Protegendo o sistema operacional e chaves criptográficas numa urna eletrônica do tipo T-DRE

José Monteiro, Saulo Lima, Robson Rodrigues, Paulo Alvarez, Marciano Meneses, Fernando Mendonça, Rodrigo Coimbra

[1]Tribunal Superior Eleitoral (TSE)
Brasília, DF

sevin@tse.jus.br

Abstract. *This document shows the new key configuration of the brazilian voting machines, that now is based on computational security in the ROM model (Random Oracle Model), and will allow the free opening of the source code to the society. The previous models, based on key custody in the source code or its execution-time derivation, have been replaced by the use of an embedded security module, which characterizes the Brazilian voting machines as the T-DRE type.*

Resumo. *Este documento mostra a nova configuração de chaves da Urna Eletrônica (UE), que agora está assentada em uma segurança computacional dentro do modelo ROM (Random Oracle Model), o que permitirá a abertura livre do código-fonte para a sociedade. Os modelos anteriores, baseados em guarda de chaves no código-fonte ou a sua derivação em tempo de execução, foram substituídos pelo uso de um módulo de segurança embarcado, que caracteriza as urnas brasileiras como do tipo T-DRE.*

1. Introdução

A urna eletrônica (UE) foi introduzida no Brasil nas Eleições 1996. Os primeiros modelos tiveram a sua arquitetura baseada num computador pessoal com processador x86, com capacidade de memória e processamento limitados. As primeiras licitações incluíam a produção do hardware e o desenvolvimento do software. Até 2006 as urnas utilizavam VirtuOS e Windows CE. A partir de 2005 a programação passou a ser feita pela equipe técnica do Tribunal Superior Eleitoral (TSE). E desde 2008 todas as urnas utilizam Linux.

Com a evolução do voto eletrônico, a urna incorporou assinatura digital e criptografia para a proteção dos dados de eleitores, candidatos e resultados, assim como do software. Uma das evoluções mais significativas se deu em 2009, quando toda a arquitetura de hardware foi revista para a inclusão de um módulo de segurança dedicado [Gallo et al. 2010]. Esse módulo é capaz de fazer a autenticação de toda a cadeia de software de inicialização (BIOS, bootloader e kernel do Linux), autenticação de periféricos, geração e guarda segura de chaves e serviços de criptografia, assinatura digital e geração de números aleatórios. Esse novo equipamento, como proposto por Gallo et al., é caracterizado como um T-DRE - *Trusted Direct Recording Electronic*.

Ainda em 2009, o sistema eletrônico de votação foi submetido à primeira edição do Teste Público de Segurança (TPS). Outras foram realizadas em 2012, 2016 e 2017. O TPS 2017 evidenciou um problema na sistemática de guarda de chaves utilizadas pelo software da urna. Após aquela edição do TPS, foi implementado um novo mecanismo para a guarda de chaves por software, baseado na derivação de chaves considerando uma informação presente no hardware [Sevin 2018]. Contudo, essa não é a solução ideal.

Este trabalho apresenta uma solução para o problema da guarda de chaves pelo software da urna, baseada em seu módulo de segurança em hardware. Esse novo mecanismo será submetido a tentativas de ataque no próximo TPS, a ser realizado antes das Eleições 2020.

2. Arquitetura de segurança das urnas T-DRE

A segurança da urna está baseada no hardware (incluindo seu firmware) e software, que juntos implementam a segurança baseada em assinaturas sobre a cadeia de inicialização, impedindo que software ilegítimo seja executado, mas mantendo a auditabilidade. O modelo de ameaças e segurança segue o estabelecido por Gallo et al., indo além das recomendações do VVSG [USA-EAC 2007], segundo os autores.

2.1. Arquitetura de hardware

Desde 2009, a segurança de hardware se baseia principalmente no uso do Módulo de Segurança Embarcado - MSE [Segele 2019]. O MSE é um módulo computacional ARM, com memória não volátil própria e um módulo gerador de números aleatórios (TRNG). O componente possui áreas protegidas para o armazenamento de chaves e certificados.

O acesso se faz por interface USB, que se conecta à CPU por uma trilha interna da placa-mãe. O perímetro da placa-mãe onde se localiza o MSE está resinado na parte superior e protegido por uma camada de *mesh* na parte inferior. Essas proteções o caracterizam como um dispositivo *tamperproof*.

A UE também incorpora elementos de segurança aos terminais periféricos: teclado do terminal do eleitor, terminal do mesário e impressora. Todos são autenticados pelo MSE durante a inicialização da urna com o uso de assinaturas RSA. Os periféricos também estabelecem conexões cifradas, cujas mensagens precisam ser decifradas pelo MSE para uso pelo software.

