Trabalho Prático 2: Honey Encryption

Lucas Emanuel de Oliveira Santos
Universidade Federal do Paraná
leos22@inf.ufpr.br

21 de novembro de 2025

1 Introdução

A Honey Encryption (HE) é uma técnica de criptografia apresentada em 2014 por Ari Juels e Thomas Ristenpart na conferência Eurocrypt. A proposta surgiu da observação de que sistemas protegidos por senha frequentemente sofrem com baixa entropia, o que os torna vulneráveis a ataques de dicionário e força bruta. Mesmo utilizando criptografia moderna, como AES ou SHA-256, um invasor pode testar inúmeras senhas até encontrar uma que gere um texto coerente, identificando assim a chave correta.

Antes da Honey Encryption, já existiam mecanismos conhecidos como *honey systems* — por exemplo, *honeypots*, *honey tokens* e *honey accounts* — que utilizam dados falsos para enganar invasores. A HE aplica esse mesmo princípio ao processo criptográfico: em vez de retornar um texto ilegível quando a senha é errada, ela produz uma mensagem falsa, porém plausível, tornando praticamente impossível distinguir erros de acertos.

2 Estratégia de Defesa

Em ataques tradicionais de força bruta, tentativas com chaves incorretas retornam textos aleatórios. Isso facilita a vida do atacante, pois o texto correto se destaca imediatamente por ser o único que faz sentido.

A Honey Encryption neutraliza essa vantagem transformando o espaço das mensagens por meio de um **DTE** (Distribution-Transforming Encoder), que distribui as mensagens legítimas em um espaço com probabilidade uniforme. Assim:

- Com a senha correta, o DTE reconstrói a mensagem legítima.

- Com qualquer senha incorreta, o sistema gera uma mensagem falsa.

Dessa forma, todas as saídas parecem válidas, impedindo o atacante de saber se encontrou a chave certa.

3 Funcionamento do Algoritmo

A Honey Encryption utiliza um componente chamado **Distribution-Transforming Encoder** (DTE). Ele garante que qualquer tentativa de descriptografia — correta ou incorreta — produza uma mensagem que pareça válida. Para isso, o DTE associa cada mensagem possível a uma região do intervalo $[0, 1)$, onde mensagens mais prováveis recebem intervalos maiores e mensagens menos prováveis recebem intervalos menores.

3.1 Modelagem da Distribuição das Mensagens

Para funcionar, o DTE precisa conhecer o tipo de dado protegido (por exemplo, cartões de crédito, senhas, identificadores). Ele usa essas características para construir a distribuição das mensagens possíveis. Com base nessa distribuição, o intervalo $[0, 1)$ é dividido em várias fatias contíguas, cada uma representando uma mensagem válida. Isso garante que qualquer ponto dentro do intervalo possa ser convertido em uma mensagem coerente.

3.2 Como as Mensagens Falsas São Geradas

O DTE não guarda mensagens falsas. Elas surgem automaticamente do mapeamento dos intervalos. Cada mensagem ocupa um subintervalo, então qualquer chave incorreta produz um ponto aleatório dentro de $[0, 1)$, que inevitavelmente cai em algum desses subintervalos. O DTE interpreta esse ponto como uma mensagem válida, gerando uma *honey message*: falsa, porém estruturada corretamente.

3.3 Codificação

Para codificar uma mensagem, o DTE: encontra o intervalo correspondente a ela; escolhe um ponto aleatório dentro desse intervalo; cifra esse ponto com a chave derivada da senha. O ciphertext final é apenas esse ponto cifrado.

3.4 Decodificação com Chave Correta

Com a senha correta, a descriptografia recupera exatamente o ponto usado na codificação. Como ele pertence ao intervalo da mensagem original, o DTE reconstrói o dado verdadeiro.

3.5 Decodificação com Chave Incorreta

Com uma senha incorreta, o ponto obtido após a descriptografia será outro ponto qualquer dentro de $[0, 1)$. Esse novo ponto sempre cairá em um subintervalo válido do DTE, e o sistema reconstruirá a mensagem correspondente, gerando uma saída falsa, mas plausível.

3.6 Exemplo: Cartão de Crédito

Um número de cartão de crédito criptografado é vulnerável a ataques de força bruta porque nem toda sequência de dígitos é igualmente provável. Um cartão pode ter entre 13 e 19 dígitos, embora 16 seja o mais comum. Além disso, deve possuir um *Issuer Identification Number* (IIN) válido e o último dígito deve corresponder ao *checksum*. Um atacante também pode levar em conta a popularidade das bandeiras: por exemplo, um IIN da MasterCard é geralmente mais provável do que um da Diners Club Carte Blanche.

A Honey Encryption protege contra esses ataques ao mapear números de cartão para um espaço maior, no qual cada região corresponde à probabilidade de legitimidade do cartão. Números com IIN inválido ou checksum incorreto simplesmente não são mapeados (isto é, têm probabilidade zero de serem legítimos). Cartões de grandes bandeiras, como MasterCard e Visa, ocupam regiões maiores desse espaço, enquanto bandeiras menos populares ocupam regiões menores. Assim, um atacante realizando força bruta só verá números de cartão que parecem legítimos, distribuídos com frequências compatíveis com o mundo real.

Suponha que o cartão verdadeiro seja:

```
4539 4512 0398 7356
```

Se uma senha incorreta for usada, o ponto obtido após a descriptografia cairá em outra região do DTE — por exemplo, uma região correspondente a uma bandeira mais comum como a MasterCard. O DTE então reconstruirá automaticamente um cartão falso, mas válido, como:

```
5294 9038 1142 6625
```

Ambos parecem legítimos, seguem as regras estruturais e passam nas validações necessárias, tornando impossível ao invasor distinguir qual é o verdadeiro.

4 Conclusão

A Honey Encryption representa uma solução promissora para fortalecer a segurança de dados sensíveis, especialmente em sistemas baseados em senhas de baixa entropia. Sua capacidade de gerar mensagens falsas torna ataques de força bruta significativamente menos eficazes, já que cada tentativa incorreta resulta em dados que parecem legítimos para o invasor.

Além de complementar os mecanismos tradicionais de criptografia, a Honey Encryption atua como uma camada adicional que continua protegendo as informações mesmo se a criptografia convencional for comprometida. Isso a torna especialmente relevante em cenários envolvendo dados de consumidores, autenticação baseada em senhas e sistemas suscetíveis a vazamentos de credenciais.

Embora seja promissora, a técnica ainda enfrenta obstáculos significativos. Gerar informações falsas suficientemente convincentes é uma tarefa complexa, principalmente em sistemas que armazenam grandes quantidades de dados. Enquanto contas simples ou dados estruturados podem ser falsificados com facilidade, conteúdos mais complexos exigem modelagem detalhada para enganar um atacante experiente. Esse ponto permanece como um dos principais obstáculos para a adoção prática da Honey Encryption. Assim, apesar de seu potencial, a técnica segue limitada ao ambiente acadêmico.

Referências

[1] Honey encryption. `https://en.wikipedia.org/wiki/Honey_encryption`.

[2] Esther Omolara Abiodun et al. A comprehensive review of honey encryption scheme. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 2019.

[3] Ari Juels and Thomas Ristenpart. Honey encryption: Security beyond the brute-force bound. *EUROCRYPT*, 2014.

[4] Daniel Zuot. Honey encryption implementation. GitHub repository. Disponível em: `https://github.com/danielzuot/honeyencryption`.