Um Método de Anonimização com Preservação de Prefixo baseado em Cifra de Feistel Desbalanceada e Heterogênea

Caroline Campos Carvalho [1], Ligia F. Borges [1], Anderson B. de Neira[2], Daniel M. Batista[3], Michele Nogueira [1,2]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

[2]Departamento de Informática - Universidade Federal do Paraná

[3]Departamento de Ciência da Computação - Universidade de São Paulo

{carvalho.campos,ligia.borges,michele}@dcc.ufmg.br

abneira@inf.ufpr.br, batista@ime.usp.br

Resumo. *A preocupação com a privacidade dos dados cresce diante do aumento de ameaças à violação dos mesmos. A anonimização de dados, técnica usada para alcançar privacidade, enfrenta o desafio de equilibrar a proteção e preservação da utilidade dos dados, como nos endereços IP. Para manter a utilidade desses endereços, a anonimização com preservação de prefixo é empregada, porém, ela é vulnerável a ataques semânticos capazes de reverter os valores originais. Soluções como multi-visão de dados ou cifras de Feistel têm sido investigadas a fim de fortalecer essa técnica, mas ainda enfrentam a necessidade de tratar da falta de robustez contra ataques lineares e diferenciais (i.e., deduzir dados originais a partir de mudanças na entrada). Assim, este artigo apresenta PROTECT, um método de anonimização com PReservação de prefixO para IPv4 baseada em uma cifra de feisTEl desbalanCeada e heTerogênea. A solução possui a propriedade de efeito avalanche e supera a literatura na métrica de vazamento de dados a partir 20% de conhecimento do atacante.*

Abstract. *Data privacy concerns are growing due to an increase in threats of data breaches. Data anonymization, a technique employed to achieve privacy, faces the challenge of balancing data protection and preserving data utility, as for IP addresses. In order to maintain the utility of these addresses, prefix-preserving anonymization is employed, but it is vulnerable to semantic attacks that can reverse the original values. Solutions such as multi-view data schemes or Feistel ciphers have been explored to strengthen this technique. Yet, they still need to address vulnerabilities against linear and differential attacks (i.e., deducing original data from changes in input). Hence, this paper introduces PROTECT, a method of anonymization with prefix preservation for IPv4 based on an unbalanced and heterogeneous Feistel cipher. The solution features avalanche effect properties and exceeds existing literature for the data leakage metric by 20% knowledge from the attacker.*

1. Introdução

A preservação da privacidade atrai cada vez mais atenção diante das crescentes e graves violações de dados reportadas e da aprovação de leis de proteção de dados no mundo.

Apenas em 2021, houve aproximadamente 623,25 milhões de ataques de *ransomware* no mundo [Petrosyan 2024]. Este tipo de ataque impacta diretamente a privacidade dos dados, pois envolve o sequestro de dados digitais e a criptografia dos mesmos, exigindo o pagamento de um resgate para decriptá-los [Nogueira et al. 2024]. O sequestro dos dados torna-os vulneráveis a acesso não autorizado. A dimensão desse problema é significativa, ainda mais considerando setores sensíveis, como o financeiro [López 2024]. Por exemplo, a *First American Financial Corporation*, uma empresa do ramo de seguros, revelou que aproximadamente 44.000 informações pessoais foram comprometidas em 2023 por um desses ataques [Arghire 2024]. A preocupação com dados pessoais levou à criação de legislações como a Lei Geral de Proteção de Dados (LGPD), que prevê punições para casos de descumprimento [ANPD 2023]. Portanto, é essencial adotar medidas eficazes para preservar a privacidade e mitigar vazamentos de dados.

A anonimização é uma técnica amplamente utilizada para preservar a privacidade dos dados. Entretanto, ela sofre do impasse de necessitar manter as propriedades úteis dos dados ao mesmo tempo que realiza a anonimização. Para enfrentar esse desafio, a literatura explora técnicas como *black marking*, generalização e criptografia com preservação de formato [Dijkhuizen and Ham 2018]. Contudo, muitas dessas técnicas resultam na perda de informações [Dijkhuizen and Ham 2018]. Essa questão é notável no caso da anonimização de endereço do Protocolo da Internet (do inglês *Internet Protocol* - IP). Esses endereços seguem uma semântica em sua formatação, indicando, por exemplo, o subconjunto de bits relacionado à rede e o subconjunto de bits relacionado ao dispositivo [Isaac 2016]. Para preservar essa informação, existe a técnica de anonimização com preservação de prefixo, que mantém a correspondência de IPs com o mesmo prefixo, permitindo uma análise de rede mais precisa sem comprometer a privacidade [Xu et al. 2002]. Entretanto, essa técnica é suscetível a ataques semânticos, em que o atacante identifica uma relação entre os dados originais e os anonimizados, permitindo, assim, a reversão ao valor original do IP [Xu et al. 2002]. A literatura tem explorado soluções para mitigar esses ataques, como a utilização de multi-visão dos dados [Gu and Dong 2023] ou da cifra de Feistel [Dandyan et al. 2022]. Contudo, essas abordagens enfrentam desafios, como a complexidade de lidar com muitos arquivos simultaneamente [Gu and Dong 2023] ou deixam de utilizar propriedades criptográficas que aumentam a segurança da anonimização.

O método PROTECT, um método de anonimização com PReservação de prefixO para IPv4 baseada em uma cifra de feisTEl desbalanCeada e heTerogênea, apresentado nesse trabalho, é uma solução de anonimização com preservação de prefixo em endereços IPv4 baseada na cifra de Feistel desbalanceada e heterogênea [Schneier and Kelsey 1996]. Ele mantém o prefixo de endereço IP e utiliza propriedades da cifra para melhorar a segurança da anonimização. A implementação do algoritmo realiza as iterações da cifra. Entretanto, ao invés de dividir o endereço IP de maneira única e na metade, ele considera as diferentes classes de IP e, portanto, divide de maneira particular cada uma. Outro fator aplicado é a utilização de duas funções de anonimização distintas que dificulta identificar a informação propagada durante as iterações [Schneier and Kelsey 1996]. O objetivo do método é garantir que o IP original não seja trivialmente inferido por um atacante.