2.1.1. Firmware do MSE

O firmware do MSE possui funções criptográficas de geração de chaves, cifração, assinatura, MAC, hash e PRNG, além de um gerador aleatório (TRNG).

As funções de cifração são: AES, GUARANA[1] e uma implementação do algoritmo assimétrico ECIES (baseado em curvas elípticas - EC), similar aos padrões mostrados em [Martínez et al. 2010]. Usa-se EC com curvas

[1]Algoritmo de Estado desenvolvido pelo CEPESC/Abin.

P521 [NIST 2009, Certicom 2000], AES, HMAC com SHA-512, à exceção da função de derivação de chaves, que usa a CKDF.

Os algoritmos de assinatura são: RSA e ECDSA (P521). Os algoritmos de hash são: SHA-512, SHA-384 e SHA-256. HMAC é feito com SHA-512. Em função dos questionamentos recentes sobre a curva P521 [Bernstein and Lange 2014], a partir da UE2020 será utilizada a curva E521 [Segele 2019].

O firmware exporta as suas funções por meio de uma API PKCS11 para o software através da interface USB do MSE, utilizando o protocolo USB HID para a comunicação com o driver.

2.2. Cadeia de certificados

A cadeia de certificados usada pelo TSE foi baseada no trabalho de Gallo et al. O trabalho define características como a execução de código assinado, o que permitiu que: (i) somente o software oficial da votação seja executado; (ii) haja fiscalização e o rastreamento de incidentes; e (iii) auditoria de binários em todas as fases de uma eleição. Outra característica é o atendimento à recomendação do VVSG [USA-EAC 2007] para uso de um módulo de assinatura em hardware *tamperproof* nas urnas, além do uso de uma PKI própria da autoridade eleitoral.

Assim, a cadeia de certificados do TSE (Figura 1) possui certificados e chaves para uso nas fases de votação oficial, em simulados de eleição e no desenvolvimento. Todos os certificados ficam embarcados no MSE. O modelo implementado pelo TSE difere do trabalho Gallo et al., pois há também um certificado para uso em simulados, que são os testes do sistema eleitoral em conjunto com TREs. Essa diferenciação de certificados isola o escopo de trabalho e alcance do pessoal envolvido nas várias etapas do processo eleitoral.

2.3. Cadeia de confiança

O processo de *boot* das urnas foi definido por meio de um esquema chamado de Cadeia de Confiança (Figura 2). Por esse esquema, incorporam-se elementos de *boot* seguro e confiável, pois usa o componente de segurança criptográfica para fazer o avanço nas várias etapas do *boot*. Essa arquitetura possui similaridades com o Intel Boot Guard e o Secure Boot do UEFI. O processo de *boot* com o MSE foi definido com as seguintes etapas (Figura 2):

1. Processo de hardware – o MSE é energizado, faz autoverificação, carrega o BIOS por meio de um barramento LPC e verifica a sua assinatura. Se for válida, então energiza a CPU que inicia o BIOS;
2. BIOS – carrega o bootloader e solicita a verificação de sua assinatura ao MSE. Se for válida, então passa à execução normal do bootloader e o MSE inicializa um temporizador para validação do hardware. Se não houver a validação do hardware em 4 minutos, os terminais da UE são bloqueados pelo MSE;
3. Bootloader – carrega o kernel cifrado, decifra-o e solicita ao MSE a verificação de sua assinatura. Se válida, então inicia o kernel;

