Aplicação Conceitual de Criptografia Homomórfica

Pedro Alves,
pdroalves@gmail.com

MO422 - Algoritmos Criptográficos
Professor: Julio López
Instituto de Computação,
Universidade Estadual de Campinas

Dezembro de 2014

- O recente barateamento do poder computacional pelo paradigma de computação em nuvem tornou popular o uso desse tipo de serviço. **Há interesse em terceirizar a instalação, manutenção e a escalabilidade** de servidores.
- As técnicas para garantir a privacidade dos dados, contudo, ainda não são suficientes para este cenário.

- O recente barateamento do poder computacional pelo paradigma de computação em nuvem tornou popular o uso desse tipo de serviço. **Há interesse em terceirizar a instalação, manutenção e a escalabilidade** de servidores.
- As técnicas para garantir a privacidade dos dados, contudo, ainda não são suficientes para este cenário.
- O uso de criptossistemas com a propriedade do homomorfismo é ideal para melhorar os critérios de confidencialidade dos dados nesse contexto.

Em nosso trabalho tivemos a intenção de explorar a propriedade homomórfica de alguns criptossistemas.
Para isso, realizamos a implementação dos esquemas de Paillier e Elgamal em Python e de um sistema conceito de votação eletrônica que opera sobre dados cifrados.

Agenda

1. Definição
   - Criptografia Homomórfica
2. Criptossistemas
   - Paillier
   - Elgamal
3. Aplicação
   - Definição do problema
   - Implementação
     - Simulação de votação
     - Gerador de números pseudo-aleatórios
     - Testes de consistência
4. Conclusão

Agenda

1. Definição
   - Criptografia Homomórfica
2. Criptossistemas
   - Paillier
   - Elgamal
3. Aplicação
   - Definição do problema
   - Implementação
     - Simulação de votação
     - Gerador de números pseudo-aleatórios
     - Testes de consistência
4. Conclusão

Definição (Criptossistema homomórfico)

Seja E uma função de cifração e D a função de decifração correspondente. Sejam m_1 e m_2 dados abertos. A dupla (E, D) forma uma cifra dita homomórfica com respeito a um operador $\diamond$ se a seguinte propriedade for satisfeita:

$$D\left(E\left(m_1\right) \diamond^* E\left(m_2\right)\right) = m_1 \diamond m_2$$

Definição (Criptossistema homomórfico)

Seja E uma função de cifração e D a função de decifração correspondente. Sejam m_1 e m_2 dados abertos. A dupla (E, D) forma uma cifra dita homomórfica com respeito a um operador $\diamond$ se a seguinte propriedade for satisfeita:

$$D\left(E\left(m_1\right) \diamond^* E\left(m_2\right)\right) = m_1 \diamond m_2$$

É importante notar que essa definição **não impõe restrições** quanto a simetria do criptossistema.

Um criptossistema homomórfico pode ser classificado como parcial ou completo.

Definição (Criptossistema parcialmente homomórfico)

Um criptossistema é dito parcialmente homomórfico se satisfizer a definição de homomorfismo para a operação de adição ou de multiplicação.

Do inglês, *Partially Homomorphic Encryption - PHE*

Definição (Criptossistema completamente homomórfico)

Um criptossistema é dito completamente homomórfico se satisfizer a definição de homomorfismo para as operações de adição e multiplicação.

Do inglês, *Fully Homomorphic Encryption - FHE*

Agenda

1. Definição
   - Criptografia Homomórfica
2. Criptossistemas
   - Paillier
   - Elgamal
3. Aplicação
   - Definição do problema
   - Implementação
     - Simulação de votação
     - Gerador de números pseudo-aleatórios
     - Testes de consistência
4. Conclusão

Em nosso trabalho tivemos interesse apenas em criptossistemas parcialmente homomórficos.

- Proposto em 1985 por Taher Elgamal.
- Baseado no problema de Diffie-Hellman.

