Uma Avaliação de Algoritmos Criptográficos em Redes IEC 61850: Uma Abordagem Prática*

Rafael B. Scarselli[1], **Leonardo F. Soares**[1], **Igor M. Moraes**[1]

[1]Laboratório MidiaCom, PGC - TCC
Instituto de Computação – Universidade Federal Fluminense
Niterói – RJ – Brasil

{rscarselli,leonardofiorio}@id.uff.br, igor@ic.uff.br

Abstract. *This paper evaluates cryptographic algorithms applied to the GOOSE protocol in IEC 61850 communication networks for electrical substations. The IEC suggests the use of RSA for digital signature of GOOSE messages and at the same time defines a maximum communication latency of 3 ms for critical messages. Through practical experiments with devices with low computational power, the inviability of the RSA suggested by the IEC is confirmed. Results show that the AES symmetric cryptography algorithm with the CMAC technique meets the time constraints defined by IEC, even when the entire payload of a 459-byte packet is encrypted.*

Resumo. *Este artigo avalia algoritmos criptográficos aplicados ao protocolo GOOSE em redes de comunicação IEC 61850 para subestações de energia elétrica. A IEC sugere o uso do RSA para assinatura digital das mensagens GOOSE e ao mesmo tempo define uma latência máxima de comunicação de 3 ms para mensagens críticas. Através de experimentos práticos com dispositivos de baixo poder computacional, confirma-se a inviabilidade do RSA sugerido pela IEC. Os resultados mostram que o algoritmo de criptografia simétrica AES com a técnica CMAC atende às restrições de tempo definidas pela IEC, mesmo quando se cifra toda a carga útil de um pacote de 459 bytes.*

1. Introdução

As redes elétricas inteligentes (*smart grids*) estão em franca implantação em todo o mundo [Farhangi 2010]. Um Sistema Elétrico de Potência (SEP) [ABNT5460 1992] é a infraestrutura de geração, transmissão e distribuição de energia elétrica. Um SEP pode fazer parte de uma grande rede nacional e, por isso, o controle e gerenciamento é fundamental para manutenção do sistema. Uma parte fundamental de um SEP são as subestações de energia elétrica que estão sob o domínio de alguma concessionária de energia. É a partir das subestações que a energia é distribuída por bairros e regiões e chega às casas dos consumidores. O gerenciamento e controle das subestações é feito remotamente pela concessionária através de redes de comunicação privada e restritas aos administradores do sistema, sem qualquer interferência externa.

A norma IEC 61850 (*International Electrotechnical Commission*) define um padrão para comunicação de dispositivos eletrônicos inteligentes (IEDs, *Intelligent Electronic Device*) em subestações de energia elétrica. Os IEDs são relés digitais que permitem

*Este trabalho tem suporte financeiro da TAESA/ANEEL, CNPq, CAPES, FAPERJ e FAPESP.

implementação de lógicas para chaveamento, bloqueio e abertura. A comunicação entre os IEDs é responsável pela automação e controle de toda a subestação e também pelo gerenciamento feito pela concessionária. É através da rede de comunicação entre os IEDs que as falhas são contidas localmente na subestação, não permitindo que se espalhem em um efeito avalanche por toda a rede nacional de energia elétrica.

Os IEDs de uma subestação de energia se comunicam com outros IEDs através do protocolo de comunicação *Generic Object Oriented Substation Event* (GOOSE) [IEC61850 2003]. O GOOSE é implementado diretamente na camada de enlace da pilha de protocolos TCP/IP, pois suas mensagens transportam conteúdos genéricos e possuem requisitos estritos de tempo. Essa é a razão pela qual os IEDs enviam mensagens GOOSE. Para controlar os níveis desejados de tensão e corrente na subestação, os IEDs podem emitir comandos de controle como um disjuntor, por exemplo, caso detectem anomalias de corrente, tensão ou frequência oriundas dos sensores e equipamentos de energia que são conectados a eles. Estes equipamentos de padrão industrial são desenvolvidos com pequena capacidade de processamento, apenas o suficiente para suas funções básicas. Um IED comercial lançado em 2009 para redes IEC 61850 pela empresa ABB da série 670, por exemplo, possui processador IBM 3200PowerPC 750FX de 600 MHz [ABB 2009].

A norma IEC 61850 define um requisito de 3 ms para realização da comunicação entre IEDs com o protocolo GOOSE no requisito mais estrito, por isso ele é considerado um protocolo de tempo real *publisher/subscriber*. Isso justifica o fato do GOOSE ser implementado sobre a camada de enlace, diminui os tempos de geração, recebimento e processamento das mensagens, porém implica falta de segurança que poderia ser provida por camadas superiores da pilha TCP/IP. Isso se torna uma preocupação visto que não há controle de integridade e confidencialidade dos dados enviados através de mensagens GOOSE, facilitando ação de atacantes [Lopes et al. 2016]. As subestações tornaram-se mais conectadas às redes externas devido a novas demandas emergentes das *smart grids*, como por exemplo medidores inteligentes com comunicação bidirecional com a subestação, gerando potencial para atacantes externos à rede da concessionária.

A segurança da comunicação em tempo real de redes de automação de subestação de energia permanece um problema em aberto. Algoritmos criptográficos podem auxiliar na implementação segura do protocolo GOOSE. Porém, a aplicação desses algoritmos interfere nos tempos gastos para comunicação, pois são inseridos tempos de cifragem, decifragem e para cálculos de *hash*, por exemplo. Com a baixa capacidade de processamento dos dispositivos IEDs, a influência desses algoritmos pode ser ainda maior.

Este artigo tem o objetivo de avaliar algoritmos de criptografia aplicados ao protocolo GOOSE em uma bancada de testes experimental para verificar viabilidade da implementação dos algoritmos SHA256 (*Secure Hash Algorithm*), AES128 (*Advanced Encryption Standard*), RSA2048 (*Rivest-Shamir-Adleman*), CMAC (*Cipher-based Message Authentication Code*) e HMAC (*Hash MAC*) de acordo com o impacto nos tempos de comunicação com mensagens GOOSE encriptadas e o nível de segurança oferecido por cada algoritmo. Para isso, são implementados e executados os algoritmos de emissor e receptor de mensagens em Raspberry PI, configurados como um IED, a fim de comparar os resultados de tempo medidos com os requisitos estabelecidos pela norma IEC 61850 em um *hardware* com baixa capacidade computacional. Os resultados obtidos nos experimentos constatam a inviabilidade do uso do RSA recomendado pela norma IEC 62351.

