الباحثون: حنين عمار زينة عبير فريد خليل

مروة عدنان أحمد هيا فيليب السليمان

المرتبة العلمية: السنة الخامسة في كلية الهمك

قسم هندسة الاتصالات والالكترونيات

العنوان

خوارزمية ديفي هيلمان لتبادل مفاتيح التشفير لتحقيقاتصال امن (مشفر) عبر شبكة غير امنة

إشراف:

الدكتور المهندس مهند عيسى

Diffie-Hellman

ملخص:

كان تبادل مفاتيح Diffie-Hellman أحـد أهم التطـورات في تشـفير المفتـاح العـام ولا يـزال يتم تنفيذه بشكل متكرر في مجموعة من بروتوكولات الأمان المختلفة اليوم.

تبادل مفاتيح (D – H) Diffie-Hellman هو بروتوكول أمـان وليس بروتوكـول تشـفير. يسـمح للطرفين بإنشاء مفتاح عبر قنـاة غـير آمنـة دون أي معلومـات مسـبقة من الطـرف الآخـر. يمكن استخدام هذا المفتاح كمفتاح متماثل في الاتصالات اللاحقة لتشفير محتوى الاتصال.

حساب DH سريع ولكن يصعب عكسه. هنـاك عـدد لا يحصـى من خوارزميـات Diffie-Hellman المماثلة، مثل RSA استنادًا إلى عامل عدد صحيح. من حيث المبدأ، DH هي واحدة من العديد من خوارزميات التشفير القائمة على اللوغاريتمات المنفصلة.

1. مقدمة

تتمثل فكرة أي نظام تشفيري في إخفاء المعلومات السرية بطريقة يصبح من خلالها معناها غيرّ مفهوم بالنسبة إلى أي شخص غير مصرح له بالاطلاع عليها. يتمثل الاستخدامانٍ لها عبر قناة الأكثر شيوعا للتشفير في تخزين البيانات بأمان في ملف كمبيوتر أو نقْلةً. في كلتا الحالتين، حقيقةