Síntese de uma Nova Política de Ofuscação que Garanta Diferentes Níveis de Privacidade e Utilidade

João Manoel Costa Cardoso * **Marcos Vicente de Brito Moreira** *
Lilian Kawakami Carvalho *

* *Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, RJ, e-mails: joao.cardoso@coppe.ufrj.br, moreira@dee.ufrj.br, lilian.carvalho@poli.ufrj.br.*

Abstract: The world is increasingly connected through computer networks, which makes it necessary to develop techniques capable of guaranteeing the privacy and utility of the transmitted information. In this work we consider the problem of obfuscation of discrete event systems that guarantees different levels of both privacy and utility of the generated information. The system is modeled as a labeled transition system that has secret states. The first goal is to keep the information generated by the system useful so that legitimate users can use it for decision making. At the same time, the second goal is to hide critical information about the secret states of the system. However, unlike the previous approaches presented in the literature, we consider the case in which it is possible for the obfuscator to report the secret states. Thus, this approach increases the number of systems capable of being obfuscated and, since it keeps the public view of the system with the same behavior as the original system, it makes difficult to notice the presence of the obfuscator. To perform the synthesis of the obfuscator we used the software SynthSMV.

Resumo: O mundo está cada vez mais conectado por redes de computadores, isso faz com que seja necessário o desenvolvimento de técnicas capazes de garantir a privacidade e a utilidade da informação transmitida. Neste trabalho consideramos o problema de ofuscação de sistemas a eventos discretos que garanta diferentes níveis tanto de privacidade quanto de utilidade da informação gerada. O sistema é modelado como um sistema de transição rotulado que apresenta estados considerados secretos. O primeiro objetivo é manter a informação gerada pelo sistema útil para que usuários legítimos possam utilizar da mesma para tomada de decisões. Ao mesmo tempo, o segundo objetivo é esconder informações críticas a respeito dos estados secretos do sistema, porém, diferentemente das abordagens anteriores apresentadas na literatura, consideramos o caso em que é possível para o ofuscador reportar a passagem por estados secretos. Desta forma, além de aumentar o número de sistemas capazes de serem ofuscados, esta abordagem dificulta notar a presença do ofuscador, uma vez que mantém a visão pública do sistema com o mesmo comportamento do sistema original. Para realizar a síntese do ofuscador utilizamos o software SynthSMV.

Keywords: Security, Obfuscation, Privacy, Discrete event systems, Supervisory control, Temporal logic.

Palavras-chaves: Segurança, Ofuscação, Privacidade, Sistemas a eventos discretos, Controle supervisório, Lógica temporal.

1. INTRODUÇÃO

Muito se tem discutido, nos últimos anos, acerca do tema Indústria 4.0 que remete à quarta revolução industrial e representa uma tendência atual pela utilização de tecnologias de automação e conectividade na manufatura. De acordo com Gilchrist (2016), a Indústria 4.0 busca a conectividade de máquinas, fábricas e os complexos de armazenamento através de redes globais que permitirão um controle inteligente da produção e estoque através do compartilhamento de informações. Uma tendência da Indústria 4.0 é a utilização de modelos computacionais para a representação e controle de sistemas físicos, os chamados Sistemas Ciber-Físicos, sistemas capazes de unir componentes físicos e cibernéticos através de tecnologias como *IOT*, *Big Data* e *Cloud Networks* (Alguliyev et al., 2018), permitindo assim que as empresas consigam representar a realidade do mundo físico em um ambiente virtual. Por meio dos Sistemas Ciber-Físicos é possível realizar simulações, testes de equipamentos, auxílio na manutenção preditiva, entre outros. Porém, a utilização desses sistemas acarreta diversos problemas de segurança

⋆ O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001

e confidencialidade, já que muitas informações cruciais são compartilhadas por redes como a internet.

O comportamento dos sistemas em análise neste trabalho serão modelados como sistema a eventos discretos (SEDs). Consideramos que cada evento gerado pelo sistema é enviado por redes de comunicação para usuários externos, o que pode acarretar diversos problemas relacionados à segurança e confiabilidade da informação gerada. Desta forma, diversos trabalhos abordam o tema de segurança contra usuários mal-intencionados, como os trabalhos de Carvalho et al. (2018), que considera o problema de detectar e prevenir ataques em que atuadores são acionados com o intuito de causar danos ao sistema, e Lima et al. (2019), que aborda a detecção de intrusos em Sistemas Ciber-Físicos para prevenir danos causados por ataques do tipo *man-in-the-middle*. Outro trabalho que considera ataque em sensores é Alves et al. (2022) que utiliza a teoria de controle supervisório para garantir o comportamento legal do sistema, considerando a ocorrência de um conjunto de ataques conhecidos. Todos os trabalhos citados tentam de alguma forma identificar o tipo de ataque ou o conjunto de ataques.

Outra técnica utilizada para a segurança de sistemas a eventos discretos é a ofuscação, cujo principal objetivo é esconder alguma informação considerada relevante de agentes externos mal-intencionados, garantindo a privacidade e utilidade da informação. A utilidade da informação gerada pelo ofuscador depende das características individuais de cada sistema a ser ofuscado. Em sistemas baseados em geolocalização, o conceito de utilidade pode estar relacionado com a distância entre a localização real do sistema e a localização reportada pelo ofuscador para o meio externo. As observações geradas a partir dos eventos reportados pelo ofuscador devem continuar fazendo sentido para os usuários legítimos. Algumas sequências de eventos reportadas pelo ofuscador podem fazer com que a utilidade da informação transmitida se perca ou até mesmo acarrete situação em que o ofuscador não reporte a ocorrência de nenhum evento. Como não é possível diferenciar um observador legítimo de um ilegítimo, é importante desenvolver abordagens de ofuscação que consigam preservar tanto a privacidade quanto a utilidade da informação reportada.