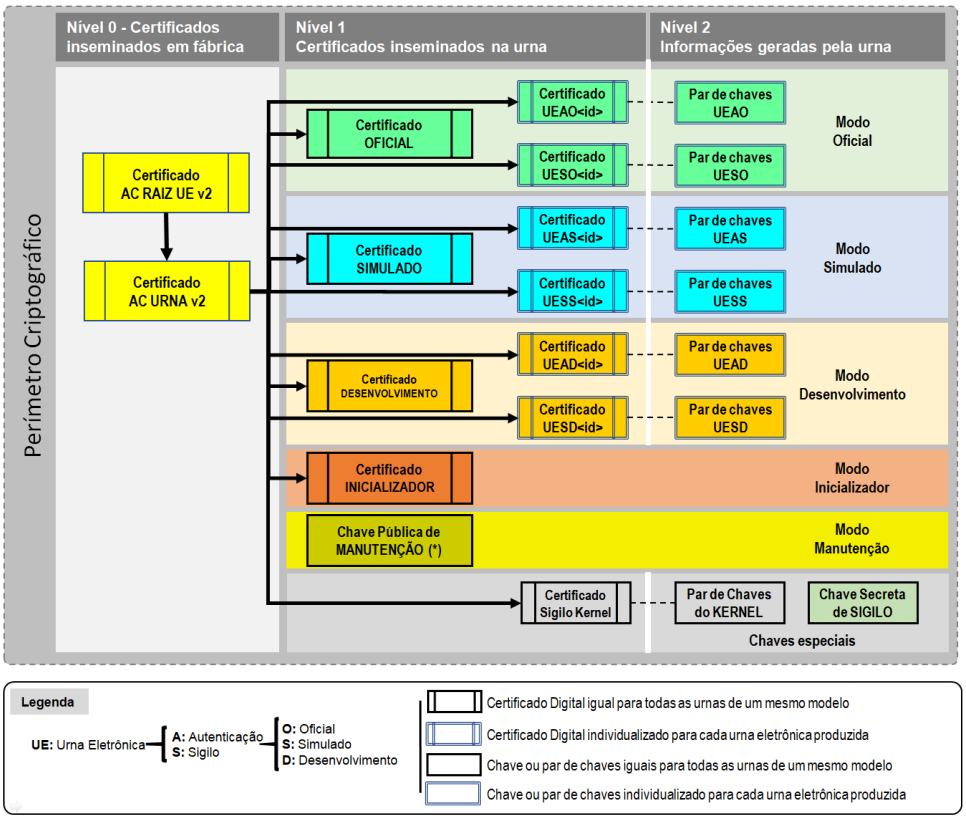

Figura 1. Cadeia de certificados e chaves da UE (Segele 2019).

4. Sistema operacional – valida o hardware de segurança por meio de um desafio. Se o desafio não for respondido corretamente os periféricos são travados. Caso contrário, carrega os aplicativos e verifica suas assinaturas. Se forem válidas, então inicia a execução dos aplicativos;
5. Aplicativos – ao serem carregados, têm suas assinaturas verificadas, assim como as bibliotecas dinâmicas das quais dependem. Por sua vez, esses aplicativos verificam a assinatura dos arquivos de dados, cifram e decifram outros arquivos de dados.

Note-se que há um elemento único, com a verificação do hardware pelo próprio sistema operacional por meio de desafios. Se alguma autenticação falhar, então a UE tem os terminais travados e para em 4 minutos a contados a partir da verificação do BIOS.

2.4. Assinaturas

A UE utiliza diversos algoritmos de assinatura, com tamanhos de chave e usos diversos. As urnas produzidas antes de 2009 não possuem o MSE e realizavam a verificação de assinaturas por software, via bibliotecas fornecidas pelo CEPESC/Abin. Tais bibliotecas usam curvas geradas aleatoriamente e assinam com algoritmo ECDSA. As bibliotecas também permitem a cifração usando o algoritmo ElGamal. Essas assinaturas são utilizadas para que um aplicativo verifique um outro aplicativo antes de sua execução, ou verifique

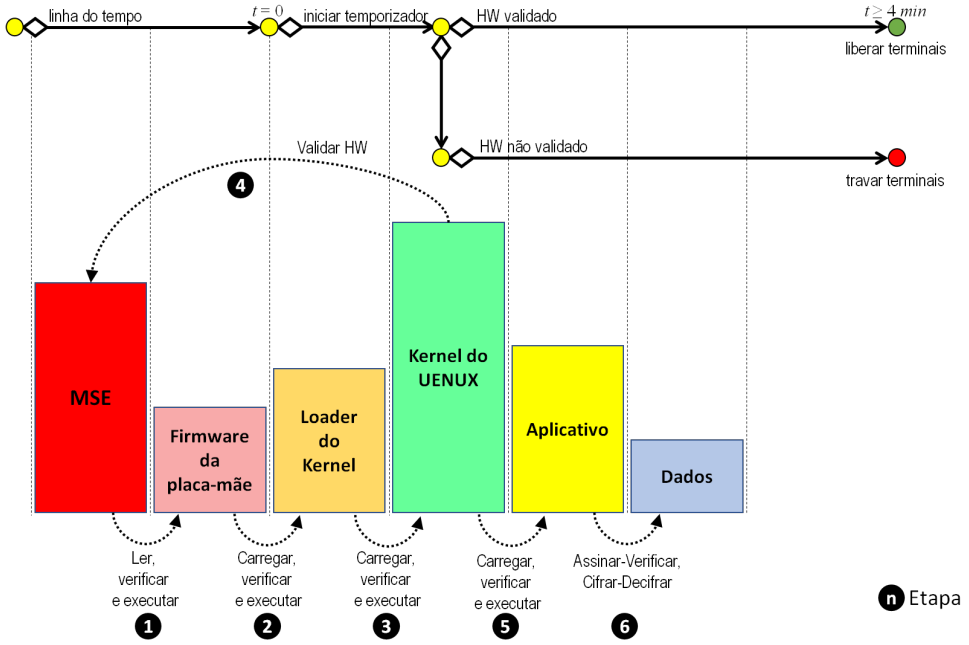

Figura 2. Cadeia de confiança da inicialização da UE (Segele 2019).

um dado antes de sua utilização. Em caso de falha na verificação de um aplicativo, ele não é executado e no caso de falha na verificação de um dado, ele não é utilizado. Em ambos os casos a execução do aplicativo na urna é interrompida e é feito registro em log.

