Agregação de Dados na Nuvem com Garantias de Segurança e Privacidade

Leandro Silva[1], **Rodolfo Silva**[1], **Andrey Brito**[1], **Pedro Barbosa**[1]

[1]Departamento de Sistemas e Computação – Universidade Federal de Campina Grande
58.429-140 – Campina Grande – PB – Brasil

leandro@ufcg.edu.br, rodolfomarinho@copin.ufcg.edu.br

andrey@computacao.ufcg.edu.br, pedroyossis@copin.ufcg.edu.br

Abstract. *The use of cloud computing has become common due to advantages such as low cost and sizing of computing resources according to demand. However, there are concerns about security and privacy, because critical data – especially in IoT applications – are stored and processed in the cloud. This paper proposes a software architecture that supports multiple approaches to secure data aggregation. The use of this architecture proved to be viable from experiments with the use of homomorphic encryption techniques and security extensions in hardware (Intel SGX), which, to our best knowledge, had never been used in a cloud environment.*

Resumo. *O uso da computação na nuvem tem se tornado comum por vantagens como baixo custo e contratação de recursos de acordo com a demanda. Todavia, surgem preocupações com segurança e privacidade, pois dados críticos – especialmente em aplicações de IoT – são armazenados e processados na nuvem. Este artigo propõe uma arquitetura de software com suporte a múltiplas abordagens para a agregação segura de dados. O uso dessa arquitetura se mostrou viável em experimentos realizados com a utilização de técnicas de criptografia homomórfica e de extensões de segurança em hardware (Intel SGX), que, segundo nossas pesquisas, ainda não havia sido aplicado em um ambiente de nuvem.*

1. Introdução

Computação na nuvem é um termo que representa serviços hospedados *on-line*. Tais serviços são acessíveis pela Internet, metaforicamente chamada de "nuvem" [Markovic et al. 2013]. Do ponto de vista empresarial, esse modelo de computação é muito atraente porque despesas com software, hardware e infraestrutura física são reduzidos drasticamente, já que a contratação de recursos computacionais é feita em grãos menores e a expansão ou redução destes recursos pode ser feita de forma automatizada de acordo com a demanda. Ainda, a operação da infraestrutura é delegada a um provedor que, por sua escala, tende a ser muito mais eficiente.

Apesar de todas as vantagens citadas, a computação na nuvem traz consigo preocupações com segurança e privacidade. Em uma pesquisa da *Cloud Industry Forum* [CIF 2015] para o Reino Unido, os dois principais inibidores para a adoção de computação na nuvem são as preocupações com segurança dos dados e privacidade, mencionados por 70% e 61% dos consultados, respectivamente. Uma vez que, na maioria dos

casos, os serviços na nuvem são fornecidos de forma compartilhada, ataques adicionais, tanto externos quanto internos podem ocorrer [Pasupuleti et al. 2016], como roubo de senhas de acesso ao serviço de nuvem e falhas de segurança na interface de programação (API) fornecida pelo provedor [Younis et al. 2013].

Em certas aplicações, o cuidado com a confidencialidade dos dados certamente precisará ser maior, que é o caso dos sistemas elétricos e redes inteligentes. Nesse contexto, dados sensíveis, como o gasto de energia de cada consumidor, devem ser tratados com cuidado, já que podem revelar muita informação sobre os consumidores, como seus padrões de comportamento, por exemplo, trechos do dia em que não há indivíduos na residência, horas de chegada e saída ou de repouso.

Este artigo propõe uma arquitetura de software para viabilizar o uso da computação na nuvem em aplicações com requisitos estritos de segurança e privacidade. Tal arquitetura considera como os componentes de uma aplicação de redes elétricas inteligentes (*smart grids*) podem ser integrados de forma que a privacidade e segurança dos dados dos usuários sejam garantidas. A parte sensível do processamento é então isolada e a arquitetura considera diferentes estratégias para agregação de dados sensíveis em ambientes onde não há garantia de plena confidencialidade.

Para ilustrar duas implementações do módulo de agregação segura, duas alternativas foram desenvolvidas, uma baseada em criptografia homomórfica, que é completamente programada em software, mas que impõe altos custos adicionais de processamento, e outra baseada na tecnologia Intel SGX [McKeen et al. 2013], que não impõe custos computacionais elevados, mas requer a disponibilidade de processadores com suporte a esta tecnologia nas máquinas onde forem executados os agregadores[1]. A primeira permite que computações sejam realizadas em cifrotextos sem comprometer a criptografia. Por exemplo, um sistema de busca homomórfico permite que termos sejam localizados sem que o computador tenha conhecimento sobre o que se deseja encontrar ou sobre a base de dados onde se realizou a procura. Já a segunda possibilita que processos possam ser executados em um modo protegido, onde toda a memória e execução são protegidas contra acessos, mesmo que de usuários ou processos com níveis maiores de privilégio, evitando a necessidade de realizar computações em cifrotextos.

Nos nossos experimentos, dois casos de uso foram abordados: o cálculo do consumo de energia em uma região e o cálculo da conta mensal de um consumidor. A estratégia de agregação com Intel SGX – pela primeira vez utilizada em uma plataforma de gerência de computação na nuvem – se mostrou muito mais eficiente do que a criptografia homomórfica. Todavia, como a criptografia homomórfica não possui exigências de hardware específico, essa estratégia pode ser viável para aplicações cujo foco não esteja no volume de dados.