Em Duckham and Kulik (2005) é estabelecido uma estrutura de localização baseado em grafos para tratar casos de ofuscação para serviços. Essa estrutura busca equilibrar a necessidade de serviços de alta qualidade de informação com a necessidade de privacidade de localização. Para isso, é criada uma região de ofuscação, gerando uma incerteza sobre a localização exata do sistema, mas garante que a informação gerada continue válida. Mais recentemente, Wu et al. (2018) foi o primeiro trabalho a considerar o caso de sintetizar um ofuscador visando garantir que a utilidade da informação entregue esteja dentro de um limite desejado. Duas abordagens para solucionar o problema de ofuscação são apresentadas. A primeira modela o sistema através de autômatos e utiliza funções de edição que conseguem adicionar, deletar ou alterar a ocorrência de eventos para realizar a ofuscação do sistema e garantir a utilidade da informação entregue. A segunda modela o sistema como um sistema de transição rotulado e utiliza o *software* SynthSMV para sintetizar o ofuscador através de especificações temporais que garantem a utilidade da informação gerada. Por sua vez, Góes et al. (2018), também no contexto de ofuscação, apresenta um mecanismo para forçar a privacidade e a utilidade enquanto a posição de um usuário é rastreada. O sistema é modelado através de um sistema de transição e, em seguida, também utiliza o *software* SynthSMV para realizar a síntese do ofuscador. Através do resultado obtido no SynthSMV é desenvolvida uma aplicação em tempo real para realizar a ofuscação da posição de um robô móvel.

Nos trabalhos sobre ofuscação em SEDs (Wu et al., 2018; Góes et al., 2018) é considerado que um estado secreto do sistema nunca é reportado pelo ofuscador. Entretanto, em alguns casos é necessário que o estado secreto seja reportado para preservar todo o comportamento do sistema, como será apresentado no exemplo 1.

Exemplo *1.* Considere o caso de uma ferrovia em que um ou mais trens se deslocam sobre o mesmo conjunto de trilhos, como ilustrado na figura 1. Através da utilização de sensores, algumas informações úteis sobre a geolocalização dos trens são enviadas para usuários externos através de uma rede de comunicação. Desta forma, é possível estimar a localização dos trens, o que permite traçar rotas mais eficientes, evitando congestionamentos. Porém, se um usuário mal-intencionado conhecer a localização exata de cada trem, o mesmo poderá causar danos graves ao sistema, como danos físicos ao sistema ou a pessoas que estejam no local. Assim, pontos de interesse como interceções e pontes precisam de um cuidado especial para serem reportados aos usuários externos, modeladas como estados secretos.

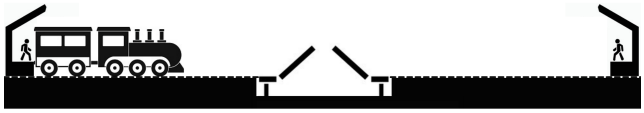

Figura 1. Linha de trem simplificada.

O exemplo 1 ilustra uma situação em que é necessário para o ofuscador reportar a passagem pela ponte, modelada como estado secreto, para preservar todo o comportamento original, uma vez que o trem só consegue transitar entre às duas estações através da ponte. Portanto, é necessário desenvolver um ofuscador que consiga reportar os estados secretos do sistema e, em simultâneo, garantir a privacidade do mesmo. Neste trabalho será apresentado uma nova definição de ofuscação para sistemas a eventos discretos cuja principal diferença é permitir que, em determinadas situações, os estados secretos do sistema sejam reportados para os usuários externos. Além de proteger o anonimato do ofuscador, uma vez que o comportamento do sistema original é preservado, esta abordagem amplia os sistemas que podem ser ofuscados, pois é mais permissiva no tratamento dos estados secretos quando comparada a abordagens anteriores como apresentadas em (Góes et al., 2018) e (Wu et al., 2018).

Diferentemente de outras abordagens de segurança computacional como a criptografia, a abordagem de ofuscação proposta neste trabalho trata a segurança da informação transmitida aos usuários externos ao nível de modelo. Dessa forma, um usuário mal-intencionado não conseguirá

perceber a presença de uma interface de segurança como seria o caso da informação criptografada, uma vez que a segurança da informação gerada no modelo ofuscado já foi realizada antes do envio da informação através de meios de comunicação.

O presente trabalho é organizado da seguinte forma. Na Seção 2 são apresentados os conhecimentos preliminares necessários para a compreensão deste tralhado. A Seção 3 expõe a formulação do problema da síntese do ofuscador. Na Seção 4 apresentamos um algoritmo para a síntese do sistema aumentado e na Seção 5 os procedimentos para sintetizar o ofuscador, proposto neste trabalho, que garante um nível variável de privacidade e utilidade. Por fim, a conclusão do trabalho é apresentada na Seção 6.

2. PRELIMINARES

Uma forma comum de modelar o comportamento de sistemas discretos é através da utilização de sistemas de transição. Um sistema de transição é composto por estados, os quais podem ser ou não rotulados. Cada estado representa uma situação específica do sistema, e a evolução ocorre por transições que ligam um estado a outro. Um sistema de transição rotulado é definido como $TS = (S, Act, \rightarrow, I, AP, L)$, em que S é o conjunto de estados, Act é o conjunto de eventos, $\rightarrow \subseteq S \times Act \times S$ é uma relação de transição, $I \subseteq S$ é o conjunto dos estados iniciais, AP é o conjunto de proposições atômicas, $L : S \rightarrow 2^{AP}$ é a função de rotulação (Baier and Katoen, 2008).

