# Protocolo para transferência parcial de conhecimento e sua aplicação à verificação segura de marcas d'água*

**Raphael Carlos Santos Machado**[2],
**Davidson Rodrigo Boccardo**[2],
**Vinícius Gusmão Pereira de Sá**[1],
**Jayme Luiz Szwarcfiter**[1,2]

[1]Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brasil

[2]Instituto Nacional de Metrologia, Qualidade e Tecnologia (INMETRO)
Duque de Caxias, RJ – Brasil

{rcmachado,drboccardo}@inmetro.gov.br,
vigusmao@dcc.ufrj.br, jayme@nce.ufrj.br

***Abstract.*** *Let $y = f(x)$ for some one-way function $f$. We present a simple algorithm that allows that the bits of $x$ are but partially exhibited in a demonstration, through a zero-knowledge proof scheme, that $x$ is indeed an element of the pre-image of $y$ under $f$. As an application, we show that it is possible to disclose a watermark embedded into a digital artifact without the need of revealing its location. The result is a secure verification protocol for software watermarking which does not increase the likelihood that an attacker is successful in a removal attack.*

***Resumo.*** *Seja $y = f(x)$ para uma função one-way $f$. Apresentamos um algoritmo simples que permite exibir a terceiros apenas parte dos bits de $x$ numa demonstração, por meio de um esquema de prova de conhecimento nulo, de que $x$ pertence de fato à pré-imagem de $y$ sob $f$. Como aplicação, mostramos que é possível exibir uma marca d'água embarcada em um artefato digital sem necessidade de revelar sua localização. O resultado é um protocolo seguro para verificação de marcas d'água de software que não aumenta a probabilidade de um atacante ser bem-sucedido em um ataque de remoção.*

## 1. Introdução

Em sua tese de doutorado, Joe Kilian [Kilian 1990] apresenta o seguinte problema. Bob deseja fatorar um número $n$ de 500 bits que, sabe-se, é o produto de cinco números primos de 100 bits. Alice conhece um dos fatores, denotado $q$, e está disposta a vender 25 de seus bits a Bob. Kilian propõe um método que possibilita a Alice comprovar que ela de fato conhece um dos fatores de $n$, e ainda permite que tal comprovação aconteça sobre *bits individuais* de $q$. O protocolo proposto por Kilian não apenas permite que sejam revelados

*Trabalho parcialmente financiado por CAPES, CNPq, FAPERJ, Pronametro 52600.017257/2013 e Eletrobrás DR/069/2012.

 

somente alguns dos bits de $q$, mas utiliza esquemas de *commitment*[1] individual dos bits de $q$ para garantir também que eles não serão revelados sem o consentimento de Alice. Finalmente, permite o emprego de *oblivious transfer* [Rabin 1981] de tal modo que a própria Alice desconheça o conjunto dos bits efetivamente revelados.

No presente trabalho, apresentamos um protocolo para um cenário simplificado de revelação de alguns dos bits de $q$, permitindo observar aspectos essenciais do que denominamos "transferência parcial de conhecimento". No cenário proposto, Alice não possui interesse financeiro sobre os bits a serem transferidos: Alice está disposta a revelar alguns dos bits de $q$ a quem quer que deseje conhecê-los. Por outro lado, Alice somente concorda em revelar um determinado subconjunto dos bits de $q$ — por convenção, assumiremos que Alice sempre revela os bits mais significativos de $q$, embora essa escolha seja arbitrária. O fato de que tal conjunto é pré-determinado dispensa o uso de *oblivious transfer*.

Pelo protocolo proposto, Alice é capaz de exibir os bits mais significativos de $q$, comprovando a quem possa interessar que, de fato, tratam-se de parte dos bits de um dos fatores de $n$. O protocolo é simples e intuitivo, e faz uso de reduções polinomiais e de provas de conhecimento nulo, baseando-se na Hipótese da Dificuldade da Fatoração (HDF). É fácil verificar que as técnicas propostas podem ser adaptadas a problemas clássicos notadamente difíceis, tais como logaritmo discreto.

Como aplicação do protocolo proposto, apresentamos um cenário de verificação segura de marcas d'água. A principal vantagem do uso de um protocolo de transferência parcial de conhecimento é a possibilidade de divulgar informações de autoria e propriedade, armazenadas exatamente nos bits a serem exibidos, sem que a necessidade de verificação eventual da marca d'água torne mais fácil sua remoção por parte de um atacante. Como a transferência de conhecimento é parcial, o atacante interessado em remover a marca d'água não disporá de informação suficiente para localizá-la dentro do artefato em que está embarcada, de modo que sua remoção permanecerá tão difícil após a verificação quanto antes dela.

O artigo está organizado da seguinte forma. Na Seção 2, apresentamos um protocolo que permite demonstrar que um conjunto de bits é o prefixo de um dos fatores de um número, sem que seja necessário, no entanto, exibir qualquer dos fatores desse número. Na Seção 3, descrevemos objetivos e conceitos básicos associados a marcas d'água digitais. Na Seção 4, mostramos como o protocolo de transmissão parcial de conhecimento pode ser adaptado à construção de um esquema de marcas d'água digitais que é resistente a ataques de remoção mesmo após a verificação de uma marca d'água embarcada em um artefato digital. Na Seção 5, apresentamos trabalhos relacionados, e, na Seção 6, nossas considerações finais.

## 2. Protocolo para transferência parcial de conhecimento

Dado um inteiro positivo $n$ que é o produto de dois números primos $p$ e $q$, queremos ser capazes de mostrar que uma dada sequência de bits $k$ corresponde aos bits mais significativos (ou *prefixo*) de $p$, sem revelar quaisquer dos fatores. O protocolo proposto é baseado, essencialmente, na aplicação de esquemas de prova de conhecimento nulo e

[1]Optamos por manter, neste texto, termos originais em inglês cujos equivalentes em língua portuguesa não se encontram ainda bem estabelecidos, como *commitment* de bits (e bits *committed*) e *oblivious transfer*.