Finalmente, certas limitações foram identificadas durante o desenvolvimento do trabalho, dentre elas: (i) para ser viável, o conjunto de computações para a criptografia homomórfica é limitado, o que dificulta ou inviabiliza o desenvolvimento de módulos de processamento de dados arbitrários; (ii) o uso de Intel SGX impede que certas operações sejam realizadas, como chamadas de sistema (*syscalls*), o que tem implicações no tipo de

[1]Alguns exemplos de processadores com suporte a Intel SGX são os processadores da sexta geração da família Intel Core e alguns processadores Xeon recentes, como os da família E3-1200.

código que executará de forma protegida; e, (iii) Intel SGX também possui limitações de uso de memória, $128\ MB$ na implementação atual, mas com possibilidade de páginação em uma versão lançada recentemente.

O restante do artigo é organizado da seguinte maneira. Os trabalhos relacionados são discutidos na Seção 2. A Seção 3 retrata a arquitetura de software elaborada para garantia de agregação segura de dados. Os experimentos e resultados são apresentados na Seção 4 e são discutidos na Seção 5. Por fim, a Seção 6 ressalta as principais conclusões do artigo, assim como suas limitações e possíveis trabalhos futuros.

2. Trabalhos Relacionados

Em [Reinhold et al. 2014] é descrita uma arquitetura híbrida onde parte dos componentes se encontra em uma nuvem privada e parte em uma nuvem pública. A parte de armazenamento dos dados fica em uma nuvem pública, de forma criptografada. Já a aplicação, com a lógica e processamento dos dados, se encontra em uma nuvem privada, onde há menores preocupações com segurança e privacidade. Somente os clientes têm acesso às chaves privadas, sendo assim, os dados só são tratados na sua forma pura quando o cliente está autenticado. Por fim, sempre que um dado precisa ser armazenado, ele passa por um servidor de criptografia, que fica na nuvem privada, e então é enviado para o servidor de armazenamento na nuvem pública.

[Bohli et al. 2013] traz um estudo com diferentes padrões de arquitetura para distribuição dos recursos em múltiplos provedores de serviços de computação na nuvem: replicação de aplicações, que permite enviar operações em distintas nuvens e comparar se os resultados são os mesmos; partição das camadas de uma aplicação em distintas nuvens, para separar, por exemplo, a lógica da aplicação em uma nuvem e o banco de dados em outra; partição da lógica da aplicação em diferentes nuvens e partição dos dados em nuvens distintas. Os autores concluem que não existe estratégia ótima para todos os casos, já que a implementação dos padrões sugeridos não é trivial e todas possuem falhas de segurança.

Possíveis usos de Intel SGX foram discutidos em [Hoekstra et al. 2013], onde foram apresentados exemplos de aplicações que fazem uso das capacidades de Intel SGX, bem como foi apresentada uma arquitetura de aplicações considerando uma partição da aplicação entre partes que demandam segurança, devendo ser executadas dentro de enclaves, e partes que não necessitam de segurança, que pode ser executado fora de enclaves.

[Barbosa et al. 2016] apresenta um conjunto de ferramentas para a construção de protocolos que ampliam as garantias de ambientes de execução isolados, como é o caso do Intel SGX, através do uso de sua capacidade de realizar atestação remota. Com isso, é possível estabelecer protocolos de troca de chave entre um participante remoto e um ambiente de execução isolado, de forma segura. Esses protocolos são definidos a partir da combinação entre um protocolo de troca de chaves passivamente seguro e o uso de um protocolo de atestação arbitrário.

Sobre computação segura, Rivest, Adleman e Dertouzous [Rivest et al. 1978a] criaram o conceito de criptografia homomórfica. Isto se deve ao fato do RSA, sistema criptográfico desenvolvido por Rivest, Shamir e Adleman [Rivest et al. 1978b], possuir um homomorfismo parcial, chamado de homomorfismo multiplicativo. Ou seja, com

o RSA é possível multiplicar dois valores criptografados e o resultado ainda será a multiplicação criptografada.

Modelos completamente homomórficos se caracterizam por permitir operações de adição e multiplicação em blocos criptografados, de modo que o valor retornado seja uma encriptação do resultado das operações aplicadas sobre os dados originais. Embora o conceito não seja recente, estes modelos foram considerados puramente teóricos até que [Gentry 2009] propuseram um sistema válido, usando reticulados ideais. Entretanto, questões relacionadas à eficiência ainda constituem uma barreira. Atualmente, todos os tipos de esquemas de criptografia completamente homomórfica propostos ainda possuem um longo caminho de evolução antes que sejam utilizados na prática [Naehrig et al. 2011].

3. Solução Proposta

A arquitetura de software proposta possui quatro tipos de componentes: barramentos de mensagens, produtores, agregadores e consumidores. Os barramentos de mensagens são responsáveis pela comunicação entre os produtores, agregadores e consumidores dos dados. Após serem produzidos, esses dados serão publicados no barramento de mensagens e, em algum momento, serão consumido e tratados por agregadores, que podem realizar operações arbitrárias, mas que devem ser capazes de executá-las de forma segura. Posteriormente, os dados agregados serão consumidos por aplicações. Um esquema ilustrativo da arquitetura pode ser visto a seguir na Figura 1.

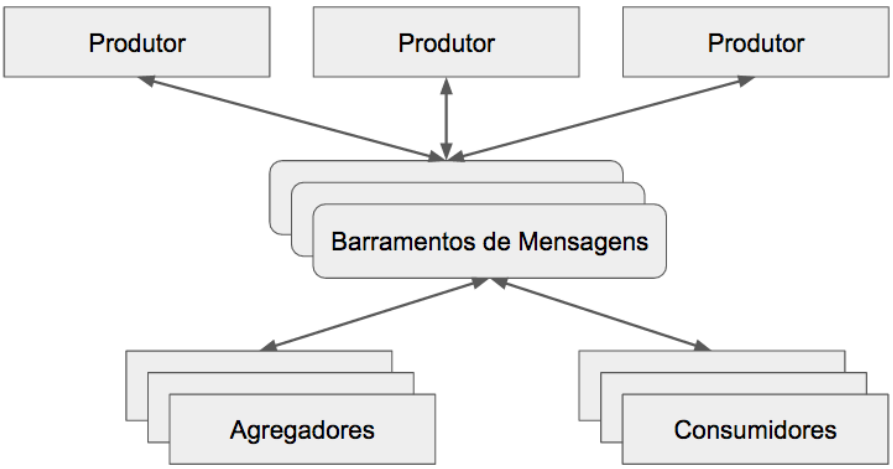

Figura 1. Esquema simplificado da arquitetura de software proposta

É importante salientar que todos os componentes ilustrados seguem interfaces bem definidas, ou seja, é possível utilizar diferentes tipos de produtores, barramentos de mensagens, agregadores ou consumidores desde que eles implementem corretamente a especificação. As seções a seguir detalham os principais componentes dessa arquitetura.

3.1. Barramentos de mensagens

Um barramento de mensagem é responsável pela troca de informações entre produtores, agregadores e consumidores de forma transparente, ou seja, um produtor pode, por exemplo, criar mensagens sem saber maiores detalhes, como localização física ou endereço IP,

sobre os agregadores. Com isso, é possível reduzir o acoplamento e garantir escalabilidade.

Cada componente da arquitetura deve se inscrever em um tópico de mensagens. Todos os agregadores ou consumidores de um tópico receberão as mensagens enviadas pelos produtores. Ademais, é permitido o consumo exclusivo de novas mensagens ou de todas as mensagens retidas de um certo tópico. É valido frisar também que qualquer parte pode enviar mensagens em um tópico a fim de requisitar dados, de realizar passos intermediários em uma ação e outros. Por fim, a quantidade de mensagens armazenadas para um tópico pode ser configurável.

Por ser crítico na comunicação entre todos os envolvidos, esse componente permite a troca segura de mensagens. Cada envolvido pode possuir um certificado emitido por uma unidade certificadora, como forma de autenticação, evitando que intrusos comecem a enviar ou receber mensagens. Além disso, mensagens com conteúdo sensível devem estar criptografadas. É papel do agregador, descrito a seguir, transformar informação sensível e criptografada, como consumo instantâneo de uma residência, em informação agregada e em purotexto, como o consumo em um intervalo maior de tempo ou o consumo de uma região.

3.2. Agregadores

Após o consumo dos dados de um tópico comum, é necessário realizar operações nos mesmos. Tais operações, como, por exemplo, soma, multiplicação ou agrupamento são realizadas pelos agregadores. Dois agregadores são exemplificados na nossa implementação da arquitetura: um agregador **homomórfico** e um agregador baseado em **Intel SGX**. Ambos são projetados para realizar **operações de soma** e serão detalhados nas seções a seguir.

3.2.1. Agregador Homomórfico

Esse agregador faz uso da criptografia homomórfica para realizar operações de forma segura e privada nos dados. A abordagem utilizada por esse componente é baseada no esquema proposto em [Busom et al. 2016], que faz uso do sistema de criptografia de ElGamal [ElGamal 1984], possuidor da propriedade homomórfica multiplicativa mas com possibilidade de adição, como pode ser visto em [Cramer et al. 1997]. Além disso, é possível combinar com um esquema de criptografia de limiar [Saroj et al. 2015], em que há necessidade da colaboração de todas as partes envolvidas para decriptar cifrotextos.

Para que esse tipo de agregação funcione é necessário que cada produtor $p \in [1, n]$ de um tópico comum possua os seguintes itens:

- Um número primo grande q (de pelo menos 2048 bits) e um gerador g de ordem q do grupo multiplicativo G de $\mathbb{Z}_q^*$;
- Uma chave privada x_p;
- Uma chave pública $y_p = g^{x_p}$;
- Um certificado $cert_p$ a ser validado por uma unidade certificadora;

A fim de permitir o envio das informações é obrigatória uma fase de configuração sempre que novos produtores se inscreverem em um tópico de mensagens. O procedimento é descrito a seguir:

1. O agregador envia uma mensagem de requerimento de configuração;
2. Cada produtor envia y_p e $cert_p$ ao tópico;
3. O agregador verifica a validade de cada $cert_p$ e insere uma mensagem com $\{y_1, ..., y_n\}$ e $\{cert_1, ..., cert_n\}$ no tópico;
4. Cada produtor verifica a validade de cada $cert_p$ e calcula uma chave pública global $y = \prod_{p=1}^{n} y_p$.