Além de não estar disponíveil em todas as urnas, o problema da assinatura pelo MSE é o seu desempenho. Uma assinatura leva até 2 segundos para ser feita pelo MSE. Contribuem para esse tempo a comunicação USB, a capacidade de processamento do processador ARM e o tamanho da curva. Com isso, outras assinaturas passaram a ser consideradas. Por exemplo, para os QR Codes dos resultados, optou-se por uma assinatura EdDSA por possuir algoritmo seguro, rápido e amplamente disponível [Coimbra et al. 2017].

A adoção de Linux na urna, por sua vez, trouxe a possibilidade de se assinar as bibliotecas de carregamento dinâmico no momento em que são carregadas pelo kernel. Trata-se de uma assinatura RSA de 4096 bits. É gerado um certificado autoassinado que é usado para verificar as assinaturas de executáveis, bibliotecas e módulos do kernel. A chave privada gerada é destruída ao final do processo de compilação de todos os binários.

3. Segurança do kernel

Mecanismos de assinatura digital e criptografia começaram a ser introduzidos no software da urna a partir de 1998. Nesse mesmo ano, o hardware da urna passou a contar com a CriptoTable — um vetor de 1024 bytes aleatórios que é gravado na extensão de BIOS de todas as urnas, ou seja, todas as urnas possuem o mesmo vetor.

Até 2006, o software da urna era construído sobre uma base de software

proprietária, com os sistemas operacionais VirtuOS e Windows CE, os quais não permitiam a incorporação de mecanismos de segurança ao processo de inicialização do sistema operacional. Com a introdução do Linux em 2008, a equipe técnica do TSE foi capaz de projetar mecanismos que protegessem a inicialização do sistema operacional.

Em 2009, foram introduzidas as urnas com MSE. Além da verificação de assinaturas da cadeia de inicialização do software, o MSE também prevê uma chave para a cifração do kernel do Linux. Nos anos seguintes ainda seriam usadas urnas sem MSE[2], então a criptografia do kernel precisava ser feita por um mecanismo presente em todos as urnas: a CriptoTable.

3.1. Método original

Em 2009, foi introduzida a decifração do kernel pelo bootloader. O kernel era cifrado com AES ECB durante o seu processo de compilação. Essa proteção trouxe o problema de gestão da chave de decifração na urna. Em geral, os sistemas computacionais protegem chaves com o uso de algum segredo inserido pelo operador. Na UE isso é impraticável, pois, seria necessário que muitas pessoas tivessem conhecimento do mesmo segredo (servidores da Justiça Eleitoral e mesários), ou seja, na prática não existiria segredo.

Assim, foi adotada uma solução que não dependia de um operador, o que implicou se ter a chave de decifração no código-fonte do bootloader. Essa mesma estratégia foi utilizada para a guarda de outras chaves na urna: criptografia do sistema de arquivos (AES XTS), criptografia do RDV[3] e criptografia de outras chaves (ambos AES CBC). Nesses casos, as chaves estavam inseridas no código-fonte do kernel, cujo binário estava cifrado nos cartões de memória das urnas.

Em 2016 o kernel passou a ser cifrado com AES CTR, mas ainda com a chave embarcada no código-fonte do bootloader.

3.2. CriptoTable

Antes do TPS 2017, a criptografia do RDV passou a utilizar um mecanismo de derivação de chaves a partir da CriptoTable.

No TPS 2017, a guarda de chaves no código-fonte se mostrou frágil. Foi feito um ataque bem sucedido sobre a criptografia do sistema de arquivos, pois a chave foi encontrada no ambiente de inspeção do código-fonte [Sevin 2018].

A solução foi expandir o mecanismo já empregado na criptografia do RDV para o kernel e a proteção de outras chaves: usar a CriptoTable para derivar chaves [Sevin 2018]. No Teste de Confirmação[4], realizado em maio de 2018, a solução foi apresentada e os pesquisadores presentes não conseguiram repetir os feitos do TPS 2017 [Sevin 2018].

---

[2]As urnas modelos 2006 e 2008, que não possuem MSE foram usadas até as Eleições 2018.

[3]Registro Digital do Voto: arquivo no qual os votos são gravados em posições aleatórias ao longo do dia.

