 4 apresenta os resultados preliminares obtidos. O artigo é finalizado com a apresentação das conclusões e dos próximos passos na Seção 5.

2. Conceitos Preliminares e Trabalhos Relacionados

Em certas aplicações, *hashes* precisam ser armazenados em um dispositivo confiável e comparados em tempo de execução com os *hashes* informados por uma entidade que pode não ser confiável. Por conta disso, comparar *hashes* é uma tarefa importante. Entretanto, como apresentado em [Perrig and Song 1999], seres humanos são lentos e propensos a erros ao verificar *strings* "sem sentido". Em [Tan et al. 2017] são apresentadas evidências de que isso pode fazer com que as pessoas passem a comparar apenas o início e o final

[1]Para ver o hashify em funcionamento, assista https://youtu.be/raFkfRRXBM0

dos *hashes* (Por mais complexo que seja conseguir *hashes* que possuam apenas alguns bits similares, isso não é impossível considerando o poder de processamento de computadores atuais).

Diversos trabalhos nos últimos anos têm proposto que, ao invés de compararem *hashes* no formato de *strings*, seres humanos comparem *hashes* no formato de imagens (animadas ou não) [Perrig and Song 1999, Lin et al. 2010, Davis 2011, Maina Olembo et al. 2014, Tan et al. 2017], o que deveria ser mais fácil e natural. Entretanto, esses trabalhos possuem algumas limitações, como por exemplo serem vulneráveis a ataques de similaridade por usarem diretamente os bits dos *hashes*, necessitarem de GPU para geração das imagens em um intervalo de tempo aceitável ou utilizarem elementos gráficos que podem ser difíceis de distinguir. Em [Ribeiro et al. 2020] é apresentado o `hashify`, um software que recebe *hashes* como entrada e devolve imagens animadas como saída. Ele resolve os problemas dos trabalhos anteriores encontrados na literatura e funciona como uma extensão do Firefox gerando uma estampa visual única a partir do *fingerprint* do certificado digital de um *website*.

A Figura 1 mostra uma sequência da animação gerada pelo `hashify` para o *hash* 6f34467165745132417831 4e562f49304c4468756472796b62336a58654345364e754c6950714e63525469.

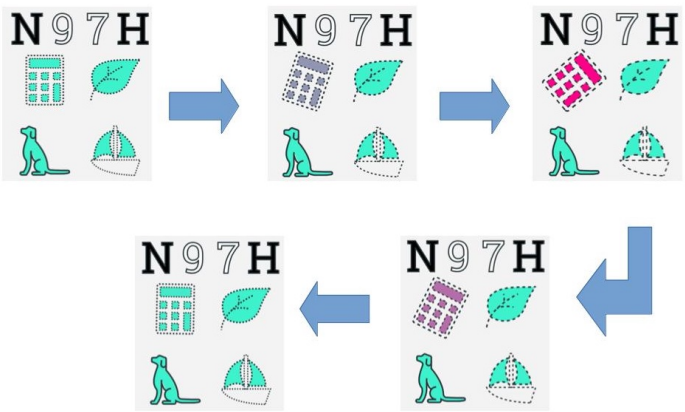

Figura 1. Exemplo de quadros gerados pelo `hashify`.

Em [de Marneffe et al. 2009] é apresentado o `Helios Voting`, um sistema para votação eletrônica via Internet de código aberto criado em 2008. Ele tem sido bastante utilizado ao redor do mundo, principalmente para eleições em ambientes acadêmicos. Os votos no `Helios Voting` são criptografados usando criptografia de cédula baseada no navegador web. Uma vez carregada, a aplicação web de criptografia não acessa a rede até que o voto seja criptografado e pronto para ser lançado. Em navegadores tradicionais rodando em um computador moderno, a criptografia leva alguns poucos segundos para ser empregada. A garantia de que o voto foi corretamente lançado é obtida por um protocolo de votação que considera o *fingerprint* gerado para o voto. A auditoria no `Helios Voting` consiste na comparação de *hashes* criptográficos.