Definição (Geração de chaves)

1. *Escolha um primo p grande e um elemento gerado* α *do grupo cíclico* $\mathbb{Z}_p^*$.
2. *Escolha* $d \in \left\{Z_{p-1}^* - \{1\}\right\}$ *e defina* $\beta \equiv \alpha^d \mod p$.
3. *Defina* (p, α, β) *com a chave pública e* (d) *como a chave privada.*

Definição (Cifração)

Dado a chave pública (p, α, β) *e uma mensagem m,*

1. *Escolha aleatoriamente* $i \in \left\{ Z_{p-1}^{*} - \{1\} \right\}$.
2. *Calcule* $(k_E, c) = \left(\alpha^i, m\beta^i\right) \in Z_p^*$.

Definição (Cifração)

Dado a chave pública (p, α, β) *e uma mensagem* m,

1. *Escolha aleatoriamente* $i \in \left\{ Z_{p-1}^{*} - \{1\} \right\}$.
2. *Calcule* $(k_E, c) = \left(\alpha^i, m\beta^i\right) \in Z_p^{*}$.

Definição (Decifração)

Dado a chave privada (d), *uma cifra* c *e sua chave efêmera* k_E,

1. *Calcule* $m = c.k_E^{-d} \in \mathbb{Z}_p^{*}$.

Definição (Cifração)

Dado a chave pública (p, α, β) *e uma mensagem* m,

1. *Escolha aleatoriamente* $i \in \left\{ Z^*_{p-1} - \{1\} \right\}$.
2. *Calcule* $(k_E, c) = \left(\alpha^i, m\beta^i\right) \in Z^*_p$.

Definição (Decifração)

Dado a chave privada (d), *uma cifra* c *e sua chave efêmera* k_E,

1. *Calcule* $m = c.k_E^{-d} \in \mathbb{Z}^*_p$.

Corretude: $c.k_E^{-d} = \left(m.\beta^i\right) . \left(\alpha^i\right)^{-d} = m.\alpha^{d.i}.\alpha^{-d.i} = m.$

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Existem c_1 e c_2 tais que:

$$c_1 = m_1.\beta^{i_1} \mod p$$
$$c_2 = m_2.\beta^{i_2} \mod p.$$

.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Existem c_1 e c_2 tais que:

$$\begin{aligned} c_1 &= m_1.\beta^{i_1} \mod p \\ c_2 &= m_2.\beta^{i_2} \mod p. \end{aligned}$$

Logo,

$$\begin{aligned} c_1 \times c_2 &= \left(m_1.\beta^{i_1} \mod p\right) \times \left(m_2.\beta^{i_2} \mod p\right) \\ &= m_1.m_2.\beta^{i_1}.\beta^{i_2} \mod p \\ &= m.\beta^{i} \mod p \\ &= c \end{aligned}$$

onde $m = m_1 \times m_2$ e $i = i_1 + i_2$.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Existem c_1 e c_2 tais que:

$$c_1 = m_1.\beta^{i_1} \mod p$$
$$c_2 = m_2.\beta^{i_2} \mod p.$$

Logo,

$$\begin{aligned} c_1 \times c_2 &= \left(m_1.\beta^{i_1} \mod p\right) \times \left(m_2.\beta^{i_2} \mod p\right) \\ &= m_1.m_2.\beta^{i_1}.\beta^{i_2} \mod p \\ &= m.\beta^{i} \mod p \\ &= c \end{aligned}$$

onde $m = m_1 \times m_2$ e $i = i_1 + i_2$. **Provado!**

A proposta original possui homomorfismo para a operação de multiplicação. Contudo é possível adaptar o esquema para transforma-lo em aditivo.

Definição (Cifração)

Dado a chave pública (p, α, β) *e uma mensagem m,*

1. *Escolha aleatoriamente* $i \in \left\{ Z^*_{p-1} - \{1\} \right\}$.
2. *Calcule* $(k_E, c) = \left(\alpha^i, \alpha^m \beta^i\right) \in Z^*_p$. k_E *é a chave efêmera.*

Definição (Cifração)

Dado a chave pública (p, α, β) *e uma mensagem* m*,*

1. *Escolha aleatoriamente* $i \in \left\{ Z^*_{p-1} - \{1\} \right\}$.
2. *Calcule* $(k_E, c) = \left(\alpha^i, \alpha^m \beta^i\right) \in Z^*_p$. k_E *é a chave efêmera.*

Definição (Decifração)

Dado a chave privada (d)*, uma cifra* c *e sua chave efêmera* k_E*,*

1. *Calcule* $\alpha^m = c.k_M^{-1} \in \mathbb{Z}^*_p$.
2. *A mensagem* m *pode ser recuperada através da solução do problema do logaritmo discreto ou através do uso de uma tabela de busca* [a].