[4]`http://www.tse.jus.br/imprensa/noticias-tse/2018/Maio/tse-conclui-teste-publico-de-seguranca-do-sistema-eletronico-de-votacao`

Para cifrar o kernel, foi desenvolvido um aplicativo para a urna que gera aleatoriamente as "Tabelas de Janelas" para o bootloader e as chaves K_{AES} e o IC, em que K_{AES} é a chave secreta do AES CTR, IC é um valor inicial de contador para o AES CTR, derivadas a partir da CriptoTable e "Tabela de Janelas". Como o aplicativo é executado na UE, ele tem acesso a esses parâmetros secretos. Nesta implementação, mesmo o IC é secreto.

As chaves K_{AES} e o IC são cifradas para o programa de cifração de kernel no desktop do desenvolvedor. Esse programa usa o AES CTR para cifrar o kernel. Note-se que como o desktop não possui a CriptoTable, então precisa das chaves reais (K_{AES} e o IC) derivadas da CriptoTable.

Para decifrar o kernel, a CriptoTable T é usada da seguinte forma:

1. Gera-se aleatoriamente uma "Tabela de Janelas", J, que nada mais é que uma sequência de 64 índices i t.q. $0 \leq J(i) \leq 1023$. O termo "Janela" refere-se ao fato do índice permitir abrir, ou revelar, o valor da CriptoTable na posição da "Janela" $J(i)$;
2. A "Tabela de Janelas" é inserida no bootloader, o qual é assinado na sala-cofre do TSE;
3. Ao executar, o bootloader carrega os 64 bytes aleatórios da CriptoTable, $\{A_i, i = 0..63\}$, correspondentes às posições dadas pela "Tabela de Janelas": $A_i = T(J(i)), i = 0..63$;
4. Os bytes A_i são usados para preencher um HMAC baseado em SHA-512, e produzem $K_i = HMAC(A_{low}, A_{high}), i = 0..63$, em que $A_{low} = \{A_0..A_{31}\}$ e $A_{high} = \{A_{32}..A_{63}\}$;
5. Os valores K_i são usados como chave para um AES CTR que é usado para decifrar o kernel da seguinte forma: $K_{AES} = \{K_0..K_{31}\}$, e $IC = \{K_{32}..K_{48}\}$, em que K_{AES} e IC foram definidos acima.

O endereço da CriptoTable não está *hardcoded* no código-fonte, mas sim derivado durante a execução.

Usando um processo semelhante, as chaves do sistema de arquivos, do RDV e das demais chaves são derivadas da seguinte forma:

1. Gera-se aleatoriamente uma "Tabela de Janelas", J, com 32 índices i t.q. $0 \leq J(i) \leq 1023$;
2. A "Tabela de Janelas" é inserida no código-fonte do software da urna;
3. Ao ser executado, o software carrega os 32 bytes aleatórios da CriptoTable, $\{A_i, i = 0..31\}$, correspondentes às posições dadas pela "Tabela de Janelas": $A_i = T(J(i)), i = 0..31$;
4. Os bytes A_i são usados para preencher um HKDF, e produzem a chave $K = HKDF(A, Salt, Info)$, em que $Salt$ e $Info$ são valores não-secretos, no entanto são escolhidos aleatoriamente para cada aplicativo;
5. Os valores K_i são usados como chave secreta e IV para a criptografia do RDV, de outras chaves ou do sistema de arquivos.

4. A nova solução

Com o descarte das urnas modelos 2006 e 2008 e a aquisição de novas urnas prevista para 2020, todas as urnas da Justiça Eleitoral contarão com o

MSE. Isso permite que se faça o uso completo das funções de criptografia da UE. Existe uma chave de kernel no MSE para uso em cifração (Figura 1). É uma chave assimétrica baseada na P521, cuja característica principal é ser a mesma para todas as urnas. Isso significa que se algo for cifrado usando sua chave pública, então qualquer urna poderá decifrá-lo.

Dessa forma, pode-se cifrar um aleatório (chave de SO) com a chave pública de kernel, usar o MSE para fazer essa decifração e, então, usar esse aleatório para gerar as chaves do AES CTR. A geração do par de chaves de kernel está descrito na subseção 4.3.

Para cifrar o kernel, foi desenvolvido um aplicativo para a urna que gera aleatoriamente uma chave de SO, com 128 bytes, e a cifra com o ECIES-TSE, produzindo o cifrado para o bootloader. Também são derivadas as chaves K_{AES} e o IC a partir da chave de SO, que são armazenadas cifradas para o programa de cifração de kernel em desktop. Esse programa é executado no desktop do desenvolvedor e executa o AES CTR para cifrar o kernel. Ao final desse processo, como o bootloader foi alterado, é necessário assiná-lo na sala-cofre do TSE ou a cadeia de confiança terá a execução quebrada (subseção 2.3).