 

em transformações polinomiais entre variantes do problema da fatoração de um inteiro e variantes do problema da satisfabilidade boolena.

### 2.1. Reduzindo EQUICOMPOSITE a SAT

Inicialmente, mostramos que o problema de determinar se um número é composto pode ser facilmente reduzido ao problema de se determinar se uma expressão lógica é satisfatível, problema esse conhecido como SAT [Schaefer 1978]. Mais precisamente, consideramos a variante EQUICOMPOSITE do problema de fatoração, em que se quer determinar se um inteiro $n$ pode ser escrito como o produto de dois fatores, cada um dos quais com no máximo $\lceil log_2(n)/2 \rceil$ bits.

**EQUICOMPOSITE**
**Entrada**: número binário $n$, com $\lceil log_2(n) \rceil$ bits.
**Saída**: SIM, se $n$ é o produto de dois números de bitsize até $\lceil log_2(n)/2 \rceil$;
NÃO, caso contrário.

Para tratar o problema EQUICOMPOSITE por meio de provas de conhecimento nulo, estudaremos a implementação de variantes da operação de multiplicação por meio de circuitos combinacionais, ou, equivalentemente, por meio de expressões lógicas envolvendo os bits dos operandos.

#### Produto de inteiros como uma função lógica

É sabido que a operação de produto de dois números binários pode ser descrita na forma de um circuito combinacional, de tal forma que cada dígito do resultado é uma expressão lógica sobre os dígitos dos operandos. Por questão de completude, revisamos brevemente a teoria sobre como construir um multiplicador binário.

**Somando bits.** É fácil implementar um circuito combinacional simples que recebe como entrada dois bits $A$ e $B$ (os operandos) e um terceiro bit $C_i$, o "vai um" (gerado por um somador em estágio anterior), e que retorna como saída o bit $S$ resultante da soma dos três bits recebidos e um novo bit $C_o$ de "vai um" (Figura 1).

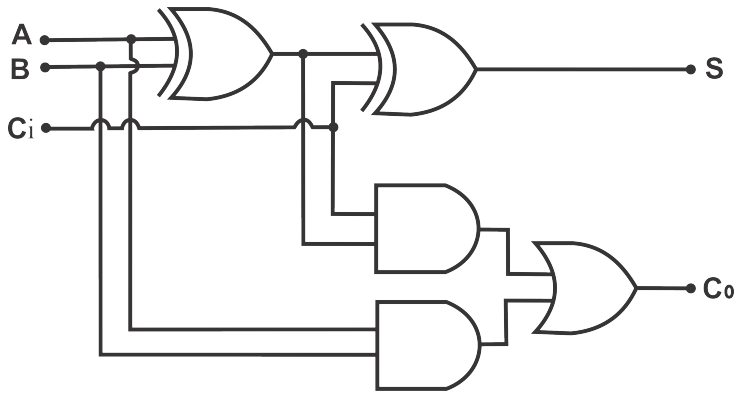


**Figura 1. Somador completo**

 

Observe que tanto o bit $S$ quanto o bit $C_o$ podem ser descritos como uma expressão lógica aplicada sobre os bits $A$, $B$ e $C_i$:

- $S = (A \oplus B) \oplus C_i$
- $C_o = (A \cdot B) + (C_i \cdot (A \oplus B))$

Naturalmente, o XOR ("ou exclusivo", representado por $\oplus$) pode ser substituído por operações OR ($+$) e AND ($\cdot$), de acordo com a fórmula $A \oplus B = \bar{A}B + A\bar{B}$.

**Encadeando somadores.** Para efetuar a soma de números binários com mais de um bit, basta encadear somadores completos, sempre enviando o "vai um" de saída de um estágio à entrada do estágio seguinte (Figura 2). Mais uma vez, cada um dos bits de saída

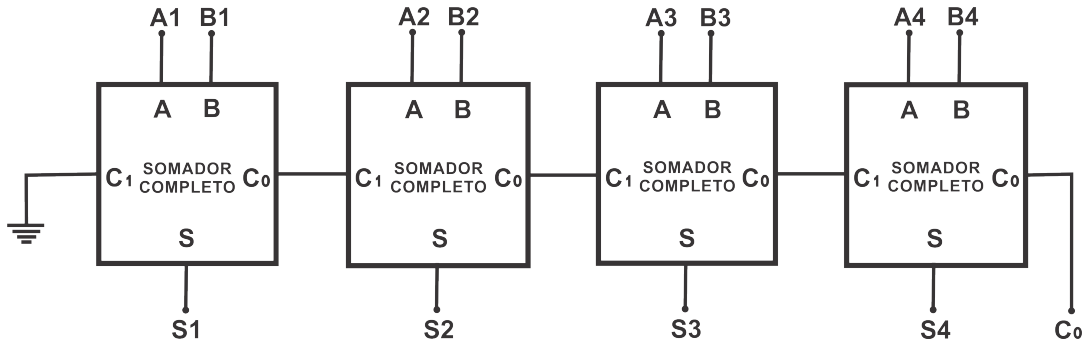


**Figura 2. Somador de quatro bits**

pode ser descrito como uma expressão lógica aplicada sobre os bits de entrada.

**Multiplicando por potências de dois.** A multiplicação por dois, em binário, pode ser executada como um simples deslocamento à esquerda, com a inclusão de um bit zero como dígito menos significativo da saída. Denotaremos o deslocamento à esquerda em $i$ bits (multiplicação por $2^i$) de um número binário $B$ por $B << i$.