Por outro lado, mostram que uma alternativa viável ao RSA é o algoritmo AES128 com segurança equivalente e desempenho de 475 μs de comunicação fim-a-fim, na técnica de código de autenticação, e 69 μs cifrando toda a carga útil de dados GOOSE. No caso do AES, há o desafio da distribuição de chaves que não é tratado neste artigo.

O restante desse artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados ao tema deste artigo, abordando principalmente o aspecto da segurança. A Seção 3 aborda as definições da norma IEC 61850 e 62351 e também traz informações de algoritmos criptográficos avaliados neste trabalho. A Seção 4 apresenta a metodologia empregada para avaliar experimentalmente os algoritmos criptográficos. Na Seção 5, a análise dos resultados é apresentada. Em seguida, na Seção 6 estão as conclusões deste artigo após os experimentos e também são propostos trabalhos futuros.

2. Trabalhos Relacionados

Existem diferentes trabalhos que abordam a segurança em redes de comunicação de subestações de energia [Wang and Lu 2013]. Um destes trabalhos [Hohlbaum et al. 2010] alerta que dispositivos de baixa capacidade de processamento, como os IEDs das subestações de energia, não são capazes de usar o algoritmo de chaves assimétricas RSA para assinar digitalmente mensagens GOOSE dentro dos requisitos de tempo da norma IEC 61850. Um IED comercial tipicamente possui um processador com relógio que vai de dezenas a poucas centenas de MHz de processamento [ABB 2009].

Novos estudos surgiram com propostas para atender aos requisitos de tempo da IEC 61850, desde então. Lu et. al. avaliam em um computador pessoal o tempo de processamento de algoritmos TESLA (*Timed Efficient Stream Loss-tolerant Authentication*) e HORS (*Hash to Obtain Random Subset*) junto com o RSA1024 [1] [Lu et al. 2012]. Nesta avaliação, varia-se o relógio da CPU do computador e apresenta-se qual o percentual de mensagens GOOSE transmitidas são entregues em menos de 3 ms. Os autores concluem que o RSA apresenta taxa de entrega inferior a 85% mesmo com uma CPU de 1,6 GHz.

Outro estudo [Yavuz 2014] propõe um novo esquema de autenticação *broadcast* com um protocolo de criptografia variante do RSA chamado RSA-condensado. A variante proposta é processada em menor tempo computacional, validada pelo experimento realizado em um computador portátil com processador Intel Core i7 de 1,6 GHz e 2 GB de memória RAM. O algoritmo RSA-condensado substitui uma operação de exponenciação por uma multiplicação, em comparação ao RSA. No cenário avaliado, não são especificados os detalhes das mensagens enviadas. Os resultados mostram um desempenho de 3792 μs para o RSA e de 226 μs para o RSA-condensado, considerando a média de tempo para enviar 10^4 mensagens. O trabalho comparou vários algoritmos (ECDSA, HORS, TV-HORS) ao RSA1024, todos considerando um nível de segurança de 80 bits[2]. Essa segurança é insuficiente para os dias atuais [Barker 2016] e foi testada em um *hardware* de maior capacidade que os IEDs.

Alguns estudos mostram uma nova abordagem para o problema. Como é economicamente inviável trocar todos os IEDs existentes nas subestações de energia, pois são equipamentos fabricados para muitos anos de longevidade, outra solução é utilização de

[1] RSA com chave de 1024 bits

[2] Comparação a um algoritmo simétrico com chave de 80 bits para ser quebrado através de força bruta

hardware de apoio. Uma pesquisa realizada em redes veiculares [Singla et al. 2015] desenvolveu um *hardware* especializado para criptografia utilizando GPUs (*Graphics Processing Unit*) com o algoritmo *Rapid Authentication* (RA) baseado no RSA Condensado [Yavuz 2014]. Utiliza-se uma placa NVidia Tesla K40c de 2880 cores de computação e 12 GB de memória RAM. O experimento foi executado em um computador pessoal com processador Intel Core i7 de 3,5 GHz e 16 GB de memória RAM. Também testa um SoC Tegra K1 com processador quadcore ARM Cortex A15 com 2,3 GHz e 192 núcleos de computação. O RSA2048[3] apresentou uma comunicação fim a fim de 4 ms. A GPU com o algoritmo RA processou em 0,21 ms e o SoC com o mesmo algoritmo teve desempenho de 2,6 ms. Na avaliação usou-se uma rajada de 8192 mensagens.

Outro trabalho [Miranda 2016] também usa um *hardware* adicional. O autor usa um NetFPGA para desenvolver um *firewall* de mensagens GOOSE aleatórias enviadas por um gerador de pacotes. O FPGA Virtex II Pro é utilizado com módulo GFIXED (*Message GOOSE Fixed*) e GPDU (*Protocol Data Unit GOOSE*) em *hardware*. Um módulo de criptografia (*Crypto*) também é utilizado. O *firewall* é testado com 10 mil mensagens do gerador de pacotes e 15 mil mensagens GOOSE de IEDs comerciais. Os algoritmos utilizados são o DES (*Data Encryption Standard*) para cifragem e o SHA3-512 para código de autenticação. O *firewall* implementado consegue filtrar com sucesso 100% das mensagens incorretas. Com o filtro DES foi registrado tempo de 77,39 μs e com o SHA-3 de 40,03 μs, considerando validação do quadro pelo firewall. O equipamento utilizado, no entanto, custa mais de $1.300,00 dólares [Digilent 2018].

A maioria dos trabalhos citados nesta seção utilizou computadores pessoais ou *hardware* específico para avaliar as propostas. A metodologia adotada neste artigo utilizará hardwares de Raspberry Pi 3. Além disso, será avaliado protocolo de cifra simétrica para substituir o RSA na assinatura digital e também na cifragem de toda a carga útil de dados da mensagem GOOSE, um cenário que não foi avaliado por outros autores.