No caso de um sistema de transição rotulado, o conjunto de eventos, Act, é utilizado para nomear as transições, assim a evolução do sistema pode ser vista através de uma linguagem. Por outro lado, as proposições atômicas presentes no conjunto AP são utilizadas para definir qual situação específica do sistema cada estado representa. Portanto, a função de rotulação $L(s)$ atribui a cada estado $s \in S$ um conjunto de proposições atômicas pertencente ao conjunto 2^{AP}. Considere a fórmula lógica proposicional Φ, então $s \models \Phi \iff L(s) \models \Phi$, onde $\models$ é a relação de satisfação. Para estudar o comportamento concorrente entre dois ou mais sistemas de transição utilizaremos o mecanismo *handshaking* (Baier and Katoen, 2008). Considere os sistemas de transição $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$, em que $i = 1, 2$. Com isso, é possível definir a composição síncrona $TS_1 \| TS_2$ da seguinte forma:
$TS_1 \| TS_2 := (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$ em que $L((s_1, s_2)) = L_1(s_1) \cup L_2(s_2)$ e a relação de transição $\rightarrow$ é definida como: (1) Se $\alpha \in Act_1 \cap Act_2$, $(s_1 \xrightarrow{\alpha}_1 s_1') \wedge (s_2 \xrightarrow{\alpha}_2 s_2')$ então $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2')$; (2) Se $\alpha \notin Act_1 \cap Act_2$, $(s_1 \xrightarrow{\alpha}_1 s_1')$ (resp. $(s_2 \xrightarrow{\alpha}_2 s_2')$) então $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2)$ (resp. $(s_1, s_2) \xrightarrow{\alpha} (s_1, s_2')$). Portanto, no mecanismo de *handshaking* a execução de um evento comum a ambos os processos, $\alpha \in Act_1 \cap Act_2$, somente poderá ser executada pelos dois processos simultaneamente. Por outro lado, um evento particular de um processo, $\alpha \in (Act_1 \setminus Act_2) \cup (Act_2 \setminus Act_1)$, pode ser executado assim que ativo de forma independente em cada processo.

Uma forma de especificar as propriedades temporais de um sistema de transição é a partir de *Computation Tree Logic* (CTL) (Baier and Katoen, 2008), um formalismo lógico temporal baseado em ramificações. As expressões em CTL podem possuir diversos operadores temporais, porém para este trabalho somente os operadores temporais AG e EF serão considerados. Dito isso, dado a fórmula em CTL $AG(\phi)$ significa que todos os estados alcançáveis do sistema devem satisfazer a fórmula ϕ. Por outro lado, a fórmula $EF(\phi)$ especifica que o sistema deve conseguir alcançar um estado em que ϕ seja satisfeito. Por fim, é válido ressaltar que $AG(\phi_1) \wedge AG(\phi_2) \iff AG(\phi_1 \wedge \phi_2)$.

Especificações em CTL no formato $AG(EF(p))$, em que p não contem nenhum outro operador temporal, expressam especificações de controlabilidade e de não bloqueio (Ehlers et al., 2017) e podem ser utilizadas para a síntese de controle supervisório. Uma extensão desse tipo considera o caso de múltiplos conjuntos de estados sempre devem ser alcançáveis. Para isso, são utilizadas as especificações chamadas de *combined invariance and reachability* (CIR) (Van Hulst et al., 2017), que possuem o seguinte formato:

$$AG\left(\bigwedge_{i \in I} p_i \wedge \bigwedge_{j \in J} EF(p_j)\right) \tag{1}$$

Qualquer especificação que possa ser escrita no formato CIR pode ser utilizada para calcular um controlador maximamente permissivo. Uma forma eficiente de se calcular esse controlador pode ser encontrado em (Rawlings et al., 2018).

3. FORMULAÇÃO DO PROBLEMA

O sentido da palavra ofuscar remete a confundir alguém ou obscurecer o sentido de algo. Essa é justamente a ideia por trás do conceito de ofuscação que tratamos neste trabalho, uma vez que tentamos confundir os observadores de um dado sistema para que eles não consigam saber quando o sistema encontra-se em um estado dito secreto. Como ilustrado na figura 2, para realizar a ofuscação consideramos que todo evento, α, gerado pelo sistema é enviado ao ofuscador que reporta ao ambiente uma palavra, α_o, para manter os segredos do sistema protegido dos agentes que observam a evolução do sistema externamente. Diferentemente da abordagem anteriores de ofuscação, consideraremos neste trabalho que, em determinadas situações, os estados secretos do sistema poderão ser reportados pelo ofuscador. Dessa forma, sistemas que possuem estados secretos no seu caminho principal e necessariamente precisariam ser reportados para preservar a dinâmica do sistema original poderão ser ofuscados. Como pode ser observado no exemplo 1, o trem necessariamente precisa passar sobre a ponte para transitar entre às duas estações, pois é o único caminho disponível. Considerando a ponte como o estado secreto do sistema, seria impossível criar, a partir das abordagens anteriores, um ofuscador que garante a utilidade da informação, uma vez que o ofuscador não conseguiria reportar a passagem pelo estado secreto. Portanto, a nova definição de ofuscação proposta neste trabalho amplia os sistemas que podem ser ofuscados, além de dificultar para os agentes mal-intencionados detectar a presença do ofuscador a partir de observações incoerentes. Com isso, as seguintes hipóteses são feitas sobre a estrutura do ofuscador:

- O evento gerado pelo sistema é imediatamente enviado para o ofuscador,

- O intruso possui uma cópia do modelo do sistema,
- O intruso somente consegue observar os eventos reportados pelo ofuscador.

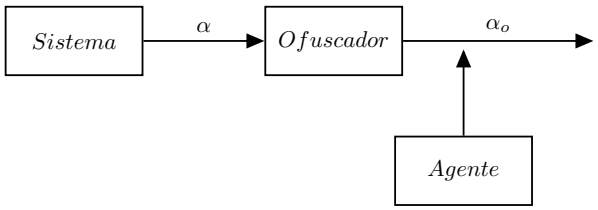

Figura 2. Estrutura do ofuscador.

Inspirado no problema de controle supervisório para sistemas de transição (Rawlings et al., 2018), nós iremos definir o problema da síntese do ofuscador como se segue.

Definição *1.* **Problema de Síntese do Ofuscador:** Dado um sistema de transição rotulado $TS = (S, Act, \rightarrow, I, AP, L)$, um conjunto de estados secretos $S_s \subseteq S$ e uma especificação em CTL θ, então o ofuscador TS_{ofusc} deve satisfazer os seguintes critérios:

- maximiza o conjunto de estados que satisfazem a especificação, θ,
- minimiza o conjunto de transições desabilitadas pelo ofuscador.