A avaliação de desempenho segue dois conjuntos de experimentos, um considera como carga de trabalho a captura 52 do *dataset* CTU-13 [Garcia 2011] e outro considera

a captura 24 do *dataset* de IoT [Sivanathan et al. 2018]. Para cada carga de trabalho, os experimentos seguem três cenários. O primeiro cenário visa medir o efeito avalanche, uma propriedade usada para verificar a eficácia da perturbação gerada por operações de substituição e permutação. O segundo cenário aplica a entropia nos endereços originais e anonimizados para verificar se houve um aumento ou diminuição na incerteza dos endereços. O terceiro cenário compara o método PROTECT com a anonimização por cifra de Feistel com preservação de prefixo [Dandyan et al. 2022]. Nele, varia-se o percentual de conhecimento do atacante sobre a relação entre IPs originais e anonimizados (10% a 50%), permitindo inferir os IPs originais.

O restante deste trabalho procede como segue. A Seção 2 apresenta os trabalhos relacionados à anonimização. A Seção 3 detalha a solução proposta. A Seção 4 apresenta a avaliação de desempenho. Por fim, a Seção 5 conclui este trabalho.

2. Trabalhos Relacionados

A anonimização com preservação de prefixo busca equilibrar a privacidade e a utilidade dos dados. O TCPdpriv foi a primeira solução de anonimização de endereços IP com preservação de prefixo proposta [Minshall 1996]. Sua implementação utiliza uma tabela para armazenar a relação entre o IP e sua anonimização, garantindo a consistência dos dados [Xu et al. 2002]. No entanto, essa abordagem não suporta anonimização paralela, limitando o desempenho em grandes volumes de dados e aumenta o tempo de execução. Para superar essa limitação, Xu et al. (2002) propuseram o Crypto-PAn, um algoritmo criptográfico que elimina a necessidade de tabelas e permite anonimização paralela [Xu et al. 2002]. Apesar disso, o Crypto-PAn permanece vulnerável a ataques semânticos, que exploram padrões previsíveis nos endereços IP para inferir informações.

Para lidar com o ataque de semântica, Mohammady et al. (2018) propuseram gerar múltiplas visões falsas dos dados, criando vários arquivos de modo que o adversário não consiga identificar a anonimização verdadeira. Para que essa abordagem seja eficaz, é necessário que as visões sejam indistinguíveis [Mohammady et al. 2018]. Contudo, o vazamento de qualquer relação de prefixo compromete essa indistinguibilidade. Para contornar esse problema, Gu e Dong (2023) propuseram uma solução chamada PD-PAn, que utiliza o algoritmo Crypto-PAn com a chave $K0$ e, em seguida, segmenta a base de dados em partes, aplicando uma função distinta para cada uma com a chave $K1$, de modo a eliminar o relacionamento de prefixos nas visões falsas [Gu and Dong 2023]. A desvantagem dessa abordagem é que, para aumentar a privacidade, é necessário gerar muitas visões dos dados, aumentando o tempo de análise e o uso de recursos computacionais.

Visando aumentar a resistência contra ataques baseados em semântica Dandyan et al. (2022) propuseram uma solução para anonimização de endereços IP baseada na cifra de Feistel. A técnica divide o endereço IP binário em duas metades e aplica permutações e substituições controladas por uma função de anonimização, mas, para preservar o prefixo, não realiza a operação XOR entre as metades. Isso embaralha a estrutura do endereço, dificultando a inferência de informações sensíveis e mantendo a utilidade dos dados para análises. Contudo, a abordagem não aproveita totalmente as propriedades de segurança da cifra de Feistel, especialmente contra ataques lineares e diferenciais [Schneier and Kelsey 1996], o que pode comprometer a robustez da anonimização.

A literatura reconhece que a anonimização com preservação de prefixo, embora

útil, é vulnerável a ataques semânticos [Xu et al. 2002]. Soluções propostas enfrentam limitações, seja pela necessidade de aprimorar a segurança [Dandyan et al. 2022], seja pelo alto custo de gerenciamento de grandes volumes de arquivos [Gu and Dong 2023]. Nesse contexto, este trabalho propõe uma nova abordagem para a anonimização com preservação de prefixo que oferece um nível aprimorado de segurança. Ao invés de depender de soluções que exigem o manuseio de múltiplos arquivos ou operações complexas, a proposta apresentada simplifica o processo ao eliminar a necessidade de visões falsas. Utilizando propriedades da cifra de Feistel, a abordagem assegura a privacidade sem comprometer a praticidade da solução, superando limitações de abordagens prévias.

3. Método

Essa seção descreve o método proposto, denominado PROTECT, que consiste em duas etapas principais: (1) classificação, transformação para formato binário e divisão do endereço IP; e (2) geração de chaves, permutação, substituição e concatenação das partes. A Figura 1 ilustra essas etapas, com os blocos laranja representando a primeira etapa, onde há a classificação e divisão, e os azuis, a segunda, que aborda a geração de chaves, a permutação e concatenação. Para melhor compreensão, a Subseção 3.1 detalha a classificação e divisão realizadas na Etapa 1. Já a Subseção 3.2 aborda a Etapa 2.