Dado que se possui a chave pública da chave de kernel gerada, que é comum a cada MSE (pub_K), a decifração do kernel se fará da seguinte forma:

1. Gera-se um aleatório de 128 bytes $A = \{A_i, i = 0..127\}$;
2. Cifra-se A usando o ECIES-TSE (seção 4.2): $Cif = ECIES - TSE(pub_K, A, 0)$;
3. O cifrado Cif é inserido no bootloader;
4. Ao executar, o bootloader decifra o cifrado inserido usando o MSE e obtém os 128 bytes aleatórios, $\{A_i, i = 0..127\}$;
5. Os bytes $A_i, i = 0..63$ são usados para preencher um HMAC baseado em SHA-512, e produzem $K_i = HMAC(A_{low}, A_{high}), i = 0..63$, em que $A_{low} = \{A_0..A_{31}\}$ e $A_{high} = \{A_{32}..A_{63}\}$;
6. Os valores K_i são usados como chave para um AES CTR que é usado para decifrar o kernel, da seguinte forma: $K_{AES} = \{K_0..K_{31}\}$, e $IC = \{K_{32}..K_{48}\}$.

Os 128 bytes do aleatório decifrado pelo bootloader também passarão por hashes (SHA-512) e serão enviados para o kernel, como parâmetro[5], para servir na geração de chaves para os aplicativos da UE.

As chaves de criptografia do RDV e as chaves de outros aplicativos serão derivadas usando um processo semelhante, i.e.:

1. Usando os 128 bytes enviados pelo bootloader ao kernel (B), i.e., $B_{Low} = HASH(A_{Low}), i = 0..63$ e $B_{High} = HASH(A_{High})), i = 64..127$, em que $A_{Low} = \{A_0..A_{63}\}$ e $A_{High} = \{A_{64}A_{127}\}$;
2. É definida uma posição inicial p_{app} para cada uma das aplicações;
3. Os bytes $B_i, i = p_{app}..(p_{app} + 31)$ são usados para preencher um HKDF e produzem $K = HKDF(B, Salt, Info)$, em que $Salt$ e $Info$ são valores que não são secretos, mas escolhidos aleatoriamente por aplicação;

[5] A segurança pode ser maior usando um esquema de cifração do hash da chave de SO. Hoje o kernel do Linux está adaptado para não revelar parâmetros passados durante o *boot*.

4. Os valores K_i são usados como chave secreta e IV para um AES CBC, conforme a aplicação.

A partir das Eleições 2020 não será mais utilizada criptografia no sistema de arquivos da urna, pois a proteção das chaves é suficientemente forte. Isso permitirá uma maior transparência na auditoria dos arquivos do sistema.

Para efeitos de publicação do código-fonte, na configuração de chave mostrada na subseção 3.1 haveria a revelação direta da chave usada para cifrar o kernel. No caso da subseção 3.2 haveria a revelação após um processo mais trabalhoso, envolvendo engenharia reversa do fonte em conjunto com uma simulação adequada, e se descobriria o endereço da CriptoTable. Uma vez conhecido esse endereço, se faria o exame do firmware do BIOS e se descobriria a CriptoTable, e, então, seria revelada a chave para decifrar o kernel.

No caso desta nova solução, não há o que revelar a não ser o cifrado que só o MSE pode decifrar. Qualquer um que obtiver esse cifrado terá que quebrar uma cifração ECIES (EC com 521 bits) para obter a chave.

4.1. Algoritmo CKDF

O algoritmo ***CKDF***, usado pelo ECIES-TSE, gera 64 bytes de chave a cada iteração. Assim, para gerar 128 bytes, o ***CKDF*** executa as seguintes operações:

Algoritmo 1 CKDF

1: **for** $counter = 1, 2$ ($counter$ com 4 bytes) **do**
2: calcule $K_{counter} = HASH(counter \mid R_X \mid AlgId \mid Q \mid P)$, em que Q e P tem 133 bytes cada e $HASH = SHA512$
3: **end for**
4: **return** $\{K_1, K_2\}$

4.2. ECIES-TSE

O ECIES-TSE é composto de 4 algoritmos: um algoritmo assimétrico (gera uma chave pública a partir de um escalar); um algoritmo KDF para a derivação de uma chave (CKDF); um algoritmo de cifração (AES CTR); e um algoritmo de MAC para autenticar o cifrado (HMAC com SHA-512).