4. SÍNTESE DO SISTEMA AUMENTADO

Para realizar a síntese do ofuscador é necessário ser possível diferenciar a ocorrência de um evento na planta de sua observação por um agente externo. Para isso, considere os conjuntos Act_r e Act_ω como cópias de Act porém com os eventos renomeados com os subscritos r e ω respectivamente. Assim, podemos definir as operações de renomeação dos eventos, R_r e R_ω, da seguinte forma.

A operação de renomeação R_r é definida como $R_r : Act^* \rightarrow Act_r^*$, em que: $R_r(\varepsilon) = \varepsilon$, $R_r(\alpha) = \alpha_r$ e $R_r(s\alpha) = (s\alpha)_r = R_r(s)R_r(\alpha) = R_r(s)\alpha_r$, em que $\alpha \in Act$ e $s \in Act^*$. Já a operação inversa de renomeação $R_r^{-1} : Act_r^* \rightarrow Act^*$ é definida como $R_r^{-1}(\varepsilon) = \varepsilon$, $R_r^{-1}(\alpha_r) = \alpha$ e $R_r^{-1}((s\alpha)_r) = R_r^{-1}(s_r\alpha_r) = R_r^{-1}(s_r)R_r^{-1}(\alpha_r) = R_r^{-1}(s_r)\alpha$. Analogamente definimos a operação de renomeação $R_\omega : Act^* \rightarrow Act_\omega^*$ porém substituindo o subscrito r por ω.

Uma forma de representar a concorrência entre as execuções dos eventos na planta e as possíveis observações realizadas pelo agente externo é através da criação de um sistema aumentado descrito no Algoritmo 1. No primeiro passo do Algoritmo 1, é criada uma cópia do conjunto de estados S chamada de S_p. Já no passo 2, é criado o conjunto Act_r a partir da operação R_r que contém todos os eventos de Act renomeados com o subscrito r. Analogamente, o passo 3 cria o conjunto $Act_{r\omega}$ a partir da operação R_ω que contém todos os eventos de Act_r renomeados com o subscrito $r\omega$. Assim, os eventos contidos em $Act_r \cup Act_{r\omega}$ representam as observações dos eventos em Act por algum agente externo. Desta forma, é possível definir o conjunto dos eventos da visão pública do sistema, Act_p, no passo 4. No passo 5 definimos a visão pública do sistema, TS_p. Para tanto, é necessário definir a sua relação de transição, $\rightarrow_p \subseteq S_p \times Act_p \times S_p$, em que substitui-se as transições $\rightarrow$ rotuladas por $\alpha \in Act$, para todos os estados $s_p \in S_p$, por duas novas transições rotuladas $\alpha_r \in Act_p$ e $\alpha_{r\omega} \in Act_p$. Após computar a visão pública do sistema, precisamos definir o comportamento concorrente entre o sistema original e a visão pública. Para isso, no passo 6 é definido o sistema de transição TS_1, que representa se é a vez do sistema original executar um evento, $\neg o$, ou da visão pública gerar uma observação, o. Desta forma, como $I_1 = \{\neg o\}$, foi definido que o sistema original tem o direito de executar o primeiro evento no sistema aumentado. O sistema original somente poderá executar um novo evento caso o evento ω ou um evento com o subscrito $r\omega$ for executado. Assim, toda vez que o sistema executa um evento, a visão pública do sistema tem a possibilidade de gerar uma ou mais observações imediatamente. No passo 7 é definido o sistema de transição TS_2, que registra se quem gerou o último evento foi o sistema, $\neg p$, ou a visão pública, p. Por fim, no passo 8 o sistema aumentado, TS_{aug}, é formado pelo *handshaking* entre os sistemas TS, TS_p, TS_1 e TS_2 e representa todas as possíveis execuções de eventos pelo sistema original e as possíveis observações realizadas pelos agentes externos através da visão pública.

Algoritmo 1 Procedimento para criar o Sistema Aumentado

Entrada: $TS = (S, Act, \rightarrow, I, AP, L)$
Saída: TS_{aug}

1: $S_p \leftarrow S$
2: $Act_r = R_r(Act)$
3: $Act_{r\omega} = R_\omega(Act_r)$
4: $Act_p \leftarrow Act_r \cup Act_{r\omega}$
5: Defina $TS_p = (S_p, Act_p, \rightarrow_p, I, AP, L)$ em que:
   - $(s_{1_p} \xrightarrow{\alpha_r}_p s_{2_p}) \leftarrow (s_1 \xrightarrow{\alpha} s_2)$, em que $s_{1_p}, s_{2_p} \in S_p$, $\alpha_r = R_r(\alpha)$, $s_1, s_2 \in S$ e $\alpha \in Act$
   - $(s_{1_p} \xrightarrow{\alpha_{r\omega}}_p s_{2_p}) \leftarrow (s_1 \xrightarrow{\alpha} s_2)$, em que $s_{1_p}, s_{2_p} \in S_p$, $\alpha_{r\omega} = R_\omega(R_r(\alpha))$, $s_1, s_2 \in S$ e $\alpha \in Act$
6: Defina $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ em que:
   - $S_1 = \{0, 1\}$
   - $Act_1 = Act \cup Act_r \cup Act_{r\omega} \cup \{\omega\}$
   - $0 \xrightarrow{\alpha}_1 1$ se $\alpha \in Act$
   - $1 \xrightarrow{\alpha}_1 1$ se $\alpha \in Act_r$
   - $1 \xrightarrow{\alpha}_1 0$ se $\alpha \in \{Act_{r\omega} \cup \{\omega\}\}$
   - $AP_1 = \{o, \neg o\}$
   - $L_1(0) = \{\neg o\}$
   - $L_1(1) = \{o\}$
   - $I_1 = \{0\}$
7: Defina $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ em que:
   - $S_2 = \{0, 1\}$
   - $Act_2 = Act \cup Act_r \cup Act_{r\omega} \cup \{\omega\}$
   - $1 \xrightarrow{\alpha}_2 1$ se $\alpha \in \{Act_r \cup Act_{r\omega} \cup \{\omega\}\}$
   - $1 \xrightarrow{\alpha}_2 0$ se $\alpha \in Act$
   - $0 \xrightarrow{\alpha}_2 0$ se $\alpha \in Act$
   - $0 \xrightarrow{\alpha}_2 1$ se $\alpha \in \{Act_r \cup Act_{r\omega} \cup \{\omega\}\}$
   - $AP_2 = \{p, \neg p\}$
   - $L_2(0) = \{\neg p\}$
   - $L_2(1) = \{p\}$
   - $I_2 = \{1\}$
8: $TS_{aug} = TS \| TS_p \| TS_1 \| TS_2$