**Obtendo o produto de dois números binários.** De maneira simplificada, a operação de multiplicação sobre binários pode ser entendida como uma série de adições e multiplicações por dois. Por exemplo, para multiplicar $A = A_3A_2A_1A_0$ por $B = B_3B_2B_1B_0$, começamos pelo bit mais à direita de um dos operandos, digamos, $A$. Se o bit $A_0$ é igual a 1, então, adicionamos o valor do outro operando, $B$, ao resultado $C$ (que é inicialmente zero); se o bit $A_0$ é igual a 0, então nenhum valor é adicionado. Para cada um dos bits consecutivos $A_i$ de $A$, se e somente se $A_i = 1$, efetuamos um deslocamento à esquerda de tamanho $i$ em $B$ (ou seja, multiplicamos $B$ por $2^i$) e o somamos ao resultado. O resultado, escrito como uma expressão lógica, equivale a $C = B \wedge A_0 + (B << 1) \wedge A_1 + (B << 2) \wedge A_2 + (B << 3) \wedge A_3$ (Figura 3).

**Construindo uma saída única.** Sabendo descrever o produto de dois números binários na forma de um circuito combinacional, é fácil adaptá-lo para um circuito modificado que possui um único bit de saída cujo valor é 1 se e somente se um determinado número $n$ é equicomposto. Para isso, basta adicionar portas NOT a cada saída do circuito multiplicador associada a um bit de $n$ que deve ser 0, e conectar todas as saídas a uma única porta AND.

Formalmente, um circuito UNI-MULT$(d, n)$, onde $d$ é um inteiro e $n$ é um número binário, é construído da forma esquematizada na Figura 4, com um circuito multiplicador

 

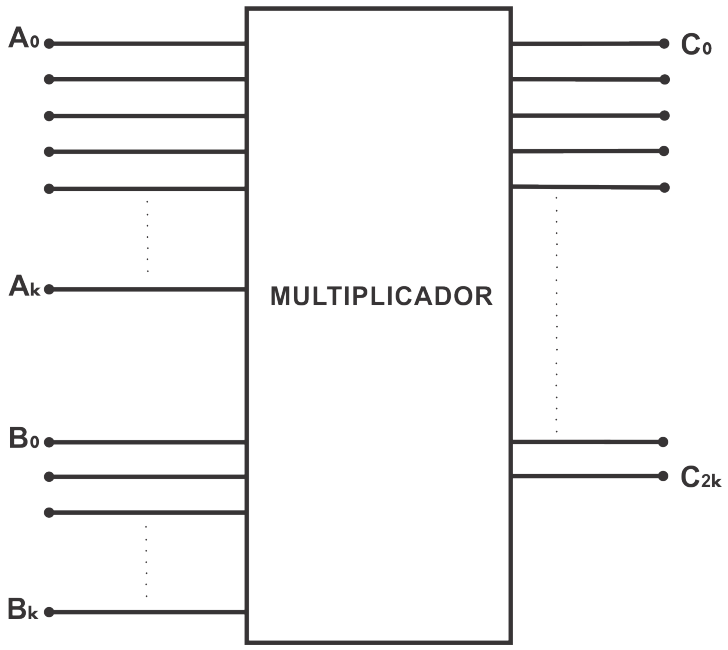


**Figura 3. Multiplicação**

de dois números binários de $d$ bits, com uma porta NOT após cada saída do multiplicador associada a um bit $0$ de $n$, e com uma porta AND conectando todas as $2d$ saídas (invertidas ou não). O Teorema 1, cuja prova é imediata, resume o resultado desejado.

**Teorema 1** *O circuito UNI-MULT*$(d, n)$ *retorna o bit* $1$ *se e somente se o número binário* $n$ *puder ser escrito como o produto de dois números binários de (até)* $d$ *bits.*

### 2.2. O problema PREFATOR

Consideramos, agora, o problema de se determinar se um número pode ser escrito como o produto de dois outros números, um dos quais possui um conjunto de bits cujos valores são previamente fixados. Mais precisamente, considere o seguinte problema de decisão, ao qual denominamos PREFATOR.

**PREFATOR**
**Entrada**: números binários $n$ e $k$.
**Saída**: SIM, se $n$ é equicomposto e possui um fator do qual $k$ é prefixo;
NÃO, caso contrário.

Sabendo-se reduzir EQUICOMPOSITE a SAT, torna-se simples entender como reduzir o problema PREFATOR a SAT. De fato, nosso objetivo é determinar se um número é o produto de dois outros números, um dos quais começando por um conjunto de bits pré-fixados. Nossa estratégia é construir um circuito semelhante ao da Figura 4, mas com a "transferência" de alguns bits da entrada diretamente para o último estágio do circuito, que recebe uma porta AND adicional conforme o diagrama da Figura 5.

Formalmente, um circuito PRE-MULT$(d, n, k)$, onde $d$ é inteiro e $n$ e $k$ são números binários, é construído da forma esquematizada na Figura 5. Inicialmente, toma-

 