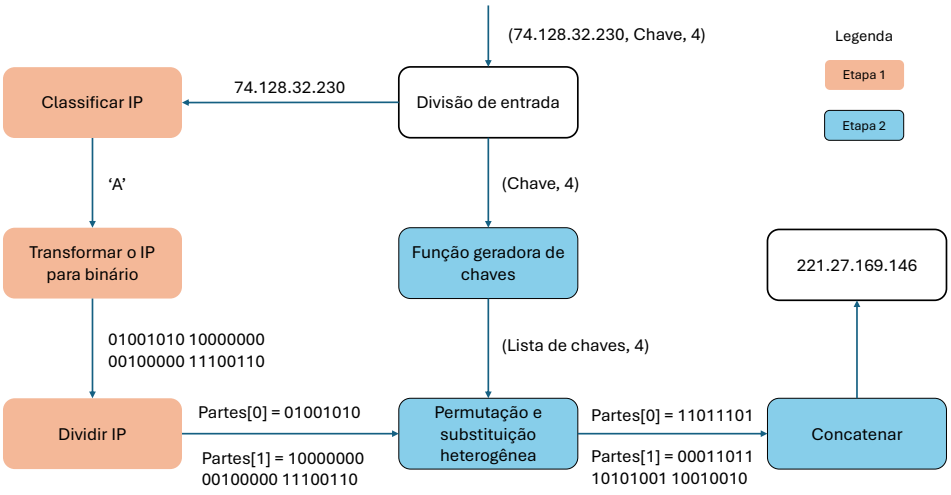


Figura 1. Fluxograma de etapas da anonimização

Conforme ilustrado no Algoritmo 1, que representa as etapas seguidas, todas as chaves são inicialmente geradas e armazenadas em um vetor de tamanho N (linha 2). Em seguida, para cada endereço IP a ser anonimizado (linhas 4 e 5), o IP é classificado conforme as classes IPV4 e seus bits de rede são identificados. O endereço IP é então transformado em binário e dividido em duas partes, de acordo com a classe identificada (linhas 6 a 8). Então, aplica-se a cifra de Feistel heterogênea (linha 9) com duas funções F e F', que será detalhada posteriormente. Por fim, as duas partes são concatenadas para gerar o endereço IP anonimizado (linha 10).

3.1. Classificação, transformação para binário e divisão

A classificação do endereço IP constitui a etapa inicial do método, sendo essencial para viabilizar uma divisão desbalanceada que aumenta a resistência contra ataques lineares e di-

Algoritmo 1 Algoritmo PROTECT

```
1: function PROTECT(lista_IP, chave, N : número de rodadas)
2:     K[N] ← Funcao_Geradora_Chaves(chave)
3:     for IP ∈ lista_IP do
4:         classe ← Classifica_IP(IP)
5:         bits_rede ← Encontra_Bit_Rede(classe)
6:         bin ← Binario(IP)
7:         partes[0] ← bin[0 : bits_rede − 1]
8:         partes[1] ← bin[bits_rede : ]
9:         partes ← Feistel_Heterogenea(partes, N)
10:        resultado ← Concatena(partes[0], partes[1])
11:    end for
12:    return resultado
13: end function
```

ferenciais [Isaac 2016]. Essa classificação é realizada com base nos dígitos do prefixo, ou seja, nos bits iniciais fixos que identificam *hosts* pertencentes à mesma rede [Isaac 2016]. Por exemplo, o endereço IP 170.56.211.113 é classificado como pertencente à classe B, com os 16 primeiros bits representando os bits de rede. O próximo passo transforma o endereço IP para o seu formato em binário. Isso é importante, pois, ao trabalhar com o IP no formato tradicional com strings de quatro números decimais separados por pontos, pode-se gerar caracteres que não são esperados, como ocorre em algumas implementações de anonimização com preservação de formato [Dunkelman et al. 2020].

$$F(a) = a_1{}'a_2{}'...a_n{}' \tag{1}$$

A Equação 1 descreve a formalização da anonimização proposta pelo Crypto-PAn. Onde cada bit $a_i{}'$ é calculado através $a_i \oplus f(a_0, a_1, ..., a_{i-1})$. Isso significa que cada bit anonimizado depende do bit original e de uma função aplicada aos bits anteriores [Xu et al. 2002]. Após a classificação e a conversão do endereço IP para formato binário, realiza-se a divisão na etapa subsequente. Conforme observado no Algoritmo 1, cada parte dessa divisão é armazenada em uma posição do vetor `partes`. A parte relativa aos bits de rede ficam na posição 0 e a parte relativa aos bits de *host* ficam na posição 1. A Cifra de Feistel, por padrão, é realizada dividindo o objeto a ser anonimizado exatamente no meio [Dandyan et al. 2022]. Neste método, o tamanho das partes depende da classe identificada anteriormente, permitindo uma Cifra de Feistel Desbalanceada, com resistência a ataques lineares e diferenciais [Schneier and Kelsey 1996].

Considerando a relação entre a classe de um endereço IP, seus bits de rede e os bits restantes para o número de *host*, é possível dividir o IP conforme seus bits de rede. Por exemplo, um IP da classe A pode ser dividido em 8 bits à esquerda e 24 bits à direita. Para os endereços das classes D e E, não há separação de bits de rede, pois essas classes são reservadas. Assim, a implementação deste método assume que a divisão será a mesma realizada para a classe C (Fig. 2). Utilizando como exemplo o endereço IP 170.56.211.113, ele é convertido para binário e dividido em duas partes. Aplicando a função de transformação para binário apresentada na linha 6 do Algoritmo 1, o endereço resulta em 10101010 00111000 11010011 01110001. Sendo de classe B, ele é dividido em duas partes de 16 bits: 10101010 00111000 e 11010011 01110001.

3.2. Função geradora de chaves, permutação e substituição

A segunda etapa do PROTECT, representada em azul na Figura 1, é a função geradora de chaves. Nela, uma chave é gerada para cada rodada do método, utilizando rotações de uma unidade para a esquerda [Dandyan et al. 2022]. Um exemplo é apresentado na Tabela 1, onde a chave inicial é a *string* "chave" e o número de rodadas é 4, com 3 rotações. A chave é convertida para binário com base na tabela ASCII de cada caractere e passa por rotações à esquerda. Por fim, gera-se uma lista de chaves correspondente ao número de rodadas.