Exemplo *2.* Para ilustrar a criação do sistema aumentado, considere o sistema de transição apresentado na figura 3 que modela o comportamento de um trem que viaja da estação 1 para a estação 3. Portanto, $TS^2 = (S^2, Act^2, \rightarrow^2, I^2, AP^2, L^2)$ em que $S^2 = \{1, 2, 3\}$, $Act^2 = \{a, b, c\}$, $I^2 = \{1\}$, $AP^2 = \emptyset$, $L^2(s) = \emptyset$, $\forall s^2 \in S^2$, e $\rightarrow^2$ é definido como ilustrado na figura 3.

Para a síntese do ofuscador, precisamos primeiro obter o sistema aumentado seguindo o Algoritmo 1. No primeiro passo do algoritmo criamos o conjunto S_p^2 a partir de uma cópia de S^2, ou seja, $S_p^2 = \{1, 2, 3\}$. Em seguida criamos Act_r^2 através da renomeação do conjunto Act^2 pela função R_r, assim obtemos $Act_r^2 = \{a_r, b_r, c_r\}$. Da mesma forma, criamos o conjunto $Act_{r\omega}^2$ através da renomeação do conjunto Act_r^2 pela função R_ω, assim $Act_{r\omega}^2 = \{a_{r\omega}, b_{r\omega}, c_{r\omega}\}$. Com isso, no passo 4 definimos o conjunto $Act_p^2 = \{a_r, a_{r\omega}, b_r, b_{r\omega}, c_r, c_{r\omega}\}$. No passo 5, definimos a visão pública do sistema dada por $TS_p^2 = (S_p^2, Act_p^2, \rightarrow_p^2, I^2, AP^2, L^2)$, com isso substituímos as transições $\rightarrow^2$ rotuladas por $\sigma \in Act^2$, para todos os estados $s_p^2 \in S_p^2$, por duas novas transições rotuladas $\sigma_r \in Act_p^2$ e $\sigma_{r\omega} \in Act_p^2$. Por exemplo, a transição do estado 1 para o estado 2, $(1 \xrightarrow{a} 2)$, é substituída pelas transições $(1 \xrightarrow{a_r}_p^2 2)$ e $(1 \xrightarrow{a_{r\omega}}_p^2 2)$. Em TS_p^2, a substituição das demais transições é realizado de forma análoga como pode ser visto na figura 4.

No passo 6 devemos montar o sistema de transição $TS_1^2 = (S_1^2, Act_1^2, \rightarrow_1^2, I_1^2, AP_1^2, L_1^2)$, em que $Act_1^2 = \{a, a_r, a_{r\omega}, b, b_r, b_{r\omega}, c, c_r, c_{r\omega}, \omega\}$, representado na figura 5.

O sistema de transição TS_1^2 representa a alternância entre o sistema original e a visão pública, assim o estado 1, com rótulo $\neg o$, caracteriza a vez do sistema original gerar um evento e o estado 2, com rótulo o, caracteriza a vez da visão pública reportar um evento. Observe que após o sistema original gerar um evento, ou seja, a ocorrência de algum evento do conjunto $Act^2 = \{a, b, c\}$, a visão pública sempre tem a possibilidade de reportar um evento contido no conjunto $Act_r^2 \cup Act_{r\omega}^2 \cup \{\omega\}$. Por fim, é válido salientar que o estado 1 é o inicial, pois possui o rótulo $\neg o$, o que garante que o sistema original é o primeiro a movimentar.

No passo 7 é definido o sistema de transição $TS_2^2 = (S_2^2, Act_2^2, \rightarrow_2^2, I_2^2, AP_2^2, L_2^2)$, representado na figura 6.

Por sua vez, o sistema de transição TS_2^2 representa qual dos agentes envolvidos na ofuscação gerou o último evento, o sistema original ou a visão pública. Desta forma, as transições que levam para o estado 1, p, são compostas apenas por eventos contidos no conjunto $Act_r^2 \cup Act_{r\omega}^2 \cup \omega$, portanto caracterizam que no estado 1 a visão pública gerou o último evento. Analogamente, as transições que levam para o estado 2, $\neg p$, são constituídas por eventos do conjunto Act^2 indicando, assim, que no estado 2 o sistema original gerou o último evento. Observe que neste caso o estado inicial é indiferente, pois no estado inicial nenhum dos agentes gerou o último evento, assim foi escolhido o estado 1 como inicial de forma arbitrária.

Por fim, para sintetizar o sistema aumentado utilizaremos a operação de *handshaking*, assim $TS_{aug}^2 = TS^2 \| TS_p^2 \| TS_1^2 \| TS_2^2$. Parte do sistema aumentado obtido é retratado na figura 7.