[a] Do inglês, *lookup-table*.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal Exponencial com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Assim, existem c_1 e c_2 tais que:

$$c_1 = \alpha^{m_1}.\beta^{i_1} \mod p$$
$$c_2 = \alpha^{m_2}.\beta^{i_2} \mod p.$$

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal Exponencial com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Assim, existem c_1 e c_2 tais que:

$$\begin{aligned} c_1 &= \alpha^{m_1}.\beta^{i_1} \mod p \\ c_2 &= \alpha^{m_2}.\beta^{i_2} \mod p. \end{aligned}$$

Logo,

$$\begin{aligned} c_1 \times c_2 &= \left(\alpha^{m_1}.\beta^{i_1} \mod p\right) \times \left(\alpha^{m_2}.\beta^{i_2} \mod p\right) \\ &= \alpha^{(m_1+m_2)}.\beta^{(i_1+i_2)} \mod p \\ &= \alpha^{m}.\beta^{i} \mod p \\ &= c \end{aligned}$$

onde $m = m_1 + m_2$ e $i = i_1 + i_2$.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de ElGamal Exponencial com uma mesma chave pública (p, α, β) e com chaves efêmeras k_{E_1} e k_{E_2}. Assim, existem c_1 e c_2 tais que:

$$
\begin{aligned}
c_1 &= \alpha^{m_1}.\beta^{i_1} \mod p \\
c_2 &= \alpha^{m_2}.\beta^{i_2} \mod p.
\end{aligned}
$$

Logo,

$$
\begin{aligned}
c_1 \times c_2 &= \left(\alpha^{m_1}.\beta^{i_1} \mod p\right) \times \left(\alpha^{m_2}.\beta^{i_2} \mod p\right) \\
&= \alpha^{(m_1+m_2)}.\beta^{(i_1+i_2)} \mod p \\
&= \alpha^{m}.\beta^{i} \mod p \\
&= c
\end{aligned}
$$

onde $m = m_1 + m_2$ e $i = i_1 + i_2$. **Provado!**

- É probabilístico.
- A segurança do esquema de ElGamal contra ataques "passivos" (recuperar uma mensagem cifrada através da chave pública e da cifra, por exemplo) se baseia na dificuldade do problema de Diffie-Hellman e possui segurança contra ataques do tipo *texto claro escolhido arbitrariamente* [1].
- O esquema de cifração de Elgamal não oferece um método de assinatura de mensagens nativo.
- Por causa da sua propriedade homomórfica, as mensagens cifradas são consideradas "maleáveis".

[1] Do inglês, *chosen plaintext attack*, ou IND-CPA.

O esquema criptográfico de Goldwasser-Micali foi proposto em 1982 e é a origem da arvore de propostas de onde o esquema de Paillier surgiu. Este é baseado no problema da fatoração de inteiros.

Definição (Geração de chaves)

1. *Escolha dois primos p,q de mesma ordem e compute* $n = pq$.
2. *Escolha* $g \in \mathbb{Z}^*_{n^2}$.
3. *Definimos* (n, g) *com a chave pública e* (p, q) *como a chave privada.*

Definição (Cifração)

Dado a chave pública (n, g) *e uma mensagem* $m \in \mathbb{Z}_n$,

1. *Escolha aleatoriamente* $r \in \mathbb{Z}_n^*$.
2. *Definimos a cifra* $c = g^m.r^n \mod n^2$.

Definição (Cifração)

Dado a chave pública (n, g) *e uma mensagem* $m \in \mathbb{Z}_n$*,*

1. *Escolha aleatoriamente* $r \in \mathbb{Z}_n^*$.
2. *Definimos a cifra* $c = g^m.r^n \mod n^2$.