Seja Q a *chave efêmera*. Q contendo 133 bytes ($Q = \{0x04, Q_X, Q_Y\}$), em que Q_X e Q_Y são as coordenadas (x, y) do ponto Q, cada uma com 66 bytes e precedido por 0x04, totalizando assim 133 bytes. Seja K um aleatório, tal que $K < n$, n o parâmetro da curva P521. Seja G o ponto gerador da curva P521. Seja R o *segredo compartilhado* inicial (ponto da curva), P uma chave pública. Seja R_X a coordenada x do segredo compartilhado (66 bytes), $AlgId$ um identificador de algoritmo (4 bytes). Seja K_{AES} a chave secreta para usar numa cifração com AES CTR (32 bytes), IC_{AES} é o contador inicial para o AES CTR (32 bytes), K_{hmac1} a chave secreta para usar no HMAC do cifrado (64 bytes), H_1, que possui 64 bytes e é gerado com o SHA-512.

Supondo que será cifrada uma mensagem m usando o ECIES-TSE, então é feito o seguinte:

Algoritmo 2 ECIES-TSE

1: $Input : P, m, K$
2: **while** $(K = 0)||(K \geq n)$ **do**
3: $K \leftarrow_{aleatorio} \{0, 255\}^{66}$
4: $K[0] \leftarrow K[0]$ **and** $0x1$
5: **end while**
6: $Q \leftarrow KG$
7: $R \leftarrow KP$
8: $S \leftarrow CKDF(R_X, AlgId, Q, P)$
9: $\{K_{AES}, IC_{AES}, K_{hmac_1}\} \leftarrow S$
10: $C_1 \leftarrow AESCTR(K_{AES}, IV_{AES}, m)$
11: $H_1 \leftarrow HMAC(K_{hmac_1}, C_1)$
12: **return** $\{Q, C_1, H_1\}$

Se m tiver 128 bytes, esse cifrado terá 133+128+64=325 bytes.

4.3. Geração da chave de kernel

A geração da chave de kernel é feita em duas etapas: i) cifração de uma semente aleatória num aplicativo desktop, usando a chave oficial de sigilo (cifração) de cada UE e, assim, é gerado um arquivo cifrado para cada urna; e ii) decifração dessa semente pelo MSE de cada UE, o qual gera a chave de kernel, que será a mesma para todas as urnas.

Assim, supondo que o desktop (A) vai gerar uma mensagem cifrada, cujo claro é a semente comum a todas as urnas, seq, e a UE (B) vai decifrá-la com o MSE, então são executados os seguintes procedimentos para implementar a primeira parte, citada acima.

Primeiramente, A gera um aleatório K a partir da semente seq:

i. A gera um aleatório seq (66 bytes), que é a semente da chave de kernel;
ii. A calcula $K = reverse(seq)$, onde $reverse$ é a sequência aleatória seq em ordem reversa, t.q., $K[i] = seq[66 - i], i = 1..65$;
iii. A faz $K[0] = K[0]\ AND\ 0x01$;
iv. Enquanto $K > n$, em que n é o parâmetro da curva P521 definido em [Certicom 2000], o processo é repetido.

Seja Pub a chave pública de sigilo da UE. A executa, então, a etapa ECIES para obter o 1º cifrado:

i. gera 128 bytes aleatórios KG;
ii. calcula $ECIES - TSE(Pub, KG, K)$ (algoritmo 2);
iii. Obtém $\{Q, C_1, H_1\}$.

A executa, então, a etapa GUARANA para obter o 2º cifrado. Essa etapa é composta por dois algoritmos: um algoritmo de cifração de blocos, o GUARANA; e um HMAC com SHA-512 para autenticar o cifrado.

i. A toma como chaves o conjunto $\{K_{GUA}, IV_{GUA}, K_{hmac_2}\} = KG$, em que K_{GUA} é a chave secreta para usar numa cifração com o GUARANA (32 bytes), IV_{GUA} é o IV para o GUARANA (32 bytes), K_{hmac2} é a chave secreta para usar no HMAC do cifrado do GUARANA (64 bytes);

ii. A calcula o cifrado $C_2 = GUARANA(K_{GUA}, IV_{GUA}, seq)$, em que o claro a ser cifrado é a sequência aleatória seq de 66 bytes, completada com bytes aleatórios. C_2 tem 80 bytes;
iii. A calcula o $H_2 = HMAC(K_{hmac_2}, C_2)$. H_2 tem 64 bytes, pois o HMAC usa o SHA-512;
iv. $A \rightarrow B : \{C_2, H_2\}$, o tamanho desse cifrado é de 80+64=144 bytes.

O cifrado completo C (contendo a sequência cifrada) é constituído de $C = \{Q, C_1, H_1, C_2, H_2\}$ e terá 469 bytes.

Para a segunda parte, o MSE fará a decifração de C, obtendo seq. Usará a sequência seq decifrada para alimentar um PRNG, obtendo um escalar $priv_K$, que será a chave privada de kernel e, multiplicada pelo ponto gerador da curva P521, produzirá pub_K, que é a chave pública de kernel. Como seq foi cifrada para todas as urnas, então todas terão a mesma chave de kernel.