N	Chave
4	01100011 01101000 01100001 01110110 01100101
3	11000110 11010000 11000010 11101100 11001010
2	10001101 10100001 10000101 11011001 10010101
1	00011011 01000011 00001011 10110011 00101011

Tabela 1. Exemplo de função geradora de chaves

O próximo passo envolve a permutação e substituição heterogênea utilizando os resultados das etapas anteriores e da função geradora de chaves. Na cifra de Feistel convencional, como mostrado nas Equações 2 e 3, o texto a ser cifrado é dividido ao meio. As metades, L (esquerda) e R (direita), passam por permutações e substituições. Na primeira rodada, o lado direito é colocado no esquerdo (Eq. 2), e o lado esquerdo, após substituição, vai para a metade direita (Eq. 3). A substituição envolve uma operação XOR entre a metade direita e a função de anonimização F, que recebe a metade direita e a chave (K_i) da i-ésima rodada como parâmetros.

$$L_{i+1} = R_i, \tag{2}$$

$$R_{i+1} = L_i \oplus F(K_i, R_i). \tag{3}$$

A cifra de Feistel proposta neste trabalho difere da convencional ao usar uma função de anonimização F que propaga informações do binário a ser anonimizado. Para aumentar a imprevisibilidade e evitar identificação das informações que estão sendo propagadas, utiliza-se duas funções distintas em vez de uma única, tornando o processo mais seguro. A cifra pode ser homogênea (convencional), quando a mesma função F é aplicada em todas as rodadas, ou heterogênea, quando funções diferentes são aplicadas nas rodadas ímpares e pares (Eqs. 4 e 5), utilizando a funções F e F' em rodadas alternadas (apresentadas nas Subseções 3.2.1 e 3.2.2). Na implementação da cifra (C) apresentada neste trabalho, a Equação 4 é aplicada nas rodadas ímpares, enquanto a Equação 5 é utilizada nas rodadas pares. Essa abordagem caracteriza a cifra como heterogênea, uma vez que utiliza funções de anonimização distintas em diferentes etapas do processo.

$$L_{i+1} = R_i, R_{i+1} = L_i \oplus F(K_i, R_i). \tag{4}$$

$$L_{i+1} = R_i, R_{i+1} = L_i \oplus F'(K_i, R_i). \tag{5}$$

A cifra de Feistel heterogênea dificulta a identificação da informação propagada, pois utiliza duas funções de anonimização diferentes em cada rodada [Schneier and Kelsey 1996]. O Algoritmo 2 mostra como esse trabalho implementa a cifra de Feistel heterogênea. A rodada (N), é um número par, por isso, na linha 2 do

Algoritmo 2 é requerido que N seja maior que 0, assim "partes[0]", que é o prefixo, terminará sempre na frente. Considerando o exemplo na Tabela 2, que mostra a sequência de permutação das partes ao longo das rodadas, se o valor de N fosse igual a 0, o prefixo acabaria na parte de trás do resultado anonimizado.

Algoritmo 2 Cifra de Feistel Heterogênea

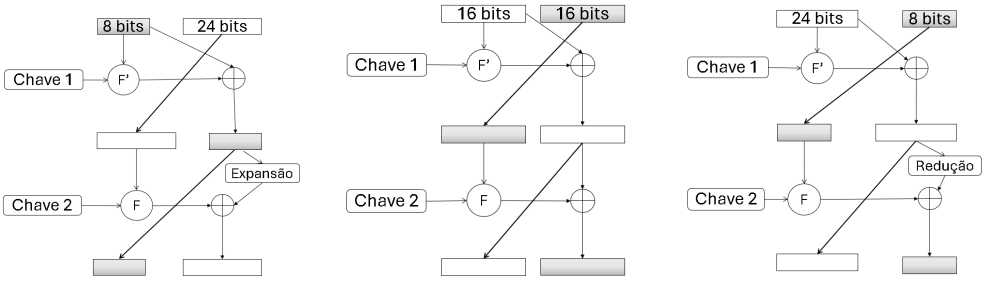
```
1: function FEISTEL_HETEROGENEA(partes, N)
2:     while N > 0 do
3:         if N é par then
4:             auxiliar ← partes[1]
5:             partes[1] ← partes[0] ⊕ F'(K_N, partes[0])
6:             partes[0] ← auxiliar
7:         end if
8:         if N é impar then
9:             auxiliar ← partes[1]
10:            partes[1] ← partes[0] ⊕ F(K_N, partes[1])
11:            partes[0] ← auxiliar
12:        end if
13:        N ← N − 1
14:    end while
15: end function
```

A figura 2 mostra o processo de separar e permutar para cada classe e rodadas. Nas rodadas pares, ocorre uma substituição utilizando a operação XOR. No entanto, para garantir a preservação do prefixo, a função F' não pode utilizar "partes[1]", pois a parte de *host* difere entre endereços IP que compartilham o mesmo prefixo. Por exemplo, considere duas listas binárias: [1,1,1,1,0,1,1,0] e [1,1,1,1,1,1,0,0]. Os 4 últimos bits dessas listas, que correspondem a "partes[1]" ([0,1,1,0] e [1,1,0,0], respectivamente), são diferentes. Se esses elementos fossem aplicados na função F, os resultados seriam distintos. Ao realizar a operação XOR entre esses resultados e "partes[0]" de cada lista, o prefixo comum seria perdido, resultando em valores diferentes para cada uma. Para evitar essa perda de informação do prefixo, o método funciona de modo que a função F trabalhe apenas com o bloco esquerdo, utilizando-o como entrada e garantindo a preservação do prefixo durante o processo de anonimização.