Definição (Decifração)

Dado a chave privada (p, q) *e uma cifra* c*,*

1. *Calcule* $m \equiv L\left(c^{\lambda(n)} \mod n^2\right)\mu \mod n$*, com* $\mu = k^{-1} \mod n$*,*
   $L(u) = \frac{u-1}{n}$*. e* $\lambda(u)$ *a função de Carmichael*
   $\lambda(n) = mmc\left[(p-1)(q-1)\right]$

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de Paillier com uma mesma chave pública (n, g) . Assim, existem c_1 e c_2 tais que:

$$c_1 = g^{m_1} r_1^n \mod n^2$$
$$c_2 = g^{m_2} r_2^n \mod n^2$$

onde $r_1, r_2 \in \mathbb{Z}_n^*$ são escolhidos aleatoriamente.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de Paillier com uma mesma chave pública (n, g) . Assim, existem c_1 e c_2 tais que:

$$c_1 = g^{m_1} r_1^n \mod n^2$$
$$c_2 = g^{m_2} r_2^n \mod n^2$$

onde $r_1, r_2 \in \mathbb{Z}_n^*$ são escolhidos aleatoriamente.
Dessa forma, temos que,

$$\begin{aligned} c_1 \times c_2 &= \left(g^{m_1} r_1^n \mod n^2\right) \times \left(g^{m_2} r_2^n \mod n^2\right) \\ &= g^{m_1+m_2} r^n \mod n^2 \\ &= g^m r^n \mod n^2 = c \end{aligned}$$

onde $r^n \equiv r_1^n r_2^n \pmod{n^2}$ e $m \equiv m_1 + m_2 \pmod{n^2}$.

Sejam m_1 e m_2 duas mensagens cifradas utilizando a cifra de Paillier com uma mesma chave pública (n, g) . Assim, existem c_1 e c_2 tais que:

$$c_1 = g^{m_1} r_1^n \mod n^2$$
$$c_2 = g^{m_2} r_2^n \mod n^2$$

onde $r_1, r_2 \in \mathbb{Z}_n^*$ são escolhidos aleatoriamente.
Dessa forma, temos que,

$$\begin{aligned} c_1 \times c_2 &= \left(g^{m_1} r_1^n \mod n^2\right) \times \left(g^{m_2} r_2^n \mod n^2\right) \\ &= g^{m_1+m_2} r^n \mod n^2 \\ &= g^m r^n \mod n^2 = c \end{aligned}$$

onde $r^n \equiv r_1^n r_2^n \pmod{n^2}$ e $m \equiv m_1 + m_2 \pmod{n^2}$.**Provado!**

- É probabilístico.
- O criptossistema de Paillier tem segurança contra ataques "passivos" (recuperar uma mensagem cifrada através da chave pública e da cifra, por exemplo) baseada no problema da fatoração de inteiros e possui segurança contra ataques do tipo *texto claro escolhido arbitrariamente* [2].
- O esquema de cifração de Paillier não oferece um método de assinatura de mensagens nativo.
- Por causa da sua propriedade homomórfica, as mensagens cifradas são consideradas "maleáveis".

[2] Do inglês, *chosen plaintext attack*, ou IND-CPA.

1. Definição
   - Criptografia Homomórfica
2. Criptossistemas
   - Paillier
   - Elgamal
3. Aplicação
   - Definição do problema
   - Implementação
     - Simulação de votação
     - Gerador de números pseudo-aleatórios
     - Testes de consistência
4. Conclusão

Problema: Em uma sociedade democrática, precisa-se de ferramentas para auxiliar a escolha de representantes políticos. O voto eletrônico é uma dessas ferramentas e se caracteriza pelo uso de uma urna eletrônica que armazena os votos dos eleitores em um banco de dados digital. Ao final do processo eleitoral, os votos são somados e se obtém o resultado.

Neste cenário, é importante que os votos sejam armazenados de forma segura. Caso contrário, um atacante que tenha acesso ao mecanismo de armazenamento poderia adquirir conhecimento sobre os votos recebidos por uma urna e inclusive altera-los valores.

Proposta: Desejamos desenvolver um sistema que possa gerar um banco de dados cifrado com a estrutura necessária para um sistema de votação, operar sobre ele **sem compartilhar a chave de decifração** e, ao final do processo, recuperar o total de votos.