Os seguintes passos serão realizados a cada instante de tempo que for necessária uma coleta de dados para agregação:

1. O agregador envia uma mensagem de requerimento de dados ou, alternativamente, os produtores podem iniciar uma transmissão periodicamente;
2. Cada produtor p gera um número aleatório $z_p \in \mathbb{Z}_q^*$ e calcula $C_p = E_y(g^{v_p + z_p}) = (c_p, d_p)$, em que v_p representa o valor coletado por p e a função E_y é a função de criptografia de ElGamal;
3. Todos os valores C_p são publicados no tópico do barramento de mensagens associado àquele produtor (por exemplo, a região em que o medidor está instalado);
4. O agregador realiza sua computação: $C = \left(\prod_{p=1}^{n} c_p, \prod_{p=1}^{n} d_p \right)$ e insere C no tópico;
5. Cada produtor calcula $T_p = c^{x_p}.g^{z_p}$ e publica-o no tópico;
6. O agregador pode, então, obter $D = d.\left(\prod_{p=1}^{n} T_p \right)^{-1}$, em que $d = \prod_{p=1}^{n} d_p$;
7. Por fim, é possível obter $V = \sum_{p=1}^{n} v_p$ ao calcular $\log_g D$;
8. O agregador publica o resultado obtido no mesmo ou em outro tópico do barramento de mensagens, de forma que ele fique disponível para possíveis consumidores.

3.2.2. Agregador Intel SGX

O agregador Intel SGX faz uso da tecnologia de mesmo nome para prover segurança e privacidade dos dados sensíveis sendo agregados, através do uso de áreas protegidas de memória (**enclaves**), inacessíveis até mesmo por usuários com mais privilégios. Em nossa implementação, usamos o algoritmo de criptografia simétrica *AES Galois/Counter Mode* (AES-GCM) descrito em [Dworkin 2007], com chave de tamanho de 128 bits, para a troca de mensagens confidenciais entre os produtores e o agregador. Este algoritmo permite verificar a autenticidade dos dados confidenciais recebidos, através da detecção de modificações não intencionais ou modificações intencionais não autorizadas feitas aos dados, bem como torna a comunicação imune a ataques do tipo *side-channel* baseados em *software*.

Ao receber dados confidenciais encriptados, o agregador pode decriptar a mensagem usando o mesmo algoritmo AES-GCM, e após isso realizar a agregação sobre os dados confidenciais decriptados. A segurança e privacidade dos dados decriptados é alcançada através das garantias providas pelos enclaves SGX [McKeen et al. 2013].

As seguintes condições precisam ser satisfeitas para o correto funcionamento deste agregador:

- Cada produtor $p \in [1, n]$ de um tópico possui uma chave simétrica k_p;
- O agregador conhece de antemão cada uma das k_p chaves;

Para que o agregador conheça de antemão cada uma das k_p chaves, é necessária uma troca de chaves através de um protocolo seguro, como o provido pelo próprio Intel SGX, descrito em [Anati et al. 2013], onde os produtores possam atestar que estão se comunicando com o agregador correto. A atestação também será usada para validar o agregador, por exemplo, para assegurar que este não vazará informações (por exemplo, permitindo agregações com poucas medições criptografadas).

Para agregar uma coleta de dados feita em um instante de tempo, os seguintes passos serão realizados:

1. O agregador envia uma mensagem de requerimento de dados ou, alternativamente, os produtores podem iniciar uma transmissão periodicamente;
2. Cada produtor p cria um valor aleatório (***nonce***) n_p, e calcula $C_p, M_p = G_e(v_p, n_p, k_p)$, onde C_p é o valor coletado por p após a encriptação AES-GCM, M_p é o código de autenticação de mensagem de v_p, onde v_p é o valor medido por p, e a função G_e é a função de criptografia AES-GCM no modo de encriptação.
3. Cada produtor publica C_p, M_p, e n_p ao tópico;
4. O agregador obtém $V = \sum_{p=1}^{n} v_p$ ao calcular $\sum_{p=1}^{n} G_d(C_p, n_p, k_p, M_p)$, onde a função G_d é a função AES-GCM no modo decriptação, ao mesmo tempo em que verifica a integridade de cada uma das mensagens recebidas no tópico;
5. O agregador publica o resultado obtido no mesmo ou em outro tópico do barramento de mensagens, de forma que ele fique disponível para possíveis consumidores.

4. Avaliação da arquitetura proposta

Com a finalidade de avaliar a solução proposta, dois casos de uso foram escolhidos para a implementação de provas de conceito para futuros experimentos. Ambos fazem parte do contexto de redes elétricas inteligentes e exigem cautela com relação à confidencialidade dos dados. Os detalhes da implementação das duas aplicações serão discutidos na seção 4.3.

4.1. Cálculo de consumo de energia em regiões

O crescimento das necessidades de recursos de energia elétrica motivou tanto o governo quanto a indústria a buscar formas alternativas de prover energia e, principalmente, de aperfeiçoar o gerenciamento da rede elétrica. Por outro lado, aumentar a eficiência e balancear a malha energética não é uma tarefa trivial. Para isso, uma opção é utilizar medidores inteligentes (*smart meters*) que podem, periodicamente, medir e reportar o consumo energético [Erkin and Tsudik 2012].