Valor de N	Parte esquerda	Parte direita
inicio	Partes[0]	Partes[1]
4	Partes[1]	Partes[0]'
3	Partes[0]'	Partes[1]'
2	Partes[1]'	Partes[0]"
1	Partes[0]"	Partes[1]"
0	Partes[1]"	Partes[0]"'

Tabela 2. Exemplo da permutação com número de rodadas igual a 4

Quando a rodada é um número ímpar, o bloco trabalhado é o da direita, intitulado como "partes[1]" no Algoritmo 2. Nesse caso, como ele contém os bits de *host*, não há problema se houver uma diferença ao aplicar uma operação de XOR com "partes[0]". Para garantir uma solução heterogênea, as funções devem ser distintas. Assim, a função de anonimização F' é aplicada, recebendo como parâmetros o lado direito e a chave da rodada. Após as rodadas, as partes são concatenadas e formatadas como um endereço IP,



(a) Classe A (b) Classe B (c) Classe C, D e E

Figura 2. Processo de substituição e permutação

onde cada octeto possui 8 bits separados por pontos e o formato retornado pelo método será uma *string* (Fig. 2). As funções F e F', descritas a seguir, devem garantir confusão e difusão [Schneier and Kelsey 1996]. A confusão dificulta a inferência da chave a partir do texto cifrado, enquanto a difusão assegura que pequenas mudanças na entrada causem grandes alterações na saída. No método PROTECT, a confusão é obtida por substituição, e a difusão, por permutação e rotação.

3.2.1. Função F

A função de anonimização F deste método segue os passos ilustrados na Figura 3. No passo 1, é realizada uma substituição das partes em binário, provocando uma confusão necessária para o método. Em sequência, no passo 2 é realizada a expansão e confusão, onde é possível trabalhar com chaves maiores, o que aumenta a segurança da anonimização. No passo 3, o resultado obtido em 2 é colocado em uma operação XOR com a chave. Adiante, no passo 4, é realizada uma substituição redutora no resultado. Por fim, o binário, que está em forma de matriz, tem as suas colunas permutadas.

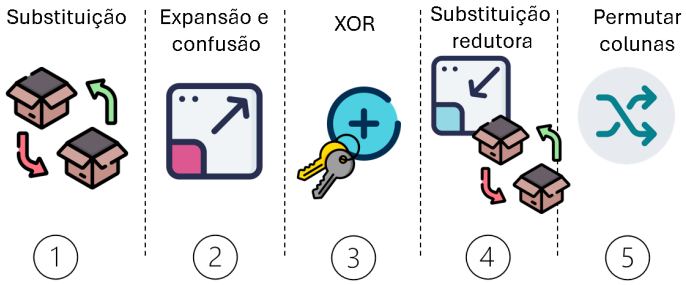


Figura 3. Processo de anonimização da função F

O Algoritmo 3 exemplifica a aplicação dessa função. Inicialmente, é realizada uma substituição de uma sequência de bits por outra (linha 2). Esta substituição inicial não busca diminuir o tamanho, conforme realizado na substituição redutora. A Tabela 3 exemplifica uma tabela de substituição, onde a sequência 00 é substituída por 01 e 10 por 11. Para uma boa confusão, uma sequência não pode ser substituída por ela mesma ou sua negação, como 00 por 00 ou 11 [Tarawneh 2023].

O passo seguinte é de expansão e permutação. Como a cifra de Feistel desbalanceada pode operar com partes de 8 bits, como o prefixo de um IP classe A, é necessário

Algoritmo 3 Função de anonimização F

1: **function** F(Chave, partes)
2: $sub \leftarrow Substituicao(partes)$
3: $EP \leftarrow Expansao\_Permutacao(sub)$
4: $resultado \leftarrow EP \oplus chave$
5: $sub\_novo \leftarrow Substituicao\_Redutora(resultado)$
6: $permuta \leftarrow Permutacao(sub\_novo)$
7: **end function**

Original	Imagem
00	01
01	00
10	11
11	10

Tabela 3. Exemplo de uma tabela de substituição

expandir esse tamanho para usar chaves maiores e aumentar a segurança da anonimização. Esse estágio consiste em duplicar elementos existentes para gerar confusão e expandir o tamanho da parte. Este passo utiliza multiplicação matricial, posicionando a parte em uma matriz de 4 colunas e aplicando uma matriz de permutação com 4 linhas. Um exemplo dessa operação é mostrado na Equação 6, onde os dois últimos elementos de cada linha são repetidos, expandindo a matriz de 8 para 12 bits. Embora as chaves tenham 40 bits, apenas 36 são usados, equivalentes ao tamanho expandido de uma parte de 24 bits.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{23} & a_{24} \end{pmatrix} \tag{6}$$

Posteriormente, aplica-se uma operação XOR entre a parte expandida e a chave, seguida da substituição redutora, que restaura o tamanho original e gera confusão. Por exemplo, a sequência 101, formada pelos bits externos 11 e o bit interno 0, poderia ser substituída por 10. O próximo passo do método realiza outra operação matricial na parte, em que as colunas da sua matriz serão permutadas. Essa modificação ocorre multiplicando a matriz por uma matriz de permutação à direita, permutando suas colunas. Essa operação pode ser exemplificada pela Equação 7, que mostra como seria um resultado possível dessa função. Neste exemplo, a coluna 3 foi para a primeira coluna, e a coluna 2 para a quarta coluna (vide Eq. 6).

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} a_{13} & a_{11} & a_{14} & a_{12} \\ a_{23} & a_{21} & a_{24} & a_{22} \end{pmatrix} \tag{7}$$

3.2.2. Função F'

A função F', ilustrada na Figura 4, segue cinco passos: (1) a parte binária é expandida por expansão e substituição; (2) o resultado passa por uma operação XOR com a chave; (3) realiza-se uma substituição redutora para retornar ao tamanho original; (4) as colunas da matriz são deslocadas; e (5) são permutadas por uma matriz de permutação.