Implementação

Nossa solução se baseia na implementação de rotinas relacionadas diretamente com o problema e de rotinas relacionadas com a cifra. Além disso, também precisamos de um conjunto de testes para garantir a corretude.

generate_candidate_list.py Recebe como parâmetro a cifra deverá ser usada ("elgamal" ou "paillier"). Gera como saída um arquivo "encrypted_votes.json" no formato JSON contendo a lista de candidatos com um número de referência e com a quantidade de votos que cada candidato recebeu (inicialmente 0), além dos arquivos "private.key" e "public.key" com, respectivamente, as chaves pública e privada a serem usadas nas operações.

add_votes.py Recebe o arquivo "encrypted_votes.json", a cifra a ser usada ("elgamal" ou "paillier"), a chave pública em "public.key" e uma *string* com a codificação do voto.

decrypt_votes.py Recebe o arquivo "encrypted_votes.json", um arquivo onde os resultados serão gravados (por padrão, "decrypted_votes.json"), a cifra a ser usada ("elgamal" ou "paillier"), a chave pública em "public.key" e a chave privada em "private.key".

A rotina decrypt_votes.py decifra os votos de cada candidato.
Caso a cifra usada seja **Elgamal Exponencial** ela também gera uma tabela de busca de tamanho arbitrário, que pode ser limitado superiormente pela quantidade de eleitores.

Foram implementados os criptossistemas de Paillier e ElGamal Exponencial, além do teste de primalidade de Miller-Rabin para a geração de primos.

elgamal_cipher.py Contém as funções necessárias para cifração e decifração do conteúdo de arquivos através da cifra de ElGamal. Recebe como entrada o arquivo a ser cifrado e um parâmetro “-e” ou “-d” definindo, respectivamente, que o arquivo de entrada deve ser cifrado ou decifrado.

paillier_cipher.py Contém as funções necessárias para cifração e decifração do conteúdo de arquivos através da cifra de Paillier. Recebe como entrada o arquivo a ser cifrado e um parâmetro “-e” ou “-d” definindo, respectivamente, que o arquivo de entrada deve ser cifrado ou decifrado.

generate_prime.py Contém a lógica para a geração de primos através do teste de Miller-Rabin.

- Criptossistemas probabilísticos tem em sua segurança um forte lastro na qualidade de seus geradores de números pseudo-aleatórios.
- Em nossa implementação utilizamos a biblioteca *random*, padrão da linguagem Python. Ela utiliza o algoritmo de Mersenne Twister.
- Este é um algoritmo completamente determinístico e não é indicado para implementações criptográficas reais.

É importante garantir o correto funcionamento das implementações das cifras antes de usa-las em uma aplicação.
Por isso, através da técnica de Desenvolvimento Orientado a Testes [3] desenvolvemos três algoritmos de teste básicos.

[3] Do inglês, *Test Driven Development*, ou *TDD*.

Implementação

Teste de consistência - Miller Rabin

O teste de primalidade de Miller-Rabin nos da garantia que um inteiro não é primo. Nosso teste verifica se o algoritmo implementado não fornece falsos negativos entre os inteiros no intervalo de $[2, 104.729]$.

Algorithm 1 Teste: Miller-Rabin

```
known_primes ← load_primes(10^4)
for n = 2 to higher(known_primes) do
    if  failed the miller_rabin_test(n) and n in known_primes then
        return FALSE
    end if
end for
return TRUE
```

Implementação

Teste de consistência - Elgamal e Paillier

Um criptossistema corretamente implementado deve, pelo menos, ser capaz de cifrar e decifrar uma mensagem corretamente.

Algorithm 2 Teste: Elgamal Exponencial e Paillier

```
pub, priv ← generate_keys()
for n = 0 to 10^3 do
    c, ke ← encrypt(pub, n)
    if c == n then
        return FALSE
    end if
    if decrypt(pub, priv, c, ke) ≠ n then
        return FALSE
    end if
end for
return TRUE
```

Implementação

github.com/pdroalves/cryptographic-homomorphism-demonstration

Agenda

1. Definição
   - Criptografia Homomórfica
2. Criptossistemas
   - Paillier
   - Elgamal
3. Aplicação
   - Definição do problema
   - Implementação
     - Simulação de votação
     - Gerador de números pseudo-aleatórios
     - Testes de consistência
4. Conclusão

Conclusão

- Implementamos os criptossistemas de Paillier e Elgamal Exponencial e implementamos testes que verificam sua corretude.
- Demonstramos que esses criptossistemas possuem a propriedade estudada através da prova matemática e de uma demonstração prática.

Obrigado!