IMPLEMENTAÇÃO DA TÉCNICA DE CHECKSUM PARA VALIDAÇÃO DE AUTENTICIDADE DE FOTOGRAFIAS DIGITAIS.

Implementação de checksum para validação de fotografias digitais

Gabriel Pagelkopf[1]; Ana Elisa Schmidt[2];

RESUMO

Com o aumento do conhecimento básico em tecnologia, as ferramentas de manipulação de imagem de forma maliciosa vem crescendo drasticamente, e a operação cópia e cola vem sendo a mais utilizada, temos a necessidade de verificar a autenticidade da imagem. Com isto em mente decidimos aplicar a técnica checksum que irá gerar um código de validação assim podendo verificar a autenticidade da foto.

Palavras-chave: manipulação. Imagem. Checksum. Cópia. Cola.

INTRODUÇÃO

Atualmente, com tanta tecnologia ao nosso redor e a facilidade de usa-la maliciosamente precisamos sempre estar verificando as informações e arquivos que recebemos.

Por ser uma operação simples e rápida de ser feita a cópia e cola é a mais utilizada, mas em contramão é de fácil verificação, por isso decidimos utilizar a técnica checksum que já é muito utilizada neste caso, e para guardar o código gerado usaremos o LBS (least significant bit), onde guardará o código no bit de menor importância para a qualidade da imagem.

Para a aplicação iremos criar um web service, em HTML e Ruby, pela facilidade da linguagem e aplicação de tal função, além de encontrar inúmeros artigos e matérias sobre esta utilização, apesar da maioria estar em inglês facilitou a aplicação.

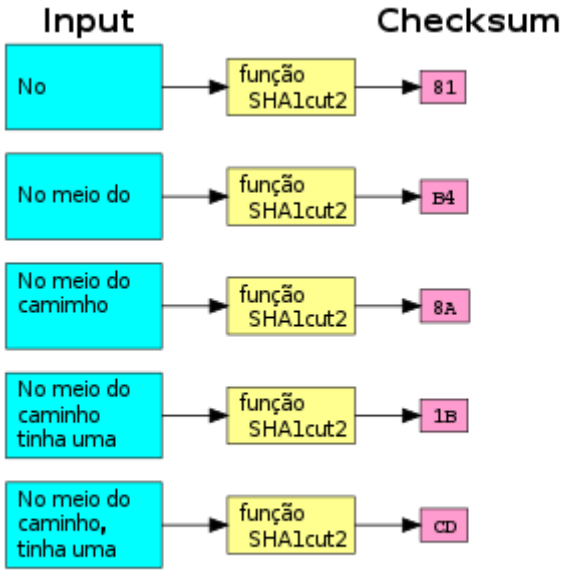
PROCEDIMENTOS METODOLÓGICOS

[1] Estudante de Graduação em Sistemas de Informação, IFC- Camboriú. E-mail: gabriel.pagelkopf@gmail.com
[2] Doutora em Informática, PUC-Rio; professora do Instituto Federal Catarinense. E-mail: anaelisa@ifc-camboriu.edu.br

Primeiro para me colocar sobre o assunto, estudei como a fotografia digital funciona, desde a digitalização até os meios de manipula-la, por ser digital você consegue guardar palavras, frases, códigos em seu arquivo se você abrir como um arquivo .txt, pois uma imagem não passa de vários bits com seu valor que corresponde a uma cor, cada um com seu valor, mas abrindo em forma .txt você consegue esconder informação no bit menos significante (LBS) da imagem de uma forma simples e rápida.

Como mencionado, uma imagem são vários bits e com isso em mente procuramos um método verificar a autenticidade da imagem baseado nessa informação, o checksum faz exatamente isso, ele calcula todos os bits da imagem e gera um código, existem outros métodos mais simples mas se você apenas mudar a ordem gerará o mesmo código, o checksum funciona de uma forma única, você pode mudar um bit da imagem mas mesmo assim o código gerado pode ser totalmente diferente do anterior.