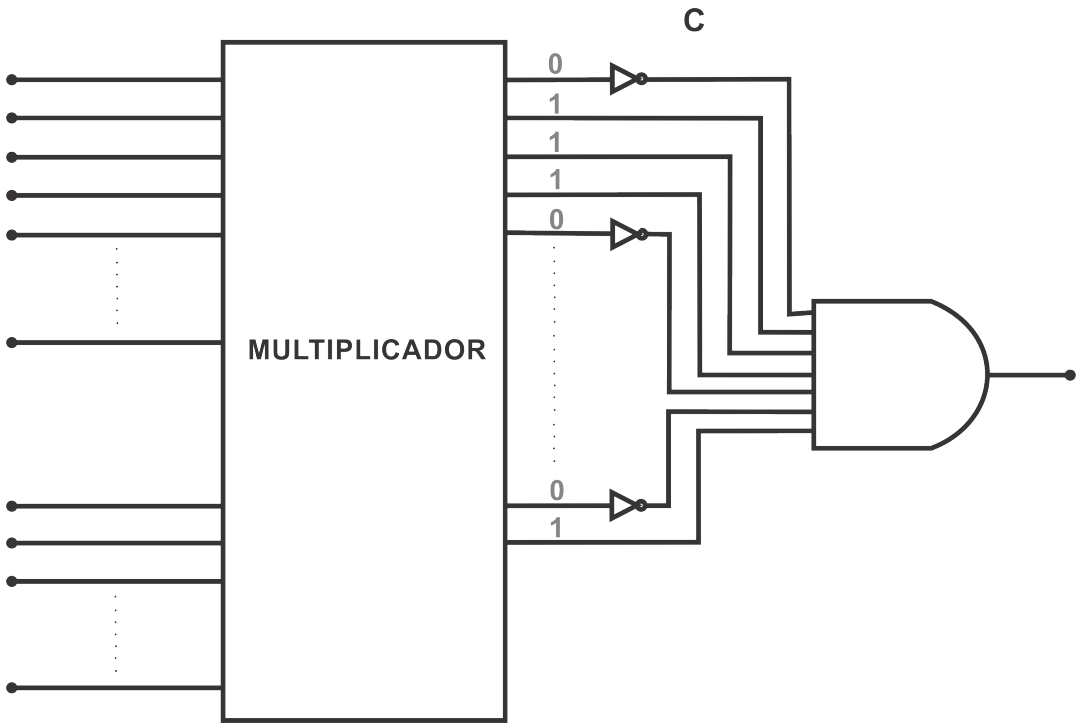


**Figura 4. UNI-MULT: fixando os bits da saída com um AND final**

se um circuito UNI-MULT$(d, n)$. Para cada bit de entrada do circuito UNI-MULT$(d, n)$ associado a um bit de $k$, cria-se uma derivação, que será conectada a uma porta NOT se esse bit for $0$ em $k$. As derivações são todas conectadas a uma porta AND, assim como o bit de saída do circuito UNI-MULT$(d, n)$. O Teorema 2 resume o que o circuito PRE-MULT permite responder.

**Teorema 2** *O circuito PRE-MULT*$(d, n, k)$ *retorna o bit* $1$ *se e somente se o número binário* $n$ *puder ser escrito como o produto de dois números binários de até* $d$ *bits, um dos quais possuindo* $k$ *como prefixo.*

### 2.3. Convertendo para a forma normal conjuntiva

O leitor observará que, novamente, a saída do circuito PRE-MULT$(d, n, k)$ é uma função lógica sobre os bits de entrada. No entanto, para utilizarmos o arcabouço da teoria da complexidade computacional e suas reduções polinomiais, é necessário dispor de uma expressão lógica na forma normal conjuntiva. Felizmente, as transformações de Tseitin [Tseitin 1983] permitem construir, a partir de uma expressão lógica $\sigma$ qualquer, uma nova expressão lógica $\sigma'$ cujo tamanho é linear no tamanho de $\sigma$. Além disso, a transformação é executada em tempo linear no tamanho de $\sigma$.

### 2.4. Utilizando provas de conhecimento nulo

Sabendo reduzir o problema PREFATOR a SAT, podemos de forma simples recorrer a provas de conhecimento nulo por meio de reduções polinomiais. Podemos, por exemplo, reduzir a instância SAT a uma instância de 3-COLORAÇÃO em tempo polinomial [Karp 1972], para, então, utilizar um esquema clássico de provas de conhecimento nulo para este último problema [Goldwasser et al. 1985].

 

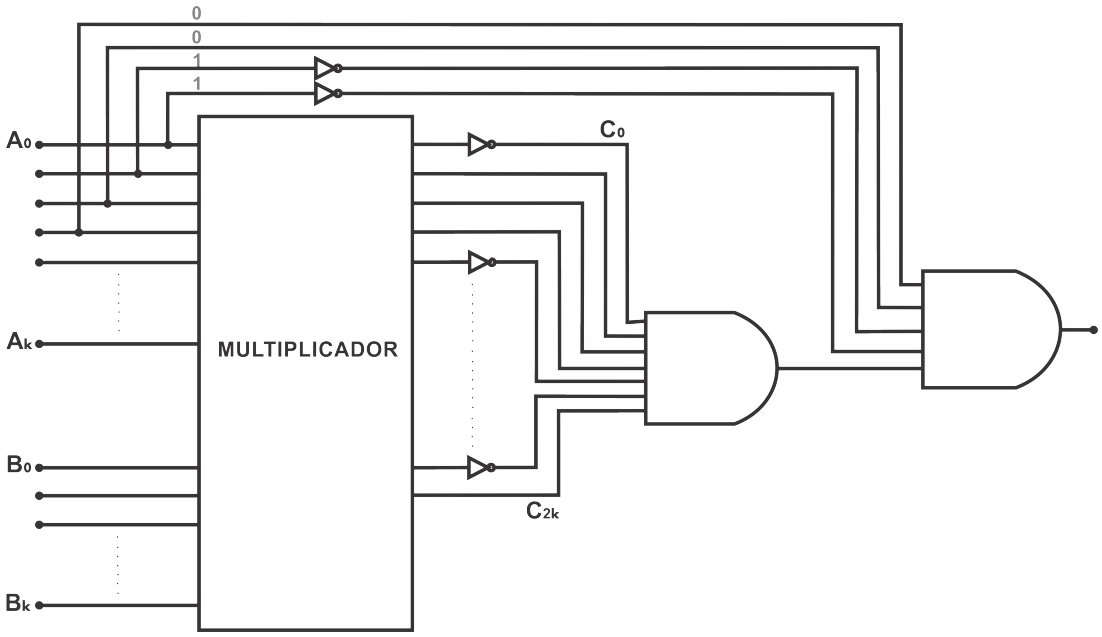


**Figura 5. PRE-MULT: fixando os quatro primeiro bits de $A$ em "1100"**

## 3. Marcas d'água

A partir deste ponto, mostraremos como o protocolo de divulgação parcial de conhecimento que estamos propondo pode ser utilizado no contexto de um esquema de marcas d'água digitais, com o fim de demonstrar autoria ou propriedade de um artefato digital. De fato, esta é a motivação original para a construção dos protocolos de divulgação parcial de conhecimento apresentados neste artigo.