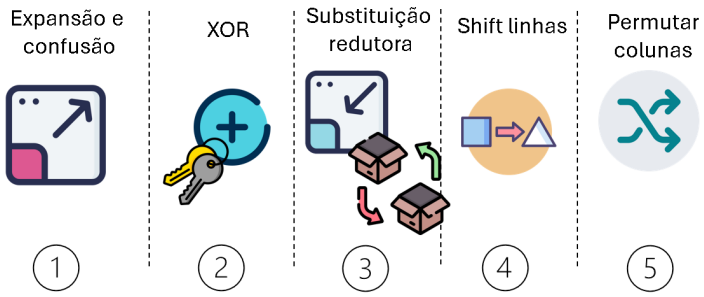


Figura 4. Processo de anonimização da função F'

O Algoritmo 4 mostra os passos destacados na Figura 4. Entretanto, como a maioria das funções já foi tratada na subseção anterior, o interesse dessa subseção é explicar sobre o passo de *shift* das linhas. Esse é o quarto passo do processo de anonimização da função F' onde todas as linhas, exceto a primeira, são deslocadas uma posição para a esquerda, conforme mostrado na Equação 8. Esse passo aumenta a difusão do algoritmo.

Algoritmo 4 Implementação da função F'

```
1: function F'(Chave, partes)
2:     EP ← Expansao_Permutacao(partes)
3:     resultado ← EP ⊕ chave
4:     sub_novo ← Substituicao_Redutora(resultado)
5:     shiftado ← Shift_Coluna(sub_novo)
6:     permuta ← Permutacao(shiftado)
7: end function
```

$$\begin{pmatrix} a_{13} & a_{11} & a_{14} & a_{12} \\ a_{23} & a_{21} & a_{24} & a_{22} \\ a_{33} & a_{31} & a_{34} & a_{32} \end{pmatrix} \rightarrow \begin{pmatrix} a_{13} & a_{11} & a_{14} & a_{12} \\ a_{21} & a_{24} & a_{22} & a_{23} \\ a_{31} & a_{34} & a_{32} & a_{33} \end{pmatrix} \tag{8}$$

4. Avaliação

A avaliação do sistema proposto segue dois experimentos. A captura 52 do *dataset* CTU-13 [Garcia 2011], utilizada no Experimento 1, contém 6.336.398 capturas, 39.885 endereços IP únicos, ocupa 555 MB e corresponde a 972 segundos. A captura 4 do *dataset* de IoT [Sivanathan et al. 2018], usada no Experimento 2, possui 1.048.576 capturas, 481 endereços IP únicos, 212 MB e 55.855 segundos. Em ambos os experimentos, apenas um segundo de cada *dataset* foi analisado. Os parâmetros incluem a chave *key_strength_nbits*, que simula uma chave forte, e a tabela de substituição da tabela 4, com as demais tabelas idênticas às dos exemplos. A principal diferença entre os experimentos está no *dataset* de entrada, enquanto os demais parâmetros foram padronizados.

Bit interno / Bits externos	00	01	10	11
0	00	10	00	10
1	01	11	01	11

Tabela 4. Exemplo de uma tabela de substituição redutora

Para a avaliação, utilizou-se o efeito avalanche, uma métrica essencial na criptografia que mede a confusão gerada por operações de substituição e permutação. Sua

ausência indica má aleatorização, possibilitando a relação entre entrada e saída. A métrica é avaliada pela diferença nos bits da saída original e após uma pequena alteração na entrada. A Equação 9 mostra que valores abaixo de 50% indicam criptografia inadequada, onde D é a diferença entre os bits originais e anonimizados, e T é o total de bits.

$$Efeito\ Avalanche = (D/T) * 100, \tag{9}$$

Outra métrica é a entropia, que mede a incerteza na previsão de uma variável. Em privacidade, ela representa o número de bits adicionais que um adversário precisa para identificar o elemento original. Neste trabalho, compara-se a entropia média dos endereços IP originais e anonimizados. Um aumento na entropia média indica maior aleatoriedade dos dados, enquanto uma diminuição aponta para maior previsibilidade. O valor é calculado pela Equação 10, onde $H(X)$ é a entropia e $|X|$ é o número de elementos em X.

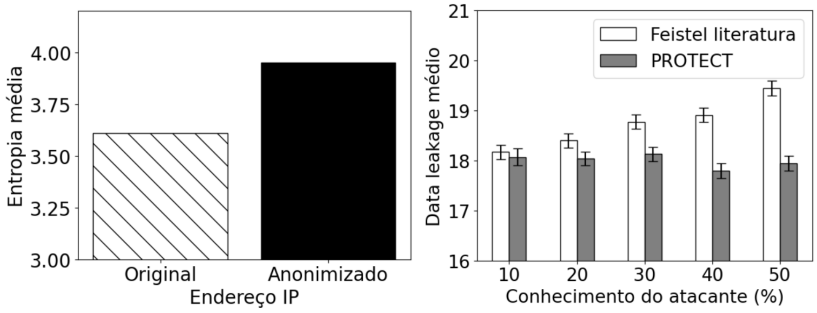
$$H(X) \leq \log |X| \tag{10}$$

Este trabalho também emprega uma métrica quantitativa para mensurar o vazamento de dados, ou seja, quantifica o vazamento de informações sensíveis de um conjunto de dados que deveria permanecer anonimizado. Essa métrica é crucial para avaliar a eficácia das técnicas de anonimização e garantir que os dados protegidos não possam ser facilmente reidentificados, especialmente na presença de um atacante com conhecimento parcial. A métrica inclui a proporção do conhecimento do atacante e a medida do vazamento. O primeiro quantifica a informação prévia do atacante, variável conforme uma porcentagem, e o segundo mede o vazamento de octetos inferidos, que deve ser minimizado. Um exemplo prático ocorre quando o atacante conhece 20% dos pares de dados originais e anonimizados, tentando inferir os dados anonimizados a partir do restante. O resultado é a soma dos octetos inferidos corretamente.