Como dito anteriormente, a medição periódica do consumo energético causa preocupações com a privacidade dos consumidores, já que é possível inferir informações pessoais a partir do que é coletado, como tipos de equipamentos na residência, assim

como a presença e o número de moradores [Anderson and Fuloria 2010]. Em casos mais extremos, é possível identificar o canal de televisão sendo assistido [Greveler et al. 2012].

Diante do exposto, uma alternativa para fornecer informações cruciais para o balanceamento energético e, ao mesmo tempo, manter a privacidade dos consumidores é agregar os dados de consumo em grupos de residências, ou regiões.

4.2. Cálculo da conta mensal dos consumidores

O segundo caso de uso consiste em calcular, de forma segura e privada, a conta mensal de cada consumidor a partir de um conjunto de dados de consumo em intervalos curtos. Esta abordagem permite que a concessionária emita as faturas no intervalo de cobrança e que ela ou o consumidor possam usufruir de certos benefícios da medição detalhada (por exemplo, calculo do consumo instantâneo da região para concessionária ou visualização local do consumo instantâneo pelo consumidor) sem que haja risco de que os dados detalhados sejam usados pela concessionária ou parceiros para inferir hábitos do consumidor.

4.3. Detalhes técnicos das provas de conceito

Com o intuito de executar experimentos, as provas de conceito foram implementadas seguindo a arquitetura explicada na Seção 3. A Figura 2 ilustra a esquemática das aplicações desenvolvidas.

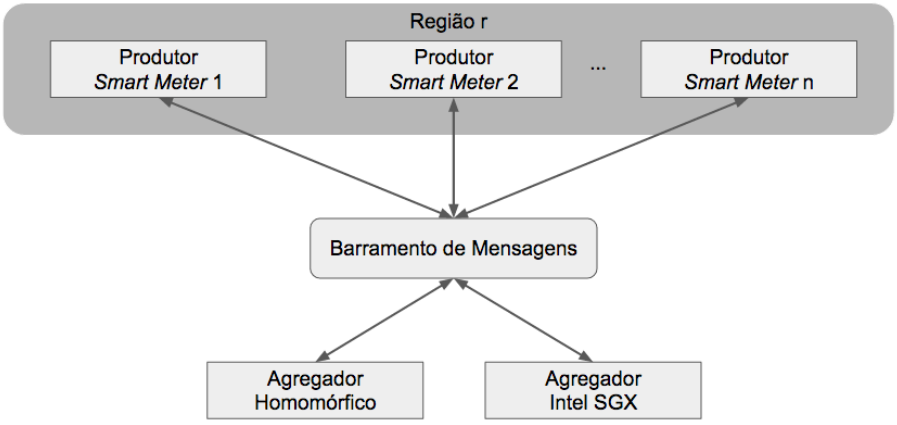

Figura 2. Esquema da aplicação para cálculo de consumo de energia por região

Dentre os componentes acima, os medidores inteligentes (produtores) e os agregadores foram desenvolvidos durante a pesquisa realizada. Para a comunicação entre produtores e agregadores/consumidores, a solução Apache Kafka[2] foi escolhida por prover todos os serviços exigidos para o barramento de mensagens, inclusive a troca segura de mensagens, permitindo a autenticação dos envolvidos e o uso de canais seguros de comunicação.

No experimento realizado, cada medidor inteligente é simulado por uma *thread* em Java que gera valores de consumo e publica no barramento de mensagens. Todos os medidores de uma região r estão inscritos no mesmo tópico do barramento de mensagens. O agregador da região r é um processo em execução que também está inscrito no tópico

[2]http://kafka.apache.org

em questão e é responsável por agrupar as medições para cada instante de tempo t e, quando os dados de consumo do conjunto de medidores da região são coletados para um instante t, é possível calcular a soma desses valores.

Para o cálculo da conta mensal com o agregador homomórfico, também desenvolvido em Java, cada medidor faz uso de k pares de chaves públicas e privadas, um par para cada medição, com um reaproveitamento de forma circular, de acordo com o módulo do identificador da medição. Por fim, a cada fim de ciclo, o processo de agregação é executado para aquele intervalo de tempo.

No caso do agregador Intel SGX, apenas uma chave simétrica fixa, previamente compartilhada com o agregador, é usada por cada cada medidor, além de um *nonce* gerado aleatoriamente à cada medição. Para efeito de simplificação, a troca de chaves de forma segura entre medidores e agregador foi abstraída, e pode ser considerada um problema resolvido na fase de atestação, como mostrado em [Barbosa et al. 2016].

Finalmente, as duas abordagens são robustas contra situações onde o agregador publica resultados agregados mas que permitem a inferência de valores instantâneos. Na abordagem homomórfica, o produtor participa do processo de agregação e pode identificar situações onde as agregações usam poucos dados ou possuem intervalos com sobreposição. Na abordagem Intel SGX, o produtor validou o código do atestador inicialmente e este não pode ser atualizado sem uma nova atestação.

4.4. Experimentos e Resultados

Após a implementação das provas de conceito, as mesmas foram submetidas a experimentos com o objetivo de mensurar e comparar os tempos de resposta para realizar a agregação com criptografia homomórfica e com Intel SGX, bem como sem o uso dessas duas abordagens (sem garantias de segurança e privacidade)

Os testes foram realizados em um ambiente de nuvem privada utilizando OpenStack para orquestração da aplicação na nuvem com contêineres Docker (usando o driver *nova-docker* do OpenStack). Os contêineres executavam Ubuntu Linux 14.04 e os drivers do Intel SGX[3]. De acordo com nossas pesquisas, este é o primeiro trabalho a aplicar uma estratégia com Intel SGX em ambientes de computação na nuvem. Na Figura 3 é possível visualizar como estão dispostos os componentes para o experimento.

Dois conjuntos de testes foram efetuados. O primeiro foi realizado para comparar as agregações utilizando Intel SGX e criptografia homomórfica. Nele, para cada caso de uso, 10 repetições foram realizadas por agregador para regiões com 10, 50, 100 e 200 medidores, resultando em 160 execuções. O segundo teve como objetivo comparar agregações com Intel SGX a agregações totalmente sem criptografia, a fim de avaliar o custo do uso dessa nova tecnologia. Neste segundo conjunto, a mesma quantidade de repetições foi feita para regiões com 200, 400, 800 e 1000 medidores.

As médias dos tempos de resposta podem ser lidos nas Tabelas 1 e 2 e visualizados nas Figuras 4 e 5. Para o consumo por região, o tempo exibido é relativo ao cálculo do consumo da região inteira para um instante de tempo t. Já para a conta mensal, esse é o tempo levado para o cálculo do consumo do mês inteiro com medições a cada 15 minutos.

[3]https://01.org/intel-softwareguard-eXtensions (desde 24 de junho de 2016).

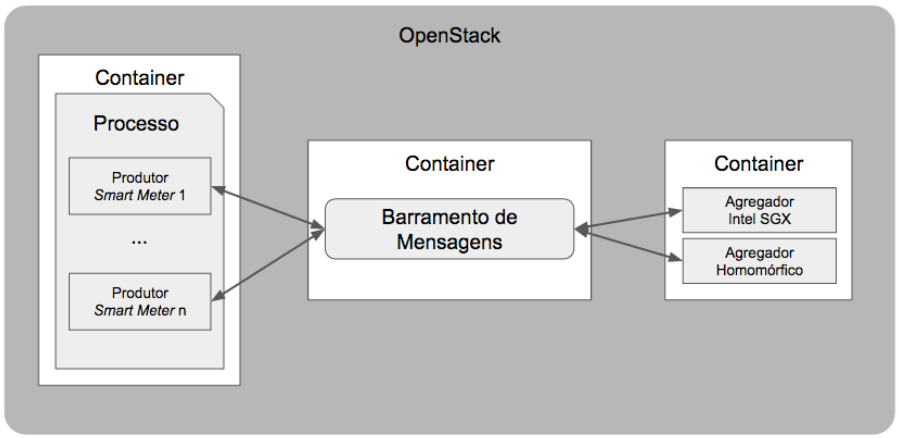

Figura 3. Topologia dos componentes para os experimentos

Caso de Uso	Agregador	$n = 10$	$n = 50$	$n = 100$	$n = 200$
Consumo por região	Homomórfico	362 ms	1812 ms	3755 ms	7596 ms
Consumo por região	Intel SGX	116 ms	117 ms	119 ms	121 ms
Conta mensal	Homomórfico	669 s	3167 s	6220 s	12305 s
Conta mensal	Intel SGX	70 s	74 s	78 s	84 s

Tabela 1. Médias dos tempos de resposta obtidos para comparação entre Intel SGX e criptografia homomórfica

Para os tempos acima, foram considerados os tempos totais, incluindo as comunicações entre os componentes (na rede local), assim como os tempos gastos para criptografia dos dados pelos produtores e descriptografia pelos agregadores Intel SGX.

5. Discussão

Como pode ser observado nos gráficos e na tabela da seção anterior, a abordagem de agregação utilizando Intel SGX possui tempos de resposta bem menores que a agregação com criptografia homomórfica. Não obstante, cada forma de agregação fornece vantagens e desvantagens que serão discutidas a seguir.

A criptografia homomórfica, embora mais lenta, possui vantagens que a torna viável em certas aplicações. Não há necessidade de hardware específico para sua execução, tornando-a facilmente implantável em vários ambientes, além disso, pode ser implementada em diversas linguagens, já que sua base é puramente matemática. Dependendo do tamanho da massa de dados a ser computada, o custo em termos de tempo pode ser irrisório diante dos benefícios de privacidade e segurança que essa abordagem traz consigo, já que todas as computações são realizadas em cifrotextos, fazendo com que o conteúdo que é processado seja desconhecido ao provedor de serviços e a quaisquer atacantes em potencial.

Outra vantagem visível na abordagem utilizada para a agregação homomórfica é que o consumidor dos dados não consegue inferir as medições de um consumidor específico, porque seria necessário que todos os outros consumidores fossem corruptos. Isso é característico da criptografia de limiar.