Esquemas de marca d'água são técnicas que permitem embarcar uma informação, muitas vezes de forma oculta, em um artefato qualquer, de tal maneira que, uma vez inserida, torna-se inviável remover tal informação por um agente mal intencionado. Estas informações embarcadas são utilizadas, em geral, para identificar autoria ou propriedade, ou ainda para conferir identidade única a um artefato. Com o advento do computador, tornou-se desejável embarcar informações em artefatos digitais, tais como arquivos de texto, imagem e áudio, além de programas de computador, entre outros.

Marcas d'água baseiam-se na imersão de uma informação — denominada *conteúdo* — em um artefato digital — denominado *hospedeiro* — de tal forma que o artefato digital obtido — ao qual denominamos *produto* — é, para todos os efeitos, equivalente ao hospedeiro.

Essencialmente, um esquema de marcas d'água é definido por meio de dois algoritmos. O primeiro deles, denominado *embarcador*, é um algoritmo que recebe como entrada um hospedeiro $u$ e um conteúdo $m$ a ser embarcado, e retorna como saída um produto $\tilde{u}$ equivalente ao hospedeiro, mas que contém embarcada a informação $m$. Adicionalmente, precisamos de um algoritmo *extrator* que recebe como entrada um artefato digital (produto) $\tilde{u}$, o qual possui embarcada uma informação-conteúdo $m$, e retorna como saída essa informação $m$.

As propriedades essenciais de uma marca d'água são geralmente furtividade, re-

 

siliência e verificabilidade. Furtividade diz respeito à propriedade de um artefato com marca d'água ser indistinguível de um artefato sem marca d'água. Resiliência diz respeito à propriedade de uma marca d'água ser de difícil remoção ou modificação, sob pena de comprometer a própria funcionalidade do artefato. Verificabilidade diz respeito à propriedade de uma marca d'água poder ser exibida a terceiros, de modo a comprovar a autoria ou a propriedade de um artefato.

A maior dificuldade em construir um esquema de marca d'água para artefatos digitais advém do fato de que tanto a informação embarcada quanto o artefato hospedeiro são, em última análise, sequências de bits, portanto, facilmente manipuláveis. Desta forma, a dificuldade em remover uma marca d'água tem se baseado na dificuldade em localizar a marca d'água [Bento et al. 2013]. Tal estratégia, no entanto, cria dificuldades quando é necessário exibir uma marca d'água, por exemplo, para comprovar a autoria de uma obra: uma vez que se conhece a localização da marca d'água, torna-se fácil removê-la.

Embora possa haver interesse em marcas d'água que necessitem ser exibidas somente uma vez — protocolos de fingerprinting são um exemplo, em que cada marca d'água identifica um único artefato, sendo embarcada somente naquele artefato, e bastando uma única verificação da marca d'água, em tribunal, para imputar culpa a quem houver feito utilização indevida do produto —, é desejável que seja possível exibir uma marca d'água sem que isto comprometa sua posterior resiliência a ataques. Denominamos *verificável* um esquema de marcas d'água que permita a verificação de uma marca d'água sem que isto reduza a dificuldade em removê-la. Note que a propriedade de verificabilidade está associada não somente à concepção da marca d'água, mas a forma de utilizá-la (verificá-la) através de um protocolo.

O problema é que o requisito de verificabilidade da marca d'água parece, de certa forma, incompatível com o requisito de resiliência: uma vez que se revela a localização da marca d'água no momento de sua verificação, desfaz-se a dificuldade, antes existente, de desabilitá-la por meio de ataques de remoção ou distorção. Mostramos a seguir como utilizar o protocolo proposto para transferência parcial de conhecimento de forma a exibir informações embarcadas em um artefato digital sem que seja necessário revelar sua localização.

## 4. Um esquema seguro de verificação de marcas d'água

### 4.1. Inserindo a marca d'água

Algumas das principais técnicas para se embarcar uma marca d'água em um artefato digital baseiam-se na possibilidade de se modificar imperceptivelmente o artefato, de forma que o produto final seja, para todos os efeitos, semanticamente equivalente ao original, e com a propriedade extra de permitir a obtenção de uma informação nele instalada de forma surreptícia. Vejamos alguns exemplos simples de classes de equivalência que podem ser exploradas.

**Texto.** Considere o campo de aplicação "língua portuguesa" e o conjunto dos textos escritos em português. Podemos dizer que dois textos são semanticamente equivalentes se são idênticos, a menos de substituições entre as palavras "entretanto" e "todavia".

Queria ir à praia; entretanto, choveu. $\equiv$ Queria ir à praia; todavia, choveu.

 

Uma outra possível classe de equivalência semântica é aquela em que períodos consecutivos são separados por "ponto final" ou por "ponto-e-vírgula".

$$\text{Penso. Logo, existo.} \equiv \text{Penso; logo, existo.}$$

Uma marca d'água que codifique um binário num texto suficientemente grande em língua portuguesa poderia utilizar a sequência de "entretanto" e "todavia" para codificar, respectivamente, os bits "1" e "0". Ou a contagem de ";" poderia corresponder ao inteiro que se deseja embarcar, ou diversas outras formas de se explorar modificações sintáticas que preservam a semântica.