Neste artigo considera-se que o `hashify` pode ser usado para a auditoria do `Helios Voting`. Vale observar que para esse uso é importante definir como será gerada a imagem a partir dos *hashes* gerados durante a votação, definindo dessa forma as mudanças no algoritmo original do `hashify`.

2.1. O Sistema de Votação Helios

A Figura 2 descreve em alto nível, e de forma resumida, o processo de votação realizado pelo `Helios Voting` [Neumann and Volkamer 2012]. O **EB** (*Election Builder* – Criador da Eleição) determina candidaturas e votantes, fornecendo dados de login e URL para votantes; **Votante/JavaScript** é o código que permite que votantes processem seus votos e interajam com o sistema. O JavaScript é lançado pelo site do `Helios Voting`. A aleatoriedade usada para criptografar a escolha dos(as) votantes é armazenada graças a esse script; O **AS** (*Authentication Server* – Servidor de Autenticação) permite que votantes se autentiquem e submetam o voto criptografado. O **BB** (*Bulletin Board* – Boletim de urna) é um canal público no qual votantes podem verificar se os seus votos lançados estão armazenados.

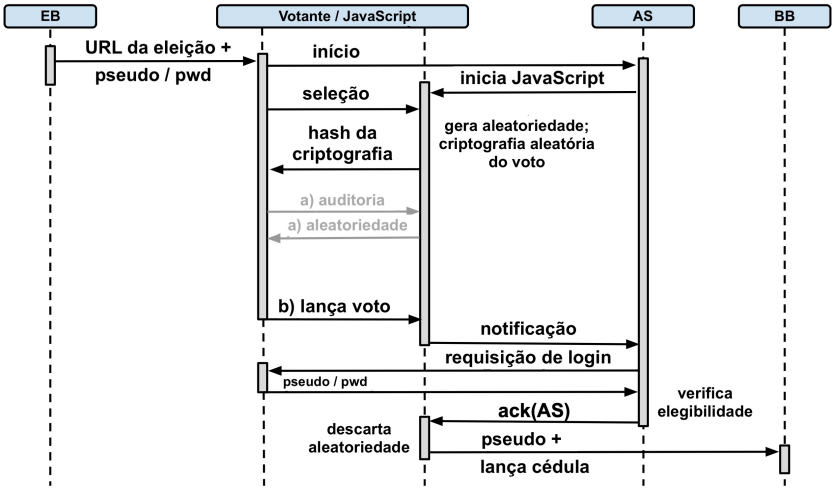

Figura 2. Esquema do `Helios` (Traduzido de [Neumann and Volkamer 2012]).

Dentre as várias etapas do processo de votação do `Helios Voting`, vale destacar a etapa de auditoria, que pode ser melhorada em termos de usabilidade para que votantes, que não sejam especialistas em desenvolvimento de software para votação eletrônica, sejam capazes de realizar os procedimentos da forma correta [PANIZO ALONSO et al. 2021]. Nessa etapa de auditoria, do ponto de vista da verificação de um voto único, há duas comparações que votantes precisam realizar:

- Verificar o *hash* da eleição para confirmar que participou da eleição correta;
- Verificar o *hash* do texto cifrado recebido ao votar.

Verificar um *hash* significa compará-lo considerando uma cópia confiável e uma cópia não confiável. Esses *hashes* funcionam como uma impressão digital (*fingerprint*) do dado ao qual ele se refere (Eleições e votos, no caso de votação eletrônica). Considerando que votantes sejam capazes de obter, e manter, um *hash* de forma segura, um modo de validar o dado é comparando o seu *hash*, armazenado de forma segura, com o *hash* informado posteriormente pelo sistema de votação eletrônica. Entretanto, a comparação de *strings* que representam *hashes* é difícil de ser realizada visualmente, como pode ser observado no caso abaixo em que há apenas 1 caractere diferente entre os *hashes*:

```
6f3446716574513241783 14e562f49304c4468756472796b62336a58654345364e754c6950714e63525469
6f3446716574513241783 14e562f49304c4468756472796d62336a58654345364e754c6950714e63525469
```