Figura 1: exemplo de funcionamento do checksum



Disponível em: < pt.wikipedia.org/wiki/Soma_de_verificação> Acesso em ago. 2017.

O nosso web service funcionara em 3 etapas:

1º etapa: ao tirar a foto com o aplicativo será gerado um código da imagem e guardado dentro da imagem

2º etapa: quando o usuário enviar a imagem para o web service ele vai retirar o código da imagem e gerar um segundo código

3º etapa: verificaremos se os 2 códigos são iguais, se conferir a imagem será armazenada e enviar o feedback para o aplicativo

Para montarmos tudo isso primeiro pensamos em usar a linguagem C++, mas encontrei dificuldade com a sintaxe e o código ficaria muito extenso além de pouco prático, e começamos a pesquisar linguagens que serviriam melhor, encontramos inúmeras linguagens que fariam o serviço, mas o ruby se destacou por estarmos trabalhando com imagem, a linguagem ruby é muito utilizada em formatação e manipulação de imagem e isso facilitou a pesquisa sobre o assunto além do código ficar mais prático.

RESULTADOS E DISCUSSÃO

Apesar de ainda estar em fase de desenvolvimento estamos muito otimistas, o web service vem resolvendo todos os problemas previstos e encontrados ao decorrer do projeto.

O método checksum já vem sendo testado e está funcionando, muito por conta de ser muito utilizado pelo mundo a fora facilitou o desenvolvimento e resolução dos problemas encontrados no caminho.

Mas não foi a única técnica encontrada, varias foram estudadas como:

Função Hash

Stegonographic

Algoritmos de codificação

Alteração no código da foto

CONSIDERAÇÕES FINAIS

Seguindo em frente com o projeto em uma segunda fase, espera-se realizar a aplicação do web servisse, assim como criar um banco de dados para usuários e arquivos já verificados, para assim manter estatísticas atualizadas.

REFERÊNCIAS

MICROSOFT. **How to compute and compare hash values by using Visual C#**. 2016. Disponível em: https://support.microsoft.com/en-us/help/307020/how-to-compute-and-compare-hash-values-by-using-visual-c#toc. Acessado em: 07 nov. 2016.

MICROSOFT. **HASHBYTES (Transact-SQL)**. 2016. Disponível em: https://docs.microsoft.com/en-us/sql/t-sql/functions/hashbytes-transact-sql. Acessado em: 09 nov. 2016.

MICROSOFT. **CHECKSUM (Transact-SQL).** 2016. Disponível em: https://docs.microsoft.com/en-us/sql/t-sql/functions/checksum-transact-sql. Acessado em: 22. nov. 2016.

PEDRINI, Hélio; SCHWARTZ, Willian Robson. **Análise de imagens digitais, princípios, algoritmos e aplicações**. São Paulo. Thomson Learning, 2008.

DALKE, Andrew. **PNG checksum.** 2013. Disponível em: http://www.dalkescientific.com/writings/diary/archive/2014/07/10/png_checksum .html. Acessado em: 07. nov. 2016.

RIVEST, Ronald L. **The MD5 Message-Digest Algorithm.** 1992. Disponível em: http://www.fastsum.com/rfc1321.php. Acessado em: 20 out. 2016.

CASTELLÓ, Thiago; VAZ, Verônica. **Tipos de Criptografia**. Disponível em: https://www.gta.ufrj.br/grad/07_1/ass-dig/TiposdeCriptografia.html. Acessado em: 20 out. 2016.

DAVID, Robin. **LSB-Steganography.** 2015. Disponível em: https://github.com/RobinDavid/LSB-Steganography. Acesso em: 22 nov. 2017.

CHAMPAKAMALA .B.S; PADMINI.K; RADHIKA .D. K. **Least Significant Bit algorithm for image steganography.** 2001. Disponível em: Https://www.cs.cf.ac.uk/Dave/Multimedia/node231.html. Acesso em 13 out. 2016.