**Imagem digital.** Considere o campo de aplicação "imagens digitais" e o conjunto dos arquivos bitmap em que cada bit é codificado como uma tripla que define a intensidade de vermelho, verde e azul (digamos, 0 a 255, cada intensidade). Uma possível classe de equivalência semântica é aquela em que intensidades ímpares podem ser reduzidas de uma unidade e intensidades pares podem ser aumentadas de uma unidade.

$$\text{... (173,189,12) ...} \equiv \text{... (172,189,13) ...}$$

Uma marca digital poderia ser embarcada através da modificação apropriada dos códigos de cor em uma sequência de pixels iniciado em algum ponto específico da imagem. Códigos ímpares estariam, por exemplo, associados a bits "1", e códigos pares, a bits "0".

**Programa de computador.** Considere o campo de aplicação "programas de computador" e o conjunto dos códigos em linguagem C. Uma possível classe de equivalência semântica é aquela em que as saídas dos programas são as mesmas (desconsiderando tempo de execução ou uso de memória).

$$\text{... printf(”Hello World”) ...} \equiv \text{... if(1)\{printf(“Hello World”)\}else\{printf(“Buraco”)\} ...}$$

Aqui, mais uma vez, pode-se utilizar a forma (apropriadamente modificada) para embarcar o conteúdo desejado.

### 4.2. Verificabilidade

Nos exemplos clássicos de marca d'água embarcadas em artefatos físicos concretos, a marca d'água é visível e verificável por qualquer um que tenha acesso ao artefato que a contém. Isto é possível pois a dificuldade em remover a marca d'água é baseada no processo físico de imersão da marca d'água. Como exemplo, considere a dificuldade em remover de um documento uma marca em baixo relevo.

A necessidade de embarcar marcas d'água em artefatos digitais leva à necessidade de uma propriedade adicional à qual denominamos *verificabilidade*, que é a possibilidade de se poder atestar a presença da marca d'água sem que isto torne mais fácil, a um atacante, removê-la. Por se tratarem de artefatos digitais, torna-se relativamente fácil modificar tais artefatos. Assim, uma marca d'água concebida ingenuamente pode se tornar facilmente removível caso um atacante tenha conhecimento de sua forma e posição. Os seguintes exemplos ilustram o argumento:

 

**Exemplo de marca d'água em texto.** Considere uma marca d'água em um texto codificado na forma de uma frase cuidadosamente construída para codificar a marca d'água. Para um atacante que não tenha conhecimento da posição desta frase especial dentro do texto, a tarefa de remover a marca d'água é bastante difícil. No entanto, uma vez que o autor do texto exiba a marca d'água — revelando a frase — a remoção da marca d'água passa a ser tarefa trivial.

**Exemplo de marca d'água em programa de computador.** Considere uma marca d'água em um programa de computador codificada na forma de um subgrafo do grafo de fluxo de controle deste programa. Para um atacante que não tenha conhecimento da posição do subgrafo dentro do grafo do programa, a tarefa de remover a marca d'água é bastante difícil, ainda que o atacante conheça o subgrafo que a codifica, pois estaria diante de um problema difícil (e clássico) em Teoria dos Grafos, qual seja o do isomorfismo de subgrafos. No entanto, uma vez que o agente embarcador da marca d'água a exiba, revelando sua localização, a remoção da marca d'água passa a ser tarefa fácil.

Para poder demonstrar a autoria ou a propriedade de um artefato digital com base no algoritmo descrito na Seção 2, utilizaremos a seguinte estratégia. Primeiramente, codificaremos a informação de autoria ou propriedade na forma de um número binário $k$. Selecionamos dois números primos $p$ e $q$, um dos quais possui exatamente $k$ como seu conjunto de bits mais significativos (prefixo), e computamos o produto $n$ dos números $p$ e $q$. O produto resultante será imerso no artefato digital a ser marcado, nele aparecendo na forma de uma *substring*, isto é, subsequência de bits (mais precisamente, aparecendo como uma subsequência da sequência de bits obtida a partir do artefato digital por meio da execução do algoritmo extrator). A motivação para esta estratégia é o fato de podermos exibir $k$ sem ser necessário revelar a localização de $n$.

Consideraremos agora uma pequena variação da definição do algoritmo extrator, ao qual denominamos algoritmo *pré-extrator*. O algoritmo pré-extrator, ao invés de retornar exatamente a marca d'água (previamente embarcada por meio do algoritmo embarcador), retorna uma sequência de bits — possivelmente longa — que contém a marca d'água como substring. Mais precisamente, a substring em questão será, como vimos, o produto de dois números primos, um dos quais possuindo a marca d'água como prefixo.

### 4.3. O problema SUBSTRING-PREFATOR

Considere o seguinte problema de decisão.

**SUBSTRING-PREFATOR**
**Entrada**: números binários $d$ e $k$, inteiro $N$.
**Saída**: SIM, se existe substring $n$ de $d$, com $N$ bits, tal que $n$ é equicomposto e um de seus fatores possui $k$ como prefixo;
NÃO, caso contrário.

É fácil ver que o problema SUBSTRING-PREFATOR também pode ser reduzido a SAT. De fato, basta construir um circuito PRE-MULT (idêntico ao construído para o problema PREFATOR) para cada substring de tamanho $N$, e aplicar um grande OR às saídas de cada um dos $bitsize(d) - N + 1$ circuitos.