Caso de Uso	Agregador	$n = 200$	$n = 400$	$n = 800$	$n = 1000$
Consumo por região	Sem criptografia	116 ms	119 ms	119 ms	121 ms
Consumo por região	Intel SGX	122 ms	125 ms	134 ms	139 ms
Conta mensal	Sem criptografia	73 s	74 s	74 s	78 s
Conta mensal	Intel SGX	84 s	96 s	115 s	126 s

Tabela 2. Médias dos tempos de resposta obtidos para comparação entre agregação com Intel SGX e nenhuma criptografia

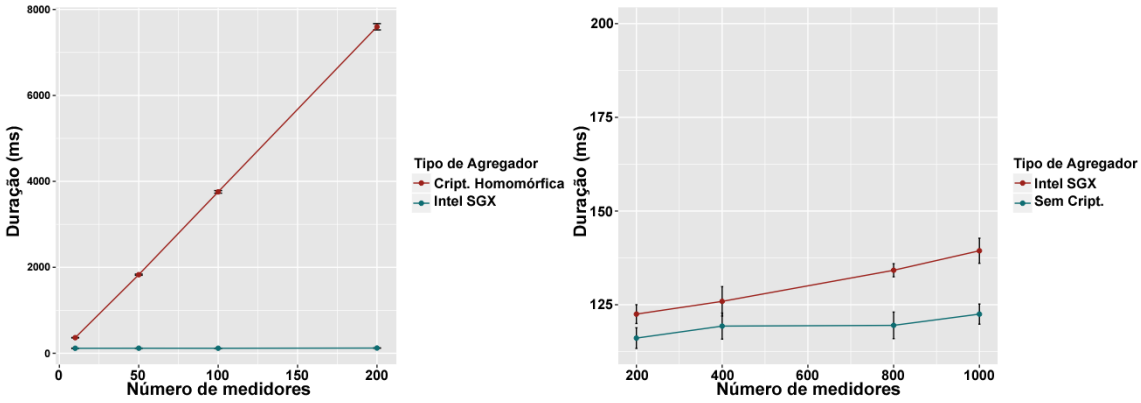

Figura 4. Gráficos do experimento realizado para o cálculo do consumo de uma região

Por outro lado, a técnica homomórfica possui uma limitação relacionada às operações que podem ser realizadas em cifrotextos, já que a maioria dos algoritmos de criptografia são parcialmente homomórficos, e, abordagens que permitem computações arbitrárias em cifrotextos, como pode ser visto em [Gentry 2009], ainda possuem tempos de processamento muito altos.

Outro fator negativo que foi observado na agregação homomórfica é que há necessidade da troca de mensagens adicionais para que o consumo regional seja computado. Isso não é trivialmente implantável, já que o envio da mensagem da concessionária para os medidores obriga que um medidor possua um endereço acessível ou que ele utilize uma estratégia de *polling* para verificar periodicamente se há novas mensagens, como as da segunda fase do algoritmo.

O Intel SGX, por sua vez, possui uma alta performance e baixo custo adicional, se comparado ao uso de criptografia homomórfica, por não precisar realizar computações sempre sobre cifrotextos, ao mesmo tempo que provê garantias de segurança e privacidade de dados de usuários, sem que haja um grande esforço para implementar aplicações que usem a tecnologia, já que o Intel SGX possui bibliotecas que dão suporte à criação, atestação e uso de enclaves. Dessa forma, é interessante o seu uso quando na presença de *hardware* com o devido suporte, e houver necessidade velocidade de execução e segurança e privacidade dos dados envolvidos.

Apesar disso, ainda há algumas questões em aberto quanto à completa integração entre a solução Intel SGX e a filosofia do ambiente de computação na nuvem. Enquanto as soluções de computação na nuvem baseiam-se no desacoplamente entre o hardware físico e as aplicações, por exemplo, através de virtualização do hardware, Intel SGX vincula a

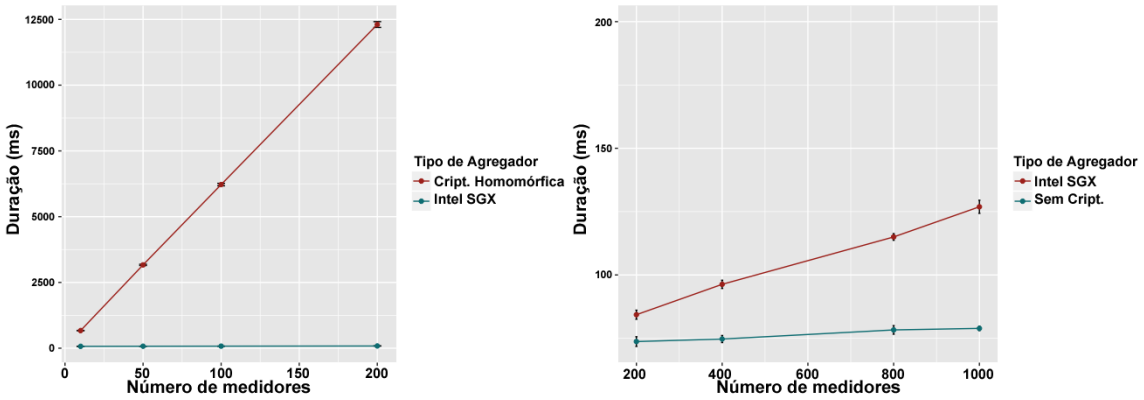

Figura 5. Gráficos do experimento realizado para o cálculo da conta mensal

segurança da aplicação diretamente ao processo de atestação, que, por enquanto, exige a participação da Intel e seu serviço de atestação.