4.1. Experimentos

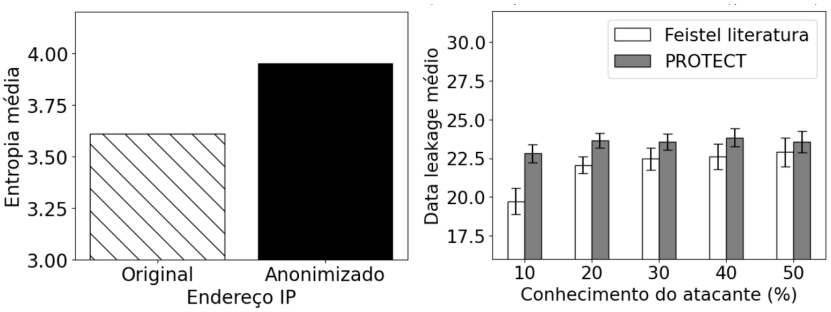
No *Experimento 1* (*dataset* CTU-13 [Garcia 2011]), a avaliação utilizou o primeiro segundo, considerando três cenários baseados nas métricas explicadas. No cenário 1, o cálculo do efeito avalanche médio abrange todo o conjunto de dados, em vez de um único endereço IP, como frequentemente ocorre na literatura [Dandyan et al. 2022]. Essa abordagem permite uma análise mais abrangente, considerando variações do efeito avalanche em diferentes endereços IP e mitigando possíveis vieses. Para a implementação, utiliza-se uma função que determina quantos bits diferem dos bits originais. O valor da diferença é dividido pelo número total de bits de um endereço IP e multiplicado por 100 (Eq. 9).

No cenário 2, a entropia média é calculada para os endereços IP originais e anonimizados, conforme abordado na literatura [Mivule and Anderson 2015]. A entropia é determinada a partir da sequência binária do endereço IP, utilizando scipy.stats.entropy do Python. Para cada endereço IP, calcula-se a entropia individual, que é somada e dividida pelo total de endereços para obter a entropia média. Essa análise permite avaliar se a entropia dos endereços aumenta ou diminui. Um aumento indica maior incerteza nos dados, enquanto uma diminuição revela que a anonimização torna os dados mais previsíveis. Outro ponto a se considerar é que, segundo a Equação 10, a entropia máxima que esse experimento deve ter é 5, pois $|X|$ é 32 e $\log 32$ é 5. Onde 0 seria o valor mínimo e 5 é o valor máximo alcançado.



(a) Entropia média (b) Vazamento de dados médio

Figura 5. Entropia média e vazamento de dados médio no dataset CTU-13



(a) Entropia média (b) Vazamento de dados médio

Figura 6. Entropia média e vazamento de dados médio no dataset de IoT

No Cenário 3, realiza-se a medida de vazamento de dados, onde o *dataset* é anonimizado com a anonimização baseada em cifra de Feistel da literatura e o PROTECT, o crypto-PAn foi desconsiderado para essa análise pois não possui robustez contra ataques de semântica.Neste experimento, a configuração de conhecimento do atacante, pode variar de 10%, 20%, 30%, 40% e 50%. O conhecimento do atacante refere-se à quantidade de pares de endereços originais e anonimizados disponíveis para ele. Por exemplo, em um total de 100 endereços distintos, 10% significa que o atacante conhece os pares correspondentes de 10 endereços IP. Para simular o conhecimento do atacante, este trabalho emprega uma rede neural para aprender a relação entre sequências binárias de endereços IP e seus resultados de anonimização. A partir desse ponto, a rede tenta inferir o endereço IP original a partir de sua versão anonimizada utilizando o *tensorflow* do python. O conhecimento do atacante é o mesmo do Experimento 1.

4.2. Resultados

A Figura 5 apresenta os resultados do *Experimento 1*. Na métrica de efeito avalanche médio, calculada pela média dos efeitos para cada endereço IP, o valor obtido foi 52,5. A Figura 5(a) mostra que a entropia média dos endereços originais foi aproximadamente 3,6, enquanto a dos anonimizados alcançou cerca de 4,0. Já a Figura 5(b) compara o vazamento de dados médio entre a anonimização com preservação de prefixo baseada na cifra de Feistel e o método PROTECT (barra cinza). Com 10% de conhecimento do atacante, o PROTECT apresentou vazamento médio de 18,07; com 20%, 18,04; com 30%, 18,13; com 40%, 17,8; e, com 50%, 17,94.

A Figura 6 apresenta os resultados do *Experimento 2*. O efeito avalanche médio foi de 40,62 na captura 4 do *dataset* de IoT [Sivanathan et al. 2018]. A Figura 5(a) mos-

tra que a entropia média dos endereços originais atingiu aproximadamente 3,6, enquanto os anonimizados alcançaram 3,9. Já a Figura 6(b) compara o vazamento de dados médio entre a anonimização com preservação de prefixo baseada na cifra de Feistel (barras brancas) e o método PROTECT (barra cinza). Para 10% de conhecimento do atacante, o PROTECT apresentou vazamento médio de 22,81; para 20%, 23,64; para 30%, 23,57; para 40%, 23,84; e para 50%, 23,56.