3. Metodologia

Embora o código-fonte do `Helios Voting` esteja disponível publicamente, decidiu-se, nessa primeira etapa da pesquisa, utilizar uma instância disponibilizada pelos desenvolvedores do `Helios Voting` em `https://vote.heliosvoting.org/` para que qualquer pessoa possa testar o sistema. Dessa forma, o fluxo considerado foi:

1. Eleição e votantes são criados em `https://vote.heliosvoting.org/`;
2. Votante vota e copia os dois *hashes* gerados pelo sistema web do passo 1: *hash* que identifica a eleição e *hash* que identifica o voto que foi registrado;
3. Votante acessa um segundo sistema web, baseado no `hashify`, que recebe os *hashes* do passo 2 e devolve duas animações, uma referente a cada *hash*;
4. Votante salva as duas animações do passo 3 como vídeos, idealmente no seu *smartphone*;
5. Ao término da eleição, votante retorna aos sistemas web dos passos 1 e 3 e verifica se as animações coincidem com aquelas armazenadas no passo 4. Caso elas não coincidam, a entidade responsável pela eleição deve ser notificada. Caso elas coincidam, considera-se que a votação não foi adulterada.

Até o momento, a principal tarefa realizada na pesquisa em andamento diz respeito ao sistema web do Passo 3 a fim de confirmar que é possível gerar as animações a partir dos *hashes* do `Helios Voting`. Observou-se que o `hashify` foi desenvolvido para receber como entrada *hashes* de certificados digitais (*hashes* SHA-256) em representação hexadecimal. Porém, embora os *hashes* gerados pelo `Helios Voting` sejam *hashes* baseados na função de *hash* SHA-256, eles são apresentados para a pessoa votante na representação base64, ou seja, como uma *string* de 43 caracteres, como por exemplo:

`o4FqetQ2Ax1NV/I0LDhudrykb3jXeCE6NuLiPqNcRTi`

Antes de alterar o algoritmo do `hashify`, os *hashes* gerados pelo `Helios Voting` foram convertidos caractere a caractere para uma representação em hexadecimal e testados no código do `hashify` sem modificações e os resultados foram positivos. As animações geradas foram específicas para cada *hash*, atendendo o objetivo dessa geração. Dessa forma, o algoritmo base do `hashify` não precisou ser modificado. Por exemplo, no caso da *string* de 43 caracteres apresentada acima, a sua representação em hexadecimal leva à animação apresentada na Figura 1. Entretanto, para garantir que as propriedades do `hashify` fossem mantidas, optou-se por transformar os *hashes* SHA-256 em base64 gerados pelo `Helios Voting` para a codificação hexadecimal de uma forma diferente. Para isso foi necessário decodificar o *hash* na base64 para binário e então converter para hexadecimal. Dessa forma, fica mais garantido que todas as propriedades relativas à probabilidade de colisão do `hashify` ficam mantidas, dado que o tipo de *hash* continuaria sendo o mesmo para o qual ele foi projetado.

4. Resultados Preliminares

Os principais resultados até o momento dizem respeito à página web que foi desenvolvida para receber os dois *hashes* gerados pelo `Helios Voting` em uma votação: o *hash* da

eleição e o *hash* da cédula de votação. Cada um desses *hashes* gera uma animação específica, que devem ser salvas pelo votante para comparação após o término da eleição. Alguns requisitos em termos da interface com o usuário foram implementados nessa página para que ela atendesse o propósito do projeto:

- Identificação de qual *hash* está sendo esperado no momento (primeiro informa-se que o *hash* da eleição deve ser inserido e, depois, o *hash* da cédula);
- Lembrete, em um texto flutuante, de que a *string* esperada deve possuir 43 caracteres e nenhum espaço;
- Validação da entrada (caso a *string* não tenha 43 caracteres, um retorno visual – a caixa de texto fica em vermelho – é apresentado para a pessoa votante)

A Figura 3 mostra uma captura de tela da página web com o lembrete sobre o formato esperado da *string*. A caixa de seleção "Habilitar loop da Animação" serve para manter a animação em repetição. A caixa de seleção "Decodificar Hash" transforma o *hash* SHA-256 em base64 para binário e depois para hexadecimal. Quando não selecionado, tratamos o *hash* como uma *string* comum e convertemos para hexadecimal caractere a caractere. No futuro, planeja-se remover essa opção uma vez que se tenha provado se 1- a conversão da base64 para hexadecimal é suficiente ou 2- o algoritmo do `hashify` funciona, sem colisão, para *strings* em hexadecimal de 43 caracteres. A Figura 4 mostra uma captura de tela do momento em que a pessoa votante preenche o campo do *hash* da eleição mas está faltando 1 caractere.

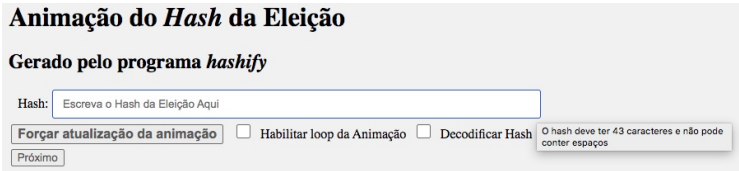

Figura 3. Captura de tela da página com a mensagem sobre o formato esperado.

Figura 4. Captura de tela da página com erro na *string* (está faltando 1 caractere).

A Figura 5 mostra uma captura de tela do sistema após a pessoa corrigir a entrada. A animação gerada para esse *hash* está representada na parte inferior da página. O botão "Próximo", ao ser pressionado, recarregará a página, dessa vez solicitando o *hash* da cédula. A Figura 6 exibe a captura com o mesmo *hash* mas com a opção de transformar a *string* em binário e depois em hexadecimal.

5. Conclusão e Próximos Passos

Este artigo apresentou uma prova de conceito de que é possível utilizar o `hashify`, um software para geração de animações a partir de *hashes*, com a finalidade de facilitar a auditoria do `Helios Voting`.

Figura 5. Captura de tela da página com a *string* corrigida e com a animação.

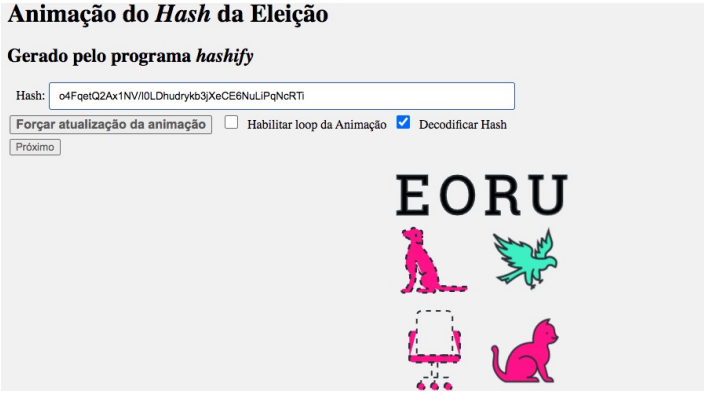

Figura 6. Captura de tela da página com a *string* corrigida e com a animação para o *hash* convertido para binário e depois para hexadecimal.

Após o entendimento de que *hashes* de 43 caracteres são gerados pelo `Helios Voting`, um para representar a eleição e um para representar o voto, um sistema web baseado no `hashify` foi desenvolvido para receber esses *hashes* como entrada e convertê-los no formato em hexadecimal esperado pelo algoritmo de geração de animações do `hashify`. Experimentos mostraram que as animações foram geradas como esperado. Essas animações podem ser armazenadas, por exemplo, em um *smartphone* no momento em que a pessoa votante vota e comparadas posteriormente ao término da eleição.