Observe que o conjunto de estados S_{aug}^2 é formado pela composição $S^2 \times S_p^2 \times S_1^2 \times S_2^2$, porém, para simplificação da figura 7, optou-se pela nomeação dos estados na forma $S^2 \times S_p^2$. Além disso, considera-se que a qualidade de informação gerada pelo ofuscador, ou a sua utilidade, diminui a medida que o número de transições que liga o estado real do sistema ao estado reportado pelo ofuscador aumenta. Por consequência, buscamos construir um ofuscador que, além de garantir a privacidade, mantenha a utilidade da informação gerada. Para isso, definimos a **função de utilidade** $D : S \times S \rightarrow N$ que modela o número de transições que ligam dois estados do sistema. Por exemplo, para o estado $(2, 3)$, ou seja, o sistema original encontra-se no estado 2 e a visão pública no estado 3, o valor da função de utilidade é dado por $D(2, 3) = 1$. Analogamente para o estado $(3, 2)$ temos que a função de utilidade é $D(3, 2) = 1$.

A partir do resultado obtido nota-se que o sistema aumentado modela quais eventos a visão pública pode reportar em resposta a qual evento o sistema original gerou. Por exemplo, a ocorrência do evento a no sistema original acarreta possíveis respostas pela visão pública: $a_r, a_{r\omega}, \omega$. O evento a_r significa que a visão pública reportou a ocorrência do evento a para os observadores externos e mantém a vez de movimentar com a visão pública. Já o evento $a_{r\omega}$ também reporta a ocorrência do evento a, mas, por conter o subscrito ω, alterna a vez e espera o sistema original gerar um novo evento. Por fim, o evento ω não reporta a ocorrência de um evento naquele intervalo e retorna a vez para o sistema original movimentar. Portanto, cada evento gerado pelo sistema original acarreta três possíveis respostas pela visão pública para os observadores externos.

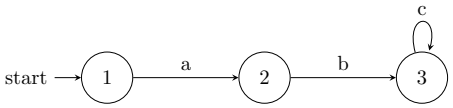

Figura 3. Sistema de transição TS^2 para o comportamento do trem do exemplo 2

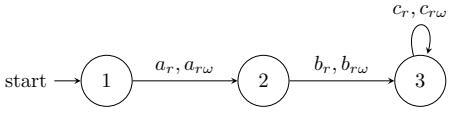

Figura 4. Visão pública do sistema de transição do exemplo 2, TS_p^2

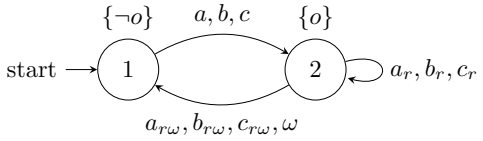

Figura 5. Sistema de transição que modela a alternância entre sistema e visão pública do exemplo 2, TS_1^2

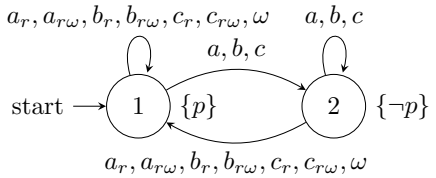

Figura 6. Sistema de transição que indica qual agente gerou o último evento, TS_2^2

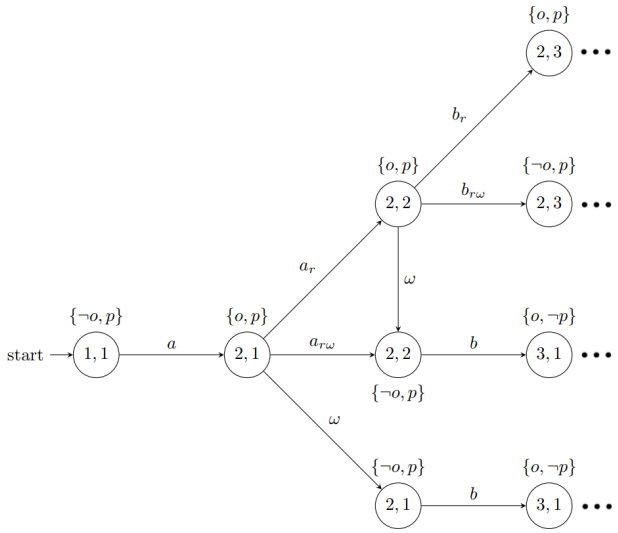

Figura 7. Parte do sistema aumentado TS_{aug}^2

5. SÍNTESE DO NOVO OFUSCADOR COM GARANTIAS DE DIFERENTES NÍVEIS DE PRIVACIDADE E UTILIDADE

A síntese do ofuscador empregada neste trabalho utiliza da teoria de controle supervisório para limitar o comportamento do sistema aumentado e, com isso, eliminar as observações indesejadas. Para resolver o problema de controle supervisório utilizamos o programa SynthSMV que implementa o Algoritmo 2 apresentado em Rawlings et al. (2018). O SynthSMV utiliza como entradas um sistema de transição rotulado e uma especificação em CTL do tipo CIR e calcula o supervisor como um único *Binary Decision Diagram* (BDD) que registra quais eventos são desabilitados em cada estado do sistema. Portanto, como já definimos a criação do sistema aumentado no algoritmo 1, precisamos definir a especificação CTL que irá limitar o comportamento de maneira adequada.

Inicialmente precisamos definir o novo critério de privacidade. Desejamos que somente seja possível para o ofuscador reportar um estado secreto caso o sistema encontre-se em uma posição considerada segura e, analogamente, caso o sistema encontre-se em um estado secreto, o ofuscador somente poderá reportar estados considerados seguros. Desta forma, em CTL podemos especificar esta restrição como $AG(p \wedge ((s \vee s_p) \in S_s) \rightarrow D(s, s_p) \geq c_1)$, ou seja, se quem gerou o último evento foi o ofuscador, p, e o sistema ou o ofuscador encontra-se em um estado secreto, $(s \vee s_p) \in S_s$, então a distância entre o estado real do sistema e o estado reportado deve ser maior ou igual a c_1, $D(s, s_p) \geq c_1$.