Um outro ponto negativo quanto ao uso de Intel SGX é a limitação quanto à API suportada atualmente, que exige o desenvolvimento de aplicações em C/C++, e a impossibilidade, por motivos de segurança, de executar chamadas de sistema em código executado dentro de enclaves, o que pode dificultar a portabilidade de aplicações para o uso da tecnologia. Por fim, um outro ponto que deve ser levado em conta em qualquer aplicação que faça uso de Intel SGX é o do atual limite de apenas $128\ MB$ de memória que pode usado para a criação e execução de enclaves por *hardware* hospedeiro. Paginação pode ser usada para criptografar a memória antes de exportá-la para fora do processador, mas isso impõe cargas adicionais.

6. Conclusões

Requisitos como segurança e privacidade não devem ser ignorados quando aplicações com dados sigilosos fazem uso da computação na nuvem. Nesse artigo, descrevemos uma arquitetura de software para endereçar tais requisitos. Essa arquitetura permite o uso de diferentes estratégias para a agregação de dados privados. Tais estratégias incluem o uso de criptografia homomórfica ou de tecnologias como Intel SGX.

Para a avaliação, identificamos dois casos de uso em que houvesse preocupações com os requisitos citados. De posse dos casos de uso, foram desenvolvidas provas de conceito que ajudaram a identificar as vantagens e desvantagens de cada forma de agregação. Para criptografia homomórfica, a principal vantagem identificada foi a implantação viável em quaisquer ambientes, embora seja menos eficiente. Já o Intel SGX, que foi pela primeira vez utilizado em um orquestrador de computação na nuvem, possui tempos de resposta bem menores e permite realizar diversas formas de computação sobre os dados, mas exige uma infraestrutura específica do provedor de serviços.

Para trabalhos futuros, almeja-se utilizar a criptografia de ElGamal com outros grupos, como, por exemplo, curvas elípticas, assim como o uso de *garbled circuits* [Kolesnikov and Schneider 2008] e também computação segura multi-parte [Cramer et al. 2000]. Usando Intel SGX, deseja-se propor soluções mais completas, que envolvam a implementação de atestação remota [Barbosa et al. 2016] de forma amigável em um ambiente de nuvem.

7. Agradecimentos

Este trabalho foi parcialmente financiado pelo projeto EU-BRA SecureCloud (MCTI/RNP 3a Chamada Coordenada) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

Anati, I., Gueron, S., Johnson, S., and Scarlata, V. (2013). Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, volume 13.

Anderson, R. and Fuloria, S. (2010). On the security economics of electricity metering. In *WEIS*. Citeseer.

Barbosa, M., Portela, B., Scerri, G., and Warinschi, B. (2016). Foundations of hardware-based attested computation and application to sgx. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 245–260. IEEE.

Bohli, J.-M., Gruschka, N., Jensen, M., Iacono, L. L., and Marnau, N. (2013). Security and privacy-enhancing multicloud architectures. *IEEE Transactions on dependable and secure computing*, 10(4):212–224.

Busom, N., Petrlic, R., Sebé, F., Sorge, C., and Valls, M. (2016). Efficient smart metering based on homomorphic encryption. *Computer Communications*, 82:95–101.

CIF (2015). Uk cloud adoption snapshot & trends for 2016: The business case for cloud.

Cramer, R., Damgård, I., and Maurer, U. (2000). General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 316–334. Springer.

Cramer, R., Gennaro, R., and Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490.

Dworkin, M. J. (2007). Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. Technical report, Gaithersburg, MD, United States.

ElGamal, T. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 10–18. Springer.

Erkin, Z. and Tsudik, G. (2012). Private computation of spatial and temporal power consumption with smart meters. *Proceedings of the 10th international conference on Applied Cryptography and Network Security*, pages 561–577.

Gentry, C. (2009). *A fully homomorphic encryption scheme*. PhD thesis, Stanford University.

Greveler, U., Glösekötterz, P., Justusy, B., and Loehr, D. (2012). Multimedia content identification through smart meter power usage profiles. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., and Del Cuvillo, J. (2013). Using innovative instructions to create trustworthy software solutions. In *HASP@ ISCA*, page 11.

Kolesnikov, V. and Schneider, T. (2008). Improved garbled circuit: Free xor gates and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 486–498. Springer.

Markovic, D. S., Zivkovic, D., Branovic, I., Popovic, R., and Cvetkovic, D. (2013). Smart power grid and cloud computing. *Renewable and Sustainable Energy Reviews*, 24:566–577.

McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. (2013). Innovative instructions and software model for isolated execution. In *HASP@ ISCA*, page 10.

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM.

Pasupuleti, S. K., Ramalingam, S., and Buyya, R. (2016). An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing. *Journal of Network and Computer Applications*, 64:12–22.

Reinhold, P., Benn, W., Krause, B., Goetz, F., and Labudde, D. (2014). Hybrid cloud architecture for software-as-a-service provider to achieve higher privacy and decrease security concerns about cloud computing. In *Conf. Cloud Computing, GRIDs, and Virtualization (IEEE, 2014)*, pages 94–99. Citeseer.

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978a). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

Rivest, R. L., Shamir, A., and Adleman, L. (1978b). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Saroj, S. K., Chauhan, S. K., Sharma, A. K., and Vats, S. (2015). Threshold cryptography based data security in cloud computing. In *Computational Intelligence & Communication Technology (CICT), 2015 IEEE International Conference on*, pages 202–207. IEEE.

Younis, Y. A., Merabti, M., and Kifayat, K. (2013). Secure Cloud Computing for Critical Infrastructure : A Survey.