Como próximos passos pretende-se integrar o sistema web ao `Helios Voting` permitindo que as animações sejam geradas automaticamente no momento do voto, sem necessidade da utilização de um segundo sistema web, e enviando-as por e-mail para a pessoa votante poder mantê-las armazenadas para posterior auditoria. Também pretende-se realizar pesquisas com seres humanos para avaliar na prática se os *hashes* visuais de fato aumentam o engajamento no processo de auditoria de eleições.

Agradecimentos

Esta pesquisa é financiada por CNPq (proc. 101921/2022-5). Ela também é parte do INCT da Internet do Futuro para Cidades Inteligentes, financiado por CNPq (proc. 465446/2014-0), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e FAPESP (procs. 14/50937-1 e 15/24485-9).

Referências

Adida, B. (2008). Helios: Web-based Open-Audit Voting. In *Anais do 17th USENIX Security Symposium (USENIX Security 08)*, pages 335–348. USENIX.

Carvalho, H., Ribeiro, J., Batista, D., and Pina, J. (2021). Análise de Desempenho de uma Ferramenta para Visualização de Hashes em Dispositivos Móveis. In *Anais Estendidos do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 248–255.

Davis, C. (2011). RoboHash. Disponível em `https://robohash.org/`. Acessado em 30 de Julho de 2020.

de Marneffe, O., Pereira, O., and Quisquater, J.-J. (2009). Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *Anais do (EVT/WOTE 09)*. USENIX.

Helios Voting (2022). GitHub - benadida/helios-server: Helios server. Disponível em `https://github.com/benadida/helios-server`. Acessado em 26 de Junho de 2022.

Lin, Y.-H., Studer, A., Chen, Y.-H., Hsiao, H.-C., Kuo, L.-H., McCune, J. M., Wang, K.-H., Krohn, M., Perrig, A., Yang, B.-Y., et al. (2010). SPATE: Small-Group PKI-less Authenticated Trust Establishment. *IEEE Transactions on Mobile Computing*, 9(12):1666–1681.

Maina Olembo, M., Kilian, T., Stockhardt, S., Hülsing, A., and Volkamer, M. (2014). Developing and Testing SCoP–a Visual Hash Scheme. *Information Management & Computer Security*, 22(4):382–392.

Neumann, S. and Volkamer, M. (2012). Formal Treatment of Distributed Trust in Electronic Voting. In *Anais do International Conference on Internet Monitoring and Protection (ICIMP 2012)*, pages 30–39.

PANIZO ALONSO, L., GASCÓ, M., MARCOS del BLANCO, D. Y., Hermida Alonso, J. A., Barrat, J., and Aláiz Moreton, H. (2021). E-Voting System Evaluation Based on The Council of Europe Recommendations: Helios Voting. *IEEE Transactions on Emerging Topics in Computing*, 9(1):161–173.

Perrig, A. and Song, D. (1999). Hash Visualization: a New Technique to Improve Real-World Security. In *Anais do CryTEC '99*.

Ribeiro, J. M., Batista, D. M., and de Pina, J. C. (2020). hashify: Uma Ferramenta para Visualização de Hashes com Animações. In *Anais do Salão de Ferramentas do XX SBSeg*.

Ribeiro, J. M. and de Pina, J. C. (2020). Jorge Miguel Ribeiro / Hashify · GitLab. Disponível em `https://gitlab.com/jorgemiguelribeiro92/hashify`. Acessado em 22 de Junho de 2022.

Tan, J., Bauer, L., Bonneau, J., Cranor, L. F., Thomas, J., and Ur, B. (2017). Can Unicorns Help Users Compare Crypto Key Fingerprints? In *Anais do ACM CHI'17*, pages 3787–3798.