 

### 4.4. Gerando a marca d'água

A geração da marca d'água a ser embarcada é um processo simples. Dada uma informação $m$ a ser embarcada, basta gerar dois números primos aleatórios $p$ e $q$ de mesmo bitsize, de tal forma que $m$ é prefixo de $p$, e computar o produto $n = p \cdot q$, sequência de bits que será, de fato, embarcada no artefato digital. A geração de $q$ segue os métodos tradicionais de sorteio de número aleatório seguido de teste de primalidade (por exemplo, Miller-Rabin) até que se obtenha um número primo. A geração de $p$ segue uma abordagem ligeiramente modificada: sorteia-se um número aleatório, mas concatena-se o número sorteado à direita de $m$ para, na sequência, testar-se a primalidade do número obtido — e o processo é repetido até obter-se um número primo.

### 4.5. Inserindo a marca d'água

O processo de inserção de marca d'água tem por objetivo modificar o artefato digital a ser marcado, fazendo com que a sequência de bits $n = p \cdot q$ apareça como uma substring da string retornada pelo programa extrator. Na prática, o processo exato de inserção — e da consequente extração — irá depender do tipo de artefato digital a ser marcado. No caso de um arquivo texto, por exemplo, o processo irá depender apenas do esquema de codificação utilizado para embarcar informações no texto. Na Seção 3, por exemplo, descrevermos um esquema em que a sequência de palavras "todavia" e "entretanto" determinaria uma informação codificada. Neste caso, bastaria selecionar uma sequência destas palavras, substituindo-as apropriadamente para conter os bits construídos pelo protocolo. Exemplos análogos poderiam ser construídos para outros campos de aplicação, e a especificação detalhada de cada caso está fora do escopo do presente trabalho.

### 4.6. Verificando a marca d'água

O processo de verificação da marca d'água contempla as seguintes etapas:

1. Extração da sequência de bits associada ao artefato digital — sequência esta que pode ser muito longa, mas que contém, como substring, $n = p \cdot q$.
2. Transformação da sequência gerada em uma instância de SAT, e daí para uma instância de 3-COLORAÇÃO.
3. Utilização de esquema de prova de conhecimento nulo para demonstrar que o grafo obtido no passo anterior é, de fato, 3-colorível.

A extração da sequência de bits a que nos referimos no passo inicial pode ser feita por meio de algoritmos específicos para esse fim, os quais devem ser definidos para cada campo de aplicação e seus correspondentes artefatos digitais. A transformação para uma instância de SAT é exatamente o algoritmo descrito na Seção 2, enquanto a redução polinomial de SAT para 3-COLORAÇÃO é um resultado clássico da literatura [Karp 1972]. Um esquema de prova interativa de conhecimento nulo para 3-COLORAÇÃO é descrito em [Goldwasser et al. 1985] e pode ser utilizado no último passo para exibir a marca d'água sem apresentar $n$ ou quaisquer de seus fatores.

## 5. Trabalhos relacionados

Marcas d'água em artefatos digitais têm sido uma área ativa de pesquisa desde a década de 90, com uma grande quantidade de trabalhos e diferentes técnicas para embarcar marcas d'água em áudio, vídeo, imagem e software, como sumariadas em

 

[Collberg and Nagra 2009]. Técnicas de marca d'água podem ser classificadas com base nas técnicas de embarcação e extração, e em sua natureza estática ou dinâmica. No entanto, marcas d'água dinâmicas são específicas para software. Tipicamente, marcas d'água em software são embarcadas por meio de substituição de código, reordenação de código, alocação de registros, grafos estáticos ou dinâmicos, interpretação abstrata e utilização de predicados opacos. Um detalhamento dessas técnicas pode ser encontrado nos trabalhos de [Zhu et al. 2005, Hamilton and Danicic 2011].

Uma marca d'água de software é estática caso a marca d'água esteja contida no segmento de código ou de dados do software; ou dinâmica, caso esteja em algum estado de execução do software, ou seja, construída durante a execução do programa. Marcas d'água dinâmicas apresentam vantagens de furtividade e resiliência em relação a marcas d'água estáticas [Collberg and Nagra 2009]. Uma marca d'água dinâmica é construída durante a execução do software por meio de instruções intercaladas a instruções da aplicação, fazendo com que a codificação e a localização da marca seja desconhecida para um adversário. Isso adiciona furtividade à marca d'água, uma vez que sua localização é distinta a cada execução do programa. Isto aumenta, consequentemente, sua resiliência a ataques, uma vez que dificulta sua detecção e adulteração/remoção.

Diversos trabalhos discutem a aplicação de provas de conhecimento nulo para áudio, vídeo, imagem [Craver 1999, Adelsbach et al. 2003, Adelsbach and Sadeghi 2001]. Para nosso conhecimento, existe apenas um trabalho que aplica prova de conhecimento nulo para marcas d'água de software [Venkatachalam 2005]. A proposta de Venkatachalam, no entanto, evita que partes da marca d'água sejam reveladas durante um processo de verificação, enquanto o presente trabalho evita a apresentação de *qualquer parte* da marca d'água. Como resultado, temos um protocolo de verificação mais robusto diante de ataques de remoção. Adicionalmente, cabe destacar que o trabalho de [Venkatachalam 2005] baseia-se em um esquema clássico de prova de conhecimento nulo para o problema dos resíduos quadráticos; nosso trabalho, por outro lado, por meio de reduções polinomiais de circuitos para problemas lógicos, cria um arcabouço que permite a aplicação de qualquer problema computacionalmente difícil ao problema de verificação segura de marca d'águas.

## 6. Considerações finais

No presente trabalho, apresentamos um protocolo simplificado que busca responder a uma pergunta clássica de Joe Kilian [Kilian 1990] sobre a possibilidade de um transferência parcial de conhecimento a respeito da solução de um problema. O protocolo faz uso de ferramentas clássicas tais como provas de conhecimento nulo e reduções polinomiais. Concluímos o trabalho com uma série de observações a respeito dos métodos desenvolvidos e questões relacionadas.