3. Segurança na Comunicação em Redes IEC 61850

A implementação de algoritmos criptográficos em mensagens GOOSE é fundamental para evitar ataques cibernéticos em subestações de energia. Isso ocorre em virtude da implementação direta do protocolo GOOSE sobre a camada de enlace para atender aos requisitos estritos de tempo da norma IEC 61850. Um possível ataque ao protocolo GOOSE sem criptografia consiste na injeção de mensagens falsas com valores muito maiores que os vigentes no campo stNum [Kush et al. 2014], fazendo com que IEDs descartem mensagens legítimas com valores de stNum inferiores. As mensagens GOOSE legítimas com stNum menores do que o gerado pelo atacante são descartadas pelos nós receptores.

A *International Electrotechnical Commission* (IEC) estabeleceu a norma IEC 62351 [IEC62351 2007] com a definição de requisitos de segurança para vários protocolos, incluindo o GOOSE. Em particular, este artigo estuda as mensagens GOOSE de tempo real. Para essas mensagens, a norma IEC 62351 sugere o uso do RSA para assinatura digital das mensagens, garantindo autenticidade e integridade.

O protocolo GOOSE adota o modelo *publish/subscriber*, um nó publicador envia a mensagem e vários outros nós assinantes a recebem. Por isso, as mensagens GOOSE

[3]RSA com chave de 2048 bits.

são difundidas por *multicast*. A IEC 62351 estabelece que a mensagem GOOSE deve ser resumida com o algoritmo SHA256 e posteriormente esse resultado será assinado com o RSA. Isso garantirá a integridade da mensagem e a autenticidade. Entretanto, um estudo [Hohlbaum et al. 2010] mostra que o uso do RSA com chave de 1024 bits e com as restrições de tempo real exigidas pela norma eram possíveis em 2010 apenas com *criptochips* dedicados para o RSA. Ademais, o *National Institute of Standards and Technology* (NIST, Estados Unidos), entidade que estabelece padrões de segurança da informação, não recomenda mais o uso do RSA1024 [Barker 2016], sugerindo seu substituto o RSA2048 como requisito mínimo de segurança para futuras aplicações.

Permanece, então, a necessidade de alternativas ao RSA para proteger a comunicação entre IEDs em subestações através de mensagens GOOSE. A IEC 62351 indica que será adicionado um novo cabeçalho de segurança na mensagem, logo após o quadro GOOSE. No entanto, esta norma não especifica os detalhes de como isso será feito. Neste trabalho, os *bytes* extras resultantes do cabeçalho de segurança foram adicionados, assim como indica a norma, após o campo GOOSE PDU. Posteriormente, esses bytes são devidamente checados no destinatário da mensagem para garantir a entrega segura.

Os ataques às redes de subestação são devastadores [Hoyos et al. 2012]. O foco deste artigo é o ataque de falsificação no qual o atacante pode forjar uma mensagem GOOSE que pode causar grandes danos ao sistema elétrico em larga escala. Para evitar este ataque, dois requisitos de segurança são importantes de serem adicionados à comunicação entre IEDs: integridade e autenticidade. Desta forma, pacotes só são aceitos no destinatário se forem verificadas a integridade e a origem do pacote.

3.1. Algoritmos de Criptografia

A Tabela 1 enumera e resume as características dos algoritmos criptográficos usados na avaliação experimental desse trabalho.

O RSA [Rivest et al. 1978] é um algoritmo de criptografia assimétrica, isto é, seu funcionamento consiste na utilização de diferentes chaves para cifragem e decifragem de mensagens. Sua complexidade é baseada no problema de fatoração da multiplicação de números primos grandes, o que torna improvável a tentativa de descoberta das chaves utilizadas. Com o RSA é possível prover autenticidade, ou seja, ter a garantia de que uma mensagem foi enviada por determinado emissor. Para tanto, um emissor pode enviar junto da mensagem a sua assinatura. A assinatura digital é o processo em que o emissor da mensagem gera um resumo desta mensagem e cifra esse resumo com sua chave privada RSA. Essa assinatura digital é exclusiva desse emissor e da respectiva mensagem, e é enviada juntamente com a mensagem. O destinatário, ao receber a mensagem e a assinatura, faz a conferência dessa assinatura digital com a chave pública RSA do emissor. Se a conferência for positiva, consegue-se ter a garantia do emissor e da integridade da mensagem.

O algoritmo SHA256 [Mendel et al. 2006] é uma função *hash* que consiste em um conjunto de cálculos efetuados por um algoritmo para geração de uma saída de tamanho único dada uma entrada de tamanho qualquer. O resultado de uma função *hash* derivado do conteúdo de uma mensagem pode ser anexado a mensagem pelo emissor e enviada ao seu destino. Ao receber, o destinatário aplica a função *hash* ao conteúdo recebido e compara o resultado com o *hash* contido na mensagem. Se forem iguais, o conteúdo

está íntegro, caso contrário, houve algum tipo de alteração durante a comunicação. A aplicação deste algoritmo seguido do RSA proporciona integridade e autenticidade da comunicação e é denominada assinatura digital.

Tabela 1. Algoritmos de segurança e seus parâmetros

Algoritmo	Tamanho da Chave (bits)	Tamanho do Bloco (bits)	Tamanho da Saída (bytes)	Nível de Segurança [Barker 2016]
Nenhum	0	0	Igual entrada	Nenhuma
SHA256	0	512	32	128 bits
AES128	128	128	Igual entrada	128 bits
RSA2048	2048	Igual entrada	Igual entrada	112 bits
HMAC SHA256	128	512	32	256 bits
CMAC AES128	128	128	16	128 bits

Uma alternativa avaliada neste trabalho é a utilização da função SHA256 combinado com o algoritmo de chave simétrica denominado AES128 substituindo o RSA sugerido na norma IEC 62351. O AES utiliza a mesma chave para cifragem e decifragem de conteúdos e implementa cifragem de bloco. A cifragem de bloco opera recebendo como entrada um conjunto de bytes do conteúdo a ser encriptado a cada iteração. Através da Tabela 1 é possível observar que o AES128[4] provê maior nível de segurança [Barker 2016] quando comparado ao RSA2048 e, por esse motivo, é uma solução candidata a ser implementada para prover segurança a quadros GOOSE substituindo o RSA na assinatura digital das mensagens.

Outro mecanismo usado para prover autenticidade e integridade é o *Message Authentication Code* (MAC). O HMAC (*Hash MAC*) é um algoritmo que implementa esta estrutura utilizando uma função *hash* junta a uma chave criptográfica. Ele é composto de duas etapas. Durante a sua execução são gerados dois componentes: o *keypad* interno (i_keypad) e o *keypad* externo (o_keypad). A primeira etapa é a geração de um *hash* do resultado da operação XOR da mensagem com o i_keypad. Na segunda etapa, é feito um *hash* da aplicação do XOR entre o resultado da etapa anterior com o o_keypad.

Neste trabalho, o HMAC é implementado com a função *hash* SHA256 devido a sua utilização na assinatura digital do RSA mencionado anteriormente. Cada mensagem gera uma saída diferente no HMAC que é concatenada ao final da mensagem em texto plano. Já no destino, o receptor consegue repetir a aplicação do HMAC à mensagem em texto plano e comparar com o conteúdo cifrado que foi anexado a mensagem pelo emissor.

O último algoritmo implementado é o CMAC (*Cipher MAC*) AES128. Assim como o anterior, também é possível prover autenticidade e integridade, porém o algoritmo usado é de cifragem de bloco ao invés da função *hash*. A implementação é feita de acordo com a RFC4493 que passou por rotinas de testes da própria RFC e do NIST. O algoritmo recebe blocos da mensagem a ser cifrada e, à medida que os *n* blocos são cifrados pelo algoritmo AES, o algoritmo recebe como entrada também o resultado da cifragem do bloco *n-1*. Portanto, a saída deste algoritmo tem um tamanho fixo.

[4]AES com chave de 128 bits.

Além dos algoritmos MAC acima implementados, também é realizado o teste de cifragem total da mensagem utilizando o RSA2048 e o AES128. Na Seção 5 isso é abordado com maiores detalhes.

4. Metodologia de Avaliação

Para avaliar os algoritmos criptográficos, montou-se uma bancada de testes com objetivo de se aproximar o máximo possível de uma rede de subestação composta por alguns IEDs. O *hardware* escolhido foi o Raspberry Pi 3 Model B. Este equipamento possui desempenho semelhante a dos IEDs: arquitetura ARM, baixo *clock* de CPU, pequena quantidade de memória RAM, armazenamento limitado, interface de rede 10/100 Mb/s e funcionamento sem resfriamento ativo [ABB 2009]. O modelo escolhido possui um processador de 1.2 GHz Broadcom BCM2837 64 bits e 1 GB de memória RAM, de baixo custo, em torno de 30 dólares americanos. Os trabalhos anteriores utilizaram computadores pessoais de arquitetura x86 e processadores mais poderosos.

Cada dispositivo é instalado com o sistema operacional Ubuntu Mate 16.04.02, compilador GCC 4.7.4, biblioteca Libgcrypt20 e suas dependências. Todos os algoritmos de criptografia usados neste trabalho, são implementações próprias validados com vetores de teste do NIST. A única exceção é o algoritmo RSA, para o qual é utilizada uma implementação da biblioteca Libgcrypt [Project 2018], apenas para comparação com os algoritmos propostos visto que outros trabalhos já identificaram a inviabilidade do RSA.A implementação própria dos algoritmos apresenta melhor desempenho comparado aos de bibliotecas como a Libgcrypt, pois possuem código mais enxuto e objetivo, com menos etapas de processamento.

Quatro dispositivos são conectados para representar nós de uma subestação se comunicando através do protocolo GOOSE por uma rede Ethernet. É utilizado um comutador Ethernet Encore ENH908-NWY de 8 portas conectando todos os nós através de cabos de rede Cat6. Além disso, é conectado à rede um computador pessoal apenas para captura de pacotes, não influenciando o tráfego durante os testes.

A Figura 1 é uma captura de tela do *software* farejador Wireshark. Ela mostra um quadro GOOSE como definido na norma IEC 61850 e capturado durante os experimentos. As informações do quadro Ethernet são padronizadas em todas as mensagens, bem como todas as informações da carga útil IEC 61850, com uma única exceção. Dentro da carga útil, existe um campo chamado GOOSE.t que marca uma estampilha de tempo. Em nosso experimento, em todos os pacotes enviados, o único campo que sempre é alterado é o GOOSE.t que sempre recebe uma nova marcação de tempo para cada pacote gerado em tempo de execução. Portanto, o *hash* de cada pacote é substancialmente diferente um do outro.

Os experimentos consistem em medições de tempo das execuções dos algoritmos com diferentes soluções de segurança implementadas. Os programas implementados na linguagem C possuem marcadores que registram o tempo de processamento total de cada execução e divide esse tempo pela quantidade de quadros gerados, exibindo a média de tempo de cada teste ao término de cada execução. Como todos os quadros são diferentes, por exemplo, em uma rajada de 1000 pacotes, nenhum é replicado, tornando o processo de cifragem diferente em cada pacote gerado.

Para a avaliação de algoritmos criptográficos, são considerados cinco opções:

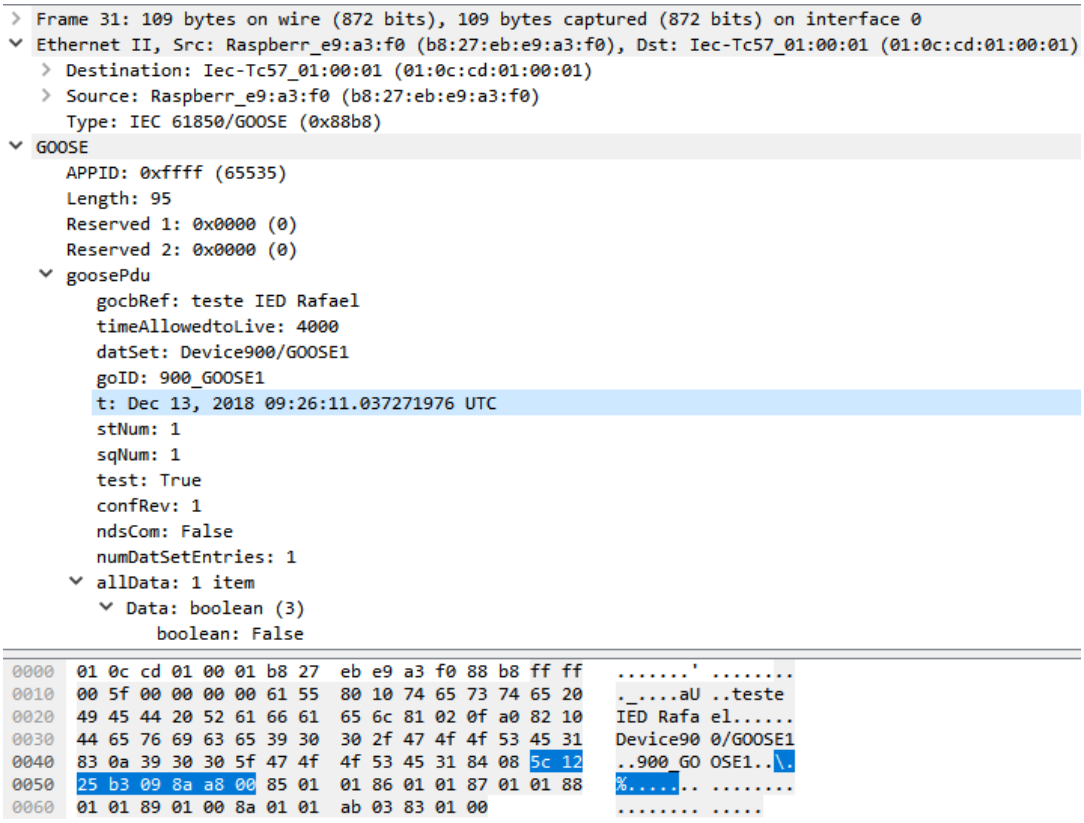


Figura 1. Estrutura do pacote GOOSE padrão utilizado nos testes. Apenas o campo GOOSE.t é alterado em cada pacote novo, em destaque

SHA256, AES128, RSA2048, HMAC SHA256 e CMAC AES 128. Todos os algoritmos são selecionados considerando as recomendações mínimas de segurança do NIST [Barker 2016] para implementações atuais e futuras. A Tabela 1 mostra em detalhes os algoritmos e suas características consideradas neste experimento. Para ter uma linha de base, todos os testes também são realizados sem nenhuma configuração de segurança, para que seja feita uma comparação direta da opção sem segurança para as outras alternativas avaliadas.

A avaliação dos tempos de geração dos quadros GOOSE é feita testando rajadas de 20, 100, 500, 1000 e 5000 mensagens. Com essas quantidades é possível observar o comportamento das médias calculadas para cada algoritmo de criptografia à medida que o número de mensagens da rajada aumenta. Em cada tipo de teste, são feitas 30 amostras de execução do algoritmo. Os gráficos da Seção 5 incluem os intervalos de confiança para um nível de confidencialidade de 95% que, em alguns casos, são valores muito pequenos e não ficam visíveis na escala utilizada.

Considerando todas as opções de algoritmos de segurança, mais as variações de tamanhos de rajadas e a quantidade de amostras por teste, ainda foram realizados 4 tipos diferentes de teste. O primeiro é a marcação de tempo no envio da mensagem, somente no transmissor. O segundo é a marcação de tempo na recepção da mensagem, nos receptores. Nesse caso foram utilizados três nós receptores e calculada uma média entre os tempos de cada um. O terceiro teste faz uma variação do tamanho do quadro GOOSE, adicionando preenchimento no mesmo, para analisar o comportamento da variação de tempo

nos algoritmos em relação ao tamanho do pacote. Foram adicionados 50, 100, 200, 350 bytes ao pacote padrão para verificar o comportamento dos algoritmos com uma carga útil maior. O quarto e último teste é realizado comparando dois algoritmos de criptografia cifrando todo a carga útil da mensagem, garantindo assim a confidencialidade, requisito de segurança além do que a norma exige, mas que é desejável.

A definição desta metodologia, em primeiro lugar, é a escolha do Raspberry por ser um *hardware* que se assemelha a um IED de produção em suas características. É estabelecido que os testes utilizem o algoritmo recomendado pela norma, para validar informações de que ele é inviável [Hohlbaum et al. 2010, Lu et al. 2012, Hoyos et al. 2012]. Posteriormente são avaliadas alternativas que proporcionem integridade às mensagens. É avaliado o AES em conjunto com o SHA256 para gerar um *hash* cifrado da mensagem e proporcionar integridade na comunicação. Por fim, as variantes MAC são testadas usando o *hash* SHA256 e a cifra simétrica AES128.

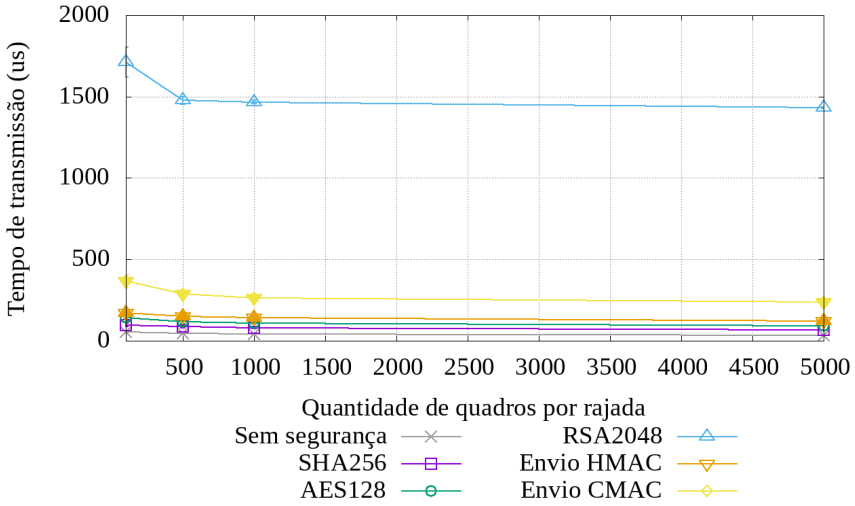
5. Avaliação dos Resultados

Como característica das redes IEC 61850, o modelo de comunicação *publisher / subscriber* apresenta um nó publicador que envia as informações e um ou vários nós assinantes, que são receptores da mesma mensagem enviada via *multicast*. Foram realizadas medições separadas da transmissão e recepção devido à diferença de relógio dos diferentes Raspberry. O *Network Time Protocol* foi testado, porém descartado pois apresentou uma precisão de tempo que afetava os resultados em vários microssegundos.