Por outro lado, a informação gerada pelo ofuscador deve ser útil, o que pode ser escrito em CTL como uma especificação do tipo $AG(p \rightarrow D(s, s_p) \leq c_2)$, em que um evento gerado pelo ofuscador, p, não poderá levar o sistema a um estado que a distância entre o estado real e o estado reportado do sistema seja maior ou igual a c_2, $D(s, s_p) \leq c_2$. Por fim, é preciso que o sistema aumentado sob controle do supervisor não entre em estados que impeçam o sistema original de evoluir, o que é representado em CTL como $AG(EF(\neg o))$, assim garantimos que existe pelo menos um caminho que leva a um estado em que é a vez da planta movimentar, $\neg o$. Portanto, para obter o comportamento desejado pelo ofuscador devemos combinar estas três especificações da seguinte forma:

$$AG(((p \wedge (s \vee s_p) \in S_s) \rightarrow (D(s, s_p) \geq c_1)) \wedge (p \rightarrow (D(s, s_p) \leq c_2)) \wedge EF(\neg o)) \quad (2)$$

Assim, o programa SynthSMV recebe como entrada o sistema aumentado, TS_{aug} (Algoritmo 1), e a especificação em CTL da equação 2. A saída do programa é um supervisor que possui a função ofuscador, denominado TS_{ofusc}, que garante a utilidade e um nível de privacidade variável. A seguir é apresentado parte do código utilizado no SynthSMV.

```
MODULE main
VAR state : # declarar os estados;
IVAR event : # declarar os eventos;
INIT state = # declarar os estados iniciais;
ASSIGN next(state) :=
  # declarar as transicoes
  case
    state = 0:
      case
        event = a : 1;
        TRUE : 0;
      esac;
  .
  .
  .
DEFINE
    #definir as propriedades de cada
      estado
SYNTH AG(((p & (s | s_p) ∈ s_s)->(D(s,s_p)>=
c_1)) & (p->(D(s,s_p) <= c_2)) & EF(¬o))
```

Como o SynthSMV foi elaborado como uma extensão do verificador de modelos simbólicos NuSMV, ambos utilizam a mesma linguagem de programação. Como apresentado no código acima, para declarar um sistema de transição no SynthSMV é necessário declarar o conjunto de estados, o conjunto de eventos, o conjunto de estados iniciais e a função de transição. Em seguida, através da função de rotulação, deve-se declarar as proposições atômicas de cada estado. Após declarar o sistema de transição é realizada a síntese do supervisor através de dada especificação CTL do tipo CIR.

Exemplo *3.* Analogamente ao exemplo 1, consideramos o caso de uma ferrovia simplificada. Considere um trem

deslocando-se da estação 1 para 6 como apresentado na figura 8. Desta forma, para este caso temos que $TS^3 = (S^3, Act^3, \rightarrow^3, I^3, AP^3, L^3)$ em que $S^3 = \{1,2,3,4,5,6\}$, $Act^3 = \{a,b,c,d,e,f\}$, $I^3 = \{1\}$, $AP^3 = \emptyset$, $L^3(s) = \emptyset$, $\forall s^3 \in S^3$, e $\rightarrow^3$ é definido como apresentado na figura 8. O estado 3, em vermelho, modela uma ponte, por isso é considerado secreto, portanto $S^3_s = \{3\}$. Deseja-se ofuscar este sistema de modo a manter o nível de utilidade e privacidade proposto.

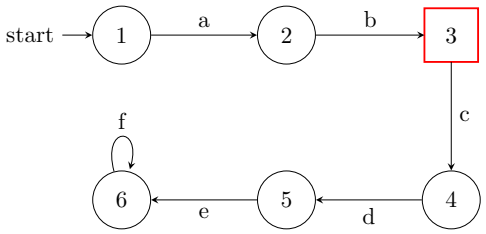

Figura 8. Exemplo ferrovia com ponte.

O primeiro passo na síntese do ofuscador é a criação do sistema aumentado a partir do algoritmo 1 o qual retorna o sistema de transição TS^3_{aug}. Em seguida devemos estabelecer os parâmetros para que a especificação definida na equação 2 satisfaça os níveis de privacidade e utilidade desejados. Para isso, definimos as constantes de privacidade $c_1 = 1$ e de utilidade $c_2 = 2$. Assim, a partir dos parâmetros propostos, obtemos a seguinte especificação em CTL:

$$AG(((p \wedge (s^3 \vee s^3_p) \in S^3_s) \rightarrow (D(s^3, s^3_p) \geq 1)) \wedge (p \rightarrow (D(s^3, s^3_p) \leq 2)) \wedge EF(\neg o)) \quad (3)$$

em que $AG((p \wedge (s^3 \vee s^3_p) \in S^3_s) \rightarrow (D(s^3, s^3_p) \geq 1))$ representa o critério de privacidade e significa que sempre que for a vez do ofuscador movimentar e o sistema ou sua visão pública estiverem em um estado secreto, então a distância entre o estado real do sistema e o estado da visão pública reportado devem estar a uma distância maior ou igual a 1. Por sua vez, o critério de utilidade é dado por $AG(p \rightarrow (D(s^3, s^3_p) \leq 2))$ e define que o ofuscador só poderá movimentar para estados em que a distância entre o estado real do sistema e o estado da visão pública reportado seja menor ou igual a 2. Já a especificação $AG(EF(\neg o))$ garante que o sistema original nunca será impedido de movimentar pelo ofuscador.

A partir disso, o sistema aumentado, TS^3_{aug}, junto a especificação proposta acima, foram utilizadas como entradas para o SynthSMV para obter o ofuscador. A solução encontrada no programa é ilustrada na forma de um sistema de transição na figura 9.

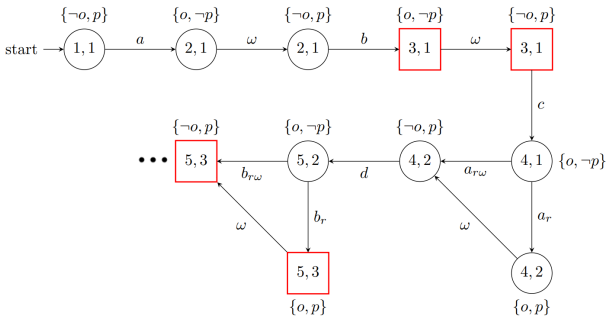

Figura 9. Ofuscador sintetizado no SynthSMV para o exemplo 3.