4.4. Avaliação da segurança

A utilização de cifras e funções de hash, tais como descritas nas seções anteriores, implicam na irreversibilidade de segredos críticos em seu transporte e guarda. As várias etapas dos algoritmos dependem de aleatórios e, assim, são esquemas baseados no modelo ROM.

A geração da chave de kernel pelo MSE requer que a urna seja iniciada em "Modo Inicializador". Para tanto é necessária a assinatura do bootloader e do kernel com a chave "ìnicializador", que não está disponível para a equipe de desenvolvimento do software.

Uma vez gerada a chave de kernel, uma urna é usada para gerar e cifrar as demais chaves usadas pelo software. Seguindo o procedimento de geração de chaves da Cerimônia de Lacração do software [Sevin 2018], essas chaves são geradas por uma urna e só podem ser decifradas pelas urnas, a partir da chave de kernel. Nesse ponto, nem a equipe de desenvolvimento do software e nem atacantes externos têm acesso a essas chaves.

Assim como já ocorria com a solução usando a CriptoTable para a criptografia do kernel [Sevin 2018], a solução apresentada implica que somente a urna é capaz de decifrar o kernel. Isso impede a execução do kernel em ambiente diferente da urna, seja em máquinas reais ou virtuais.

5. Conclusões

Um ponto sensível nas urnas é o kernel do Linux utilizado, assim é preciso protegê-lo adequadamente. Da mesma forma, é necessário proteger os conjuntos de chaves de criptografia e assinatura utilizadas pelas aplicações.

A solução implementada é um casamento de assinaturas, que não permitem a execução de código não autorizado na UE, com cifração, que não permite que o código real seja executado fora da urna. Esse mecanismo se constitui em uma solução definitiva para o problema de cifração do kernel e de proteção das chaves utilizadas pelo software. Sem o MSE, a proteção era

baseada em chaves secretas derivadas em tempo de execução. Agora a segurança está na dificuldade de reverter a cifração que usa ECIES.

Com a futura abertura do código-fonte, essa implementação permite uma ampla avaliação do software da urna. No entanto, somente o software sob guarda do TSE poderá ser executado nas urnas. Isso porque o bootloader e kernel são assinados na sala-cofre do Tribunal. Da mesma forma, o kernel e as chaves são cifradas na sala-cofre do TSE. Tentativas de forjar software ou chaves fora do ambiente seguro do Tribunal não serão bem sucedidas, uma vez que o hardware não será capaz de iniciar esse software.

6. Agradecimentos

Agradecemos a Frank Simião, Ana Heloisa Bastos, Nicole Nascimento, Ana Cristina Souza, Cláudia Almeida, Marcus Amorim, Rúbio Terra, Ricardo Chaves, Lucas Guimarães, Marcelo Pinto e Gladiston Costa pelo empenho nas atividades de programação, teste e apoio à gestão, além da revisão deste artigo.

Referências

Bernstein, D. J. and Lange, T. (2014). Safecurves: choosing safe curves for elliptic-curve cryptography. https://safecurves.cr.yp.to/.

Certicom (2000). SEC 2: Recommended Elliptic Curve Domain Parameter. Technical report, Certicom Research.

Coimbra, R. C. M., Monteiro, J. R. M., and da Silva Costa, G. (2017). Registro impresso do voto, autenticado e com garantia de anonimato. In *Anais do XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 666–681.

Gallo, R., Kawakami, H., Dahab, R., Azevedo, R., Lima, S., and Araujo, G. (2010). T-DRE: A hardware trusted computing base for direct recording electronic vote machines. In *ACSAC '10: Proceedings of the 26th Annual Computer Security Applications Conference*, pages 191–198, New York, NY, USA. ACM.

Martínez, V. G., Encinas, L. H., and Ávila, C. S. (2010). A survey of the elliptic curve integrated encryption scheme. *Journal of Computer Science and Engineering*, 2(2):7–13.

NIST (2009). Digital Signature Standard (DSS). Technical Report FIPS PUB 186-3, NIST-National Institute of Standards and Technology.

Segele (2019). UE2020 - Edital de licitação - Anexo IV - Especificações Técnicas - Segurança. http://www.tse.jus.br/servicos-judiciais/audiencias-publicas/arquivos/ue2020.

Sevin (2018). Respostas as vulnerabilidades e sugestões de melhorias encontradas no Teste Público de Segurança 2017. http://www.justicaeleitoral.jus.br/arquivos/relatorio-tecnico-tps-2017-1527192798117.

USA-EAC (2007). Recommendations to the eac voluntary voting system, guidelines recommendations, 2007.