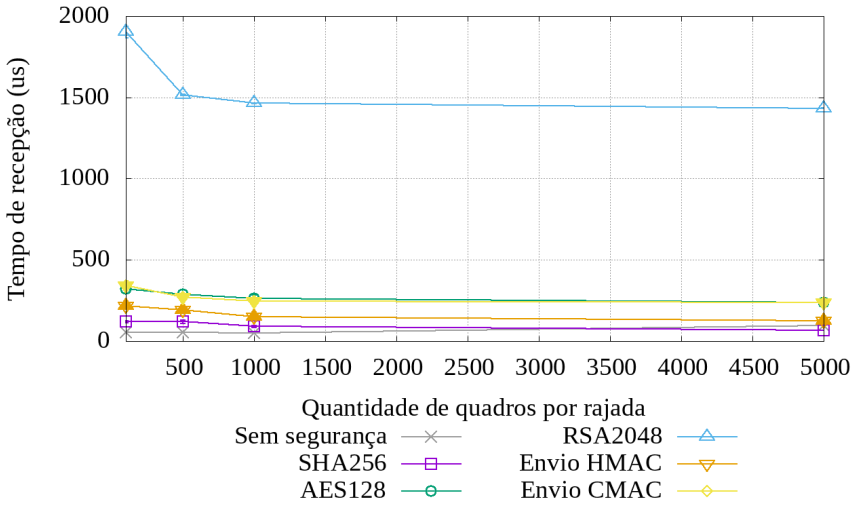
O tempo médio de envio de pacotes GOOSE em função do tamanho da rajada é apresentado na Figura 2a. Observa-se que quanto maior o tamanho da rajada, menor é o tempo de transmissão por mensagem em todos os algoritmos criptográficos avaliados. Esse comportamento ocorre porque as 10 primeiras mensagens da rajada sempre levam mais tempo de envio devido à abertura do *socket* no sistema operacional. Sobre os algoritmos criptográficos é possível observar que todos processam e enviam pacotes em menos de 500 μs, com exceção do RSA que opera próximo dos 1500 μs. O pacote com nenhuma segurança, ou seja, mensagem GOOSE padrão, tem uma média de tempo de 33 μs. Como o SHA256 marca a média de 66 μs, presume-se o tempo de calcular o *hash* é de 33 μs. Assim, todos os algoritmos, menos o RSA, operam abaixo dos 250 μs para envio incluindo o tempo de *hash*.

O tempo médio de recepção das mensagens GOOSE em função do tamanho da rajada é apresentado na Figura 2b. Observa-se que o tempo médio de recepção diminui com o aumento do tamanho da rajada, assim como aconteceu no tempo de transmissão. Observa-se também que cada um dos algoritmos criptográficos avaliados possuem tempos similares para processamento e transmissão das mensagens e recepção e processamento das mensagens. A exceção é o AES128 que apresentou tempo maior comparado ao envio, uma vez que a compilação para arquitetura ARM gera mais ciclos de relógio no processo de decifragem do AES [Bertoni et al. 2002]. Como observado por outros autores, o RSA se torna inviável para dispositivos de baixo processamento, pois somente a soma dos tempos de envio e recepção atinge o limite de 3ms definido pela norma IEC 61850, sem considerar o tempo de propagação que não foi medido nesses experimentos.

Os tempos médios de transmissão e recepção das mensagens GOOSE para uma rajada de 5000 pacotes podem ser observados na Tabela 2. Na metodologia de teste apre-



(a) Transmissão GOOSE



(b) Recepção GOOSE

Figura 2. Tempos de transmissão/recepção das mensagens GOOSE em função do tamanho da rajada para cada algoritmo criptográfico.

sentada nesta tabela, um nó transmissor envia mensagens para outros três nós receptores. Os valores na tabela são a média dos tempos de recepção das mensagens recebidas pelos 3 nós. Os resultados apresentados de tempo de comunicação são a soma dos resultados de envio e recepção, desconsiderando o tempo de propagação na rede.

Todos os algoritmos de segurança se mostram viáveis de implementação em aplicação real nas mesmas condições do cenário proposto com uma observação. O RSA está no limite do tempo de 3 ms exigido pela norma. Como os valores registrados mostram a média, é possível concluir que o RSA se torna inviável, pois apresenta o risco de extrapolar os 3 ms. Outras variáveis podem aumentar o tempo registrado como concorrência do processador do nó, condições de rede, aumento de tamanho da mensagem GOOSE, etc.

A Figura 3 apresenta o tempo médio de transmissão de mensagens GOOSE em função da quantidade de bytes acrescentados a cada mensagem originalmente com ta-

Tabela 2. Valores da média das amostras em μs.