**Exponenciação discreta e outros problemas.** Embora tenhamos nos concentrado na apresentação de métodos de transferência parcial de conhecimento a respeito dos fatores de um número, essencialmente qualquer problema em NP é adequado para a adaptação das técnicas aqui descritas. Um problema para o qual cabe destaque é o da exponenciação discreta, dada a sua fácil representação por circuitos combinacionais — e portanto, na-

 

tural redução ao SAT — e dada a sua vasta aplicação à Segurança da Informação. Na prática, os métodos aqui propostos podem ser facilmente adaptados para mostrar que o logaritmo discreto de um número possui um determinado prefixo, e é razoável que protocolos similares possam ser desenvolvidos para qualquer dos problemas considerados "computacionalmente difíceis" que fundamentam os atuais algoritmos criptográficos.

**Furtividade da marca d'água.** Uma característica que pode destacar a marca d'água dentro do artefato marcado é o fato de que ela é uma substring com propriedades bastante especiais — pelo menos no caso da marca d'água baseada no produto de primos. De fato, a substring embarcada possuirá a propriedade bastante particular de ser o produto de dois números primos grandes, enquanto a maioria dos números binários possui fatores pequenos. Isso possibilita a construção de um algoritmo simples para determinar um pequeno conjunto de substrings candidatas a marca d'água: basta procurar por substrings que representem binários sem fatores pequenos, onde a definição exata de "pequenos" poderá variar de acordo com a disposição do atacante em executar algoritmos de fatoração.

Uma saída para contornar a fraqueza acima é lançar mão de marcas d'água de tamanhos variáveis, de tal maneira que o atacante não saiba o tamanho exato da substring que deverá procurar — apenas um limite inferior. Uma outra possível saída é o uso de exponenciação discreta como a "função one-way" nos protocolos propostos.

**Plágio.** Marca d'água não protege contra plágio. Se um escritor mal-intencionado, após ler um livro, resolver rescrevê-lo inteiramente com suas próprias palavras, tratar-se-á, a princípio, de uma nova obra. A idéia da marca d'água é que, caso este escritor reescreva apenas trechos do livro, provavelmente, a marca d'água ficará intacta. Ou seja, o esforço para remover a marca d'água será tão grande quanto o de plagiar toda a obra. O mesmo vale para um programa de computador e um atacante que entenda toda a sua lógica e rescreva inteiramente seu código.

**Hipótese Forte da Dificuldade da Fatoração.** Observe que o método descrito no presente trabalho baseia-se, na verdade, numa versão modificada — de fato, ligeiramente fortalecida — da Hipótese da Dificuldade da Fatoração (HDF). A clássica HDF pressupõe que a multiplicação de números primos aleatórios é uma função one-way. Por outro lado, no protocolo proposto, um dos números primos não é perfeitamente aleatório. De fato, os bits mais significativos de um dos números primos foram obtidos deterministicamente. Assumimos que esta versão modificada do produto de primos — à qual denominamos Hipótese Forte da Dificuldade da Fatoração — também é uma função one-way. Ainda que seja perfeitamente plausível assumirmos a HFDF (e mesmo, possivelmente, prová-la), seria interessante dispor de um protocolo que se baseasse tão somente em premissas clássicas, como é o caso da HDF.

## Referências

Adelsbach, A., Katzenbeisser, S., and Sadeghi, A.-R. (2003). Watermark detection with zero-knowledge disclosure. *Multimedia Syst.*, 9(3):266–278.

 

Adelsbach, A. and Sadeghi, A.-R. (2001). Zero-knowledge watermark detection and proof of ownership. In Moskowitz, I. S., editor, *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 273–288. Springer.

Bento, L. M. S., Boccardo, D. R., Machado, R. C. S., de Sá, V. G. P., and Szwarcfiter, J. L. (2013). Towards a provably resilient scheme for graph-based watermarking. In Brandstädt, A., Jansen, K., and Reischuk, R., editors, *WG*, volume 8165 of *Lecture Notes in Computer Science*, pages 50–63. Springer.

Collberg, C. and Nagra, J. (2009). *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Professional, 1st edition.

Craver, S. (1999). Zero knowledge watermark detection. In Pfitzmann, A., editor, *Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 101–116. Springer.

Goldwasser, S., Micali, S., and Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA. ACM.

Hamilton, J. and Danicic, S. (2011). A survey of static software watermarking. *Proc. World Congress on Internet Security*, pages 100–107.

Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*. Plenum Press, New York.

Kilian, J. (1990). *Uses of randomness in algorithms and protocols.* MIT Press.

Rabin, M. O. (1981). How to exchange secrets with oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory.

Schaefer, T. J. (1978). The complexity of satisfiability problems. In *10th annual ACM symposium on Theory of Computing*, pages 216–226. ACM.

Tseitin, G. (1983). On the complexity of derivation in propositional calculus. In Siekmann, J. and Wrightson, G., editors, *Automation of Reasoning*, Symbolic Computation, pages 466–483. Springer Berlin Heidelberg.

Venkatachalam, B. (2005). Software watermarking as a proof of identity: A study of zero knowledge proof based software watermarking. In Barni, M., Cox, I. J., Kalker, T., and Kim, H. J., editors, *IWDW*, volume 3710 of *Lecture Notes in Computer Science*, pages 299–312. Springer.

Zhu, W., Thomborson, C. D., and Wang, F.-Y. (2005). A survey of software watermarking. In Kantor, P. B., Muresan, G., Roberts, F. S., Zeng, D. D., Wang, F.-Y., Chen, H., and Merkle, R. C., editors, *Proc. IEEE Int'l Conference on Intelligence and Security Informatics*, volume 3495 of *ISI'05*, pages 454–458. Springer.

 