A partir do resultado, fica claro que, para este conjunto de restrições, caso o sistema original evolua para um estado secreto, o ofuscador irá restringir o comportamento da visão pública para que ela somente consiga evoluir para estados que satisfazem o critério de privacidade. Esta situação é observada nos estados com nome (3,1), em que a visão pública é obrigada a permanecer no estado 1 para que o sistema original possa evoluir pelo estado secreto, pois $D(3,1) = 2$ satisfaz o critério de privacidade. Da mesma forma, o ofuscador somente irá permitir que a visão pública evolua para um estado secreto caso o sistema original encontre-se em um estado que satisfaça o critério de privacidade. Assim, nos estados com nome (5,3) o sistema original deve evoluir para uma posição segura para que a visão pública possa evoluir para o estado secreto.

Também é possível observar situações como a do estado $(4,1)$, cujo rótulo é $\{o, \neg p\}$, em que a distância entre o estado real e o estado reportado é maior que $c_2 = 2$, ou seja, $(D(s^3, s^3_p) \leq 2)$ é inválido. Desta forma, neste estado o ofuscador desabilitará a ocorrência do evento ω e somente permitirá que o sistema transite para estados em que $p \rightarrow (D(s^3, s^3_p) \leq 2)$ seja verdadeiro.

Portanto, este é um exemplo em que as abordagens anteriores, como as utilizadas por Wu et al. (2018) e Góes et al. (2018), não permitiriam a ofuscação, preservando a utilidade da informação gerada, pois consideram que o ofuscador não pode permitir que a visão pública transite para estados secretos, ou seja $AG(s_p \notin S_S)$, e faz com que a evolução da visão pública fique restrita aos estados 1 e 2. Além disso, pelo fato de restringir o comportamento da visão pública para não evoluir por estados secretos, as abordagens anteriores permitem que, a longo prazo, os observadores suspeitem da existência do ofuscador, pois o comportamento não é totalmente preservado. Ambos os problemas são solucionados neste trabalho ao permitir que a visão pública transite pelos estados secretos, mas mantendo a privacidade do sistema na totalidade.

6. CONCLUSÃO

Neste artigo, apresentamos uma nova abordagem de ofuscação que garante diferentes níveis de privacidade e utilidade. Pelo que sabemos, este é o primeiro trabalho a tratar de casos de ofuscação em que é possível reportar aos usuários externos que o sistema encontra-se em um estado

considerado secreto. Para solucionar o problema da síntese do ofuscador foi utilizado o *software* SynthSMV para computar o supervisor ótimo que, por sua vez, tem o papel de restringir o comportamento do sistema aumentado a partir de uma especificação do tipo CIR. A mudança na especificação sugerida na equação 2, quando comparada com a proposta por Wu et al. (2018), faz com que, em determinadas situações, seja possível para o ofuscador reportar a evolução do sistema por um estado secreto. Essa diferença faz com que seja possível ofuscar uma variedade maior de sistemas, uma vez que problemas específicos podem exigir que um estado secreto seja reportado. Além disso, permitir que o ofuscador reporte os estados secretos dificultam para os usuários mal-intencionados a identificação da presença do ofuscador, uma vez que toda a dinâmica do sistema será preservada.

Como trabalho futuro, pretende-se buscar outros exemplos em que seja possível implementar esta nova abordagem que garante diferentes níveis de privacidade e utilidade. Além disso, pretendemos de investigar novos critérios de utilidade diferente da distância física ou número de transições entre estados.

AGRADECIMENTOS

REFERÊNCIAS

Alguliyev, R., Imamverdiyev, Y., and Sukhostat, L. (2018). Cyber-physical systems and their security issues. *Computers in Industry*, 100, 212–223.

Alves, M.R., Pena, P.N., and Rudie, K. (2022). Discrete-event systems subject to unknown sensor attacks. *Discrete Event Dynamic Systems*, 32(1), 143–158.

Baier, C. and Katoen, J.P. (2008). *Principles of model checking*. MIT press.

Carvalho, L.K., Wu, Y.C., Kwong, R., and Lafortune, S. (2018). Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, 97, 121–133.

Duckham, M. and Kulik, L. (2005). A formal model of obfuscation and negotiation for location privacy. In *International conference on pervasive computing*, 152–170. Springer.

Ehlers, R., Lafortune, S., Tripakis, S., and Vardi, M.Y. (2017). Supervisory control and reactive synthesis: a comparative introduction. *Discrete Event Dynamic Systems*, 27(2), 209–260.

Gilchrist, A. (2016). *Industry 4.0: the industrial internet of things*. Springer, Apress Berkeley, CA.

Góes, R.M., Rawlings, B.C., Recker, N., Willett, G., and Lafortune, S. (2018). Demonstration of indoor location privacy enforcement using obfuscation. *IFAC-PapersOnLine*, 51(7), 145–151.

Lima, P.M., Alves, M.V.S., Carvalho, L.K., and Moreira, M.V. (2019). Security against communication network attacks of cyber-physical systems. *Journal of Control, Automation and Electrical Systems*, 30(1), 125–135.

Rawlings, B.C., Lafortune, S., and Ydstie, B.E. (2018). Supervisory control of labeled transition systems subject to multiple reachability requirements via symbolic model checking. *IEEE Transactions on Control Systems Technology*, 28(2), 644–652.

Van Hulst, A., Reniers, M.A., and Fokkink, W.J. (2017). Maximally permissive controlled system synthesis for non-determinism and modal logic. *Discrete Event Dynamic Systems*, 27(1), 109–142.

Wu, Y.C., Raman, V., Rawlings, B.C., Lafortune, S., and Seshia, S.A. (2018). Synthesis of obfuscation policies to ensure privacy and utility. *Journal of Automated Reasoning*, 60(1), 107–131.