	Nenhuma Segurança	SHA256	AES128	RSA2048	HMAC SHA256	CMAC AES128
Transmissão	33	66	93	1467	120	236
Desvio Padrão	2,56	3,37	2,59	10,74	2,58	3,66
Recepção	96	67	237	1434	124	239
Desvio Padrão	1,27	0,93	3,97	1,40	0,40	0,53
Soma	129	133	330	2901	244	475
Pior Tempo	157	140	339	2932	262	505

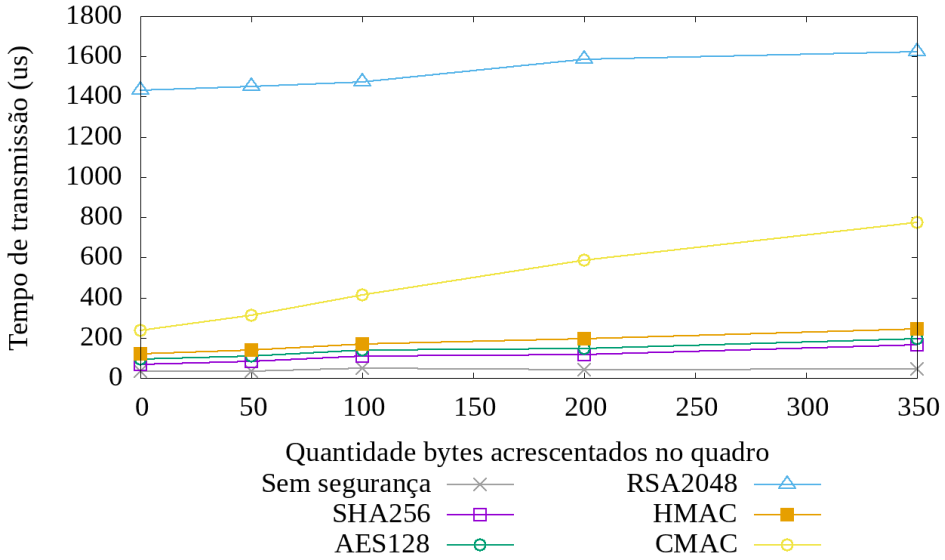


Figura 3. Tempo médio de transmissão das mensagens GOOSE em função da quantidade de bytes acrescentados a um quadro de 109 bytes.

manho de 109 bytes e rajada de 5000 mensagens. O algoritmo RSA apresenta tempo de processamento de envio superior a 1400 μs, comprovando resultados obtidos em outros trabalhos [Hohlbaum et al. 2010][Yavuz 2014]. Os tempos obtidos para o CMAC mostram que esse algoritmo tem resultados mais influenciados pelo aumento da quantidade de bytes do quadro a ser cifrado devido à característica de cifragem de blocos em cadeia mais complexa. Mesmo assim, os tempos para o quadro de maior tamanho do experimento (459 bytes) não ultrapassam 800 μs. Os demais algoritmos de criptografia apresentam crescimento linear, sendo menos influenciado pelo crescimento do tamanho do quadro a ser enviado do que o CMAC.

Os algoritmos SHA256, AES128 e HMAC e a configuração sem segurança, mostram um crescimento menor em função do aumento no tamanho da mensagem. Observa-se que em um comportamento normal, no qual a mensagem GOOSE vai variar entre 150 a 250 bytes, essas quatro variantes operam com tempo de transmissão abaixo dos 200 μs.

Como já mencionado, os algoritmos AES128 e RSA2048 receberam como entrada um resultado da função *hash* SHA256 nos experimentos. Para uma avaliação mais

detalhada de como se pode usar os algoritmos criptográficos, é realizado um último teste de desempenho. O objetivo é analisar se é possível cifrar toda a carga útil da mensagem GOOSE para prover confidencialidade. A Figura 4 apresenta um gráfico deste teste.

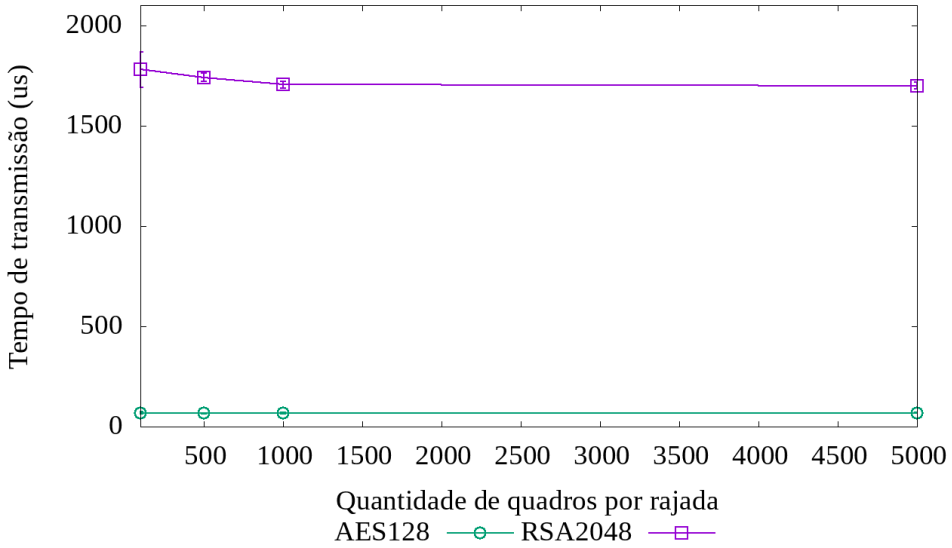


Figura 4. Tempo de cifragem e envio da mensagem GOOSE com carga útil totalmente cifrada em função da quantidade de pacotes enviados.

Observa-se uma variação mínima do tempo de transmissão em função da quantidade de pacotes enviados. O RSA apresentou desempenho em torno de 1700 μs para processar o envio com todo a carga útil cifrada. Comparando com a Figura 2a, é possível notar que houve um aumento do tempo de transmissão 1467 para 1700. Como o tamanho do quadro era o mesmo, o que mudou foi a quantidade de bytes cifrados pelo RSA. Por sua vez o AES128 enviou o quadro em tempo próximo de 69 μs, independente da quantidade de pacotes enviados na rede. Esse desempenho é melhor do que o observado na Tabela 2 para o mesmo algoritmo. Isso demonstra que o AES cifrando uma carga útil de 95 bytes (carga útil do quadro de 109 bytes padrão menos os 14 bytes do quadro Ethernet) é executado em menos tempo do que calcular o *hash* com SHA256 do quadro inteiro de 109 bytes e posteriormente cifrar com o AES os 32 bytes resultantes, procedimento este feito no primeiro teste.

6. Conclusão e Trabalhos Futuros

Neste artigo foi realizada uma avaliação experimental de algoritmos criptográficos candidatos à implementação de segurança no protocolo GOOSE. Deseja-se que estes algoritmos sejam capazes de garantir requisitos como integridade e autenticidade sem que os requisitos de tempo de entrega de mensagem da norma IEC 61850 sejam extrapolados.