4.3. Discussão

O método proposto apresentou avanços significativos na privacidade dos dados em comparação com a literatura. No *Experimento 1*, o efeito avalanche médio foi de 52,5 (CTU-13), indicando boa confusão dos dados. A entropia média dos endereços originais foi 3,6, enquanto os anonimizados atingiram 4,0 (Fig.5(a)), evidenciando um aumento na incerteza dos endereços IP. Além disso, o vazamento de dados médio (Fig.5(b)) mostrou que o PROTECT superou a cifra de Feistel [Dandyan et al. 2022]. Com 10% de conhecimento do atacante, o PROTECT obteve um vazamento médio de 18,07, mantendo 17,94 com 50% de conhecimento, enquanto a literatura alcançou 19,45. A partir de 20% de conhecimento do atacante, o PROTECT apresentou melhor desempenho, com 18,04 de quebras, em comparação aos 18,4 da literatura, ou seja, houve uma quantidade menor de quebras. Esses resultados reafirmam a superioridade do método em cenários onde o conhecimento do atacante ultrapassa 20%, assegurando maior privacidade em relação à abordagem da literatura [Dandyan et al. 2022].

No *Experimento 2*, o efeito avalanche médio foi 40,62 devido a baixa quantidade de endereços IP no segundo *dataset*.A entropia média semelhante à do Experimento 1: os endereços originais atingiram 3,6 e os anonimizados, 3,9 (Fig. 6(a)). O método PROTECT apresentou desempenho equivalente literatura, com vazamento médio de 22,81 para 10% de conhecimento do atacante, estabilizando em 23,56 para 50%. Essa similaridade reflete a menor quantidade de endereços IP únicos no *dataset* comparado ao Experimento 1. Apesar disso, o método PROTECT mantém resistência contra ataques linear e diferencial, além de simplificar a aplicação ao evitar o uso de múltiplos arquivos, como ocorre em outros estudos [Gu and Dong 2023], [Mohammady et al. 2018]. Embora o CIDR tenha substituído as classes, o método continua relevante, pois a divisão pode seguir outros critérios convenientes além das classes.

5. Conclusão

Esse artigo aborda o desafio da preservação da privacidade diante do aumento de ataques cibernéticos que comprometem dados pessoais e sensíveis. As soluções atuais de anonimização enfrentam o dilema de preservar as propriedades úteis dos dados enquanto garantem a segurança. Para avançar a literatura, o trabalho propõe o método PROTECT, uma solução inovadora baseada em cifra de Feistel desbalanceada e heterogênea, que preserva prefixos de endereços IPv4 enquanto dificulta a inferência de dados originais por atacantes. A eficácia do método foi avaliada por experimentos que analisam o efeito avalanche, a entropia e comparações com técnica existente, demonstrando seu potencial para melhorar a segurança da anonimização sem comprometer a utilidade dos dados. Trabalhos futuros investigarão formas de reduzir ainda mais a complexidade do algoritmo.

Agradecimentos

Este trabalho foi financiado pela FAPESP, bolsas 2022/06840-0, 2023/13294-4 e 2025/00612-3.

Referências

ANPD (2023). Anpd publica regulamento de aplicação de sanções administrativas Access 10/24. http://surl.li/ermeou.

Arghire, I. (2024). Data breachespersonal information of 44,000 compromised in first american cyberattack Access 06/24. https://abrir.me/ynYPG.

Dandyan, S., Louafi, H., and Sadaoui, S. (2022). A Feistel network-based prefix-preserving anonymization approach, applied to network traces. In *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, pages 1–11.

Dijkhuizen, N. V. and Ham, J. V. D. (2018). A survey of network traffic anonymisation techniques and implementations. *ACM Comput. Surv.*, 51(3).

Dunkelman, O., Kumar, A., Lambooij, E., and Sanadhya, S. K. (2020). Cryptanalysis of feistel-based format-preserving encryption. *Cryptology ePrint Archive*.

Garcia, S. (2011). Ctu-malware-capture-botnet-52: Scenario 11 in the ctu-13 dataset. `http://surl.li/awogqs`. Dataset generated in the Stratosphere Lab, CVUT University, Prague, Czech Republic.

Gu, X. and Dong, K. (2023). PD-PAn: prefix- and distribution-preserving internet of things traffic anonymization. *Electronics*, 12(20).

Isaac, S. (2016). Comparative analysis of ipv4 and ipv6. *International Journal of Computer Science and Information Technologies*, 7(2):675–678.

López, M. (2024). Ataques de ransomware Access 07/24. https://bit.ly/40y5vf0.

Minshall, G. (1996). *TCPdpriv Command Manual*. `https://bit.ly/4gK4Exj`.

Mivule, K. and Anderson, B. (2015). A study of usability-aware network trace anonymization. In *2015 Science and Information Conference (SAI)*, pages 1293–1304.

Mohammady, M., Wang, L., Hong, Y., Louafi, H., Pourzandi, M., and Debbabi, M. (2018). Preserving both privacy and utility in network trace anonymization. In *ACM SIGSAC*, CCS '18, page 459–474, New York, NY, USA. ACM.

Nogueira, M., Borges, L. F., Borges, L. F., Neira, A. B., Albano, L., and Coelho, K. K. (2024). Ciência de dados aplicada à cibersegurança: Teoria e prática. In de Computação (SBC), S. B., editor, *Minicursos do SBSeg 24*, pages 1–48.

Petrosyan, A. (2024). Annual number of ransomware attempts worldwide from 2017 to 2023 Access 06/24. https://bit.ly/3WhG8LM.

Schneier, B. and Kelsey, J. (1996). Unbalanced feistel networks and block cipher design. In *International Workshop on Fast Software Encryption*, pages 121–144. Springer.

Sivanathan, A., Habibi Gharakheili, H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V. (2018). Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Trans. on Mobile Computing*, 18(8):1745–1759.

Tarawneh, M. (2023). Perspective chapter: Cryptography–recent advances and research perspectives. *Biometrics and Cryptography*.

Xu, J., Fan, J., Ammar, M., and Moon, S. (2002). Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. In *10th IEEE ICNP, 2002*, pages 280–289.