Os resultados mostram que algoritmo AES, leva somente 12% do tempo para enviar uma mensagem GOOSE segura em relação ao tempo para enviar a mesma mensagem com o RSA. A comunicação fim-a-fim com o AES fica em torno de 330 μs, muito abaixo dos 3ms exigidos pela norma, tornando possível sua implementação em *hardware* ainda mais restrito do que o Raspberry. O AES também mostra que é possível cifrar totalmente

a carga útil das mensagens GOOSE e manter tempo de envio em 69 μs para um quadro de 109 bytes, provendo a confidencialidade, requisito acima do exigido pela norma.

Finalmente, outros dois algoritmos se mostram promissores e viáveis. Os algoritmos de MAC proporcionam a autenticidade da mensagem. O HMAC baseado em SHA256 apresenta tempo de comunicação fim-a-fim de 244 μs e há pouca variação neste tempo ao se aumentar o tamanho das mensagens enviadas. O CMAC implementado com o algoritmo AES128 tem uma variação maior com o aumento do tamanho das mensagens, mas ainda apresentou um tempo médio de transmissão abaixo de 800 μs para um pacote de 459 bytes, se mostrando também uma alternativa de aplicação em redes IEC 61850 com GOOSE. O CMAC é mais indicado para cenários com maiores restrições de memória nos nós, pois utiliza carga extra de segurança 50% menor que os outros algoritmos.

Os experimentos descritos abordam testes com diferentes algoritmos em um mesmo *hardware* em configuração padrão, implicando que as CPUs utilizadas possuam a mesma frequência de relógio. Um trabalho futuro pode inserir a frequência do processador como uma nova variável permitindo estudo dos algoritmos também em função da frequência do relógio do processador. Isso possibilitará avaliações direcionadas a requisitos de *hardware* menores ainda, atendendo a IEDs mais antigos em uso nas subestações.

Um dos pontos abertos desta pesquisa é a forma de distribuição das chaves simétricas na rede. Há uma corrente de estudos de implementações de SDN (*Software Defined Networking*) nas redes IEC61850 e esta é uma das possíveis maneiras de realizar a distribuição segura de chaves simétricas na rede e fazer a gestão de troca de chaves ao longo do tempo. Além disso, a aplicação do MAC no texto em claro bem como opções do AES de cifragem autenticada serão estudadas com maior cautela em trabalhos futuros.

Foi utilizado *hardware* similar aos IEDs devido a impossibilidade de utilizar um equipamento real em virtude de patentes e *softwares* proprietários dos fabricantes. Outro objetivo futuro é buscar parcerias com fabricantes de *hardware* IED para implementar os algoritmos diretamente no *firmware* desses equipamentos e realizar novos testes.

Referências

ABB, G. (2009). Manual técnico dos ieds da série 670. `https://library.e.abb.com/public/9830608e2e48f75fc12576f10031debf/1MRK580172-XEN_A_en_670_series_self_supervision.pdf`. Último acesso março de 2019.

ABNT5460 (1992). Sistemas elétricos de potência. In *ABNT NBR 5460:1992*. Associação Brasileira de Normas Técnicas. https://www.abntcatalogo.com.br/norma.aspx?ID=4123.

Barker, E. (2016). NIST special publication 800-57 part 1, revision 4, recomendation for key management. In *NIST Special Publication 800-57 Part 1 Revision 4*, page 160. National Institute of Standards and Technology.

Bertoni, G., Breveglieri, L., Fragneto, P., Macchetti, M., and Marchesin, S. (2002). Efficient software implementation of aes on 32-bit platforms. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 159–171. Springer.

Digilent (2018). Website do NetFPGA Virtex-II Pro FPGA Development System. `https://store.digilentinc.com/`

netfpga-virtex-ii-pro-fpga-development-system/. Último acesso em dezembro de 2018.

Farhangi, H. (2010). The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28.

Hohlbaum, F., Braendle, M., and Alvarez, F. (2010). Cyber security practical considerations for implementing IEC 62351. In *PAC World Conference*.

Hoyos, J., Dehus, M., and Brown, T. X. (2012). Exploiting the GOOSE protocol: A practical attack on cyber-infrastructure. *2012 IEEE Globecom Workshops*, pages 1508–1513.

IEC61850 (2003). Communication networks and systems in substations. In *IEC 61850 Standard*. International Electrotechnical Commission.

IEC62351 (2007). Power systems management and associated information exchange - data and communications security. In *IEC 62351 Standard*. International Electrotechnical Commission.

Kush, N., Ahmed, E., Branagan, M., and Foo, E. (2014). Poisoned GOOSE: Exploiting the GOOSE Protocol. In *Proceedings of the Twelfth Australasian Information Security Conference - Volume 149*, AISC '14, pages 17–22.

Lopes, Y., Fernandes, N., Castro, T., and Muchaluat-Saade, V. (2016). Desafios de segurança e confiabilidade na comunicação para smart grids. *XVI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

Lu, X., Wang, W., and Ma, J. (2012). Authentication and integrity in the smart grid: An empirical study in substation automation systems. *International Journal of Distributed Sensor Networks*, 2012.

Mendel, F., Pramstaller, N., Rechberger, C., and Rijmen, V. (2006). Analysis of step-reduced SHA-256. In *International workshop on fast software encryption*, pages 126 – 143.

Miranda, J. C. (2016). *Segurança Cibernética com Hardware Reconfigurável em Subestações de Energia Elétrica Utilizando o Padrão IEC 61850*. PhD thesis, Universidade de São Paulo, Escola de Engenharia de São Carlos.

Project, T. G. (2018). Libgcrypt home page. https://www.gnupg.org/software/libgcrypt/index.html. Ultimo acesso em dezembro de 2018.

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

Singla, A., Mudgerikar, A., Papapanagiotou, I., and Yavuz, A. A. (2015). HAA: Hardware-accelerated authentication for internet of things in mission critical vehicular networks. In *Milcom*, pages 1298–1304.

Wang, W. and Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344 – 1371.

Yavuz, A. A. (2014). An efficient real-time broadcast authentication scheme for command and control messages. *IEEE Transactions on Information Forensics and Security*, 9(10):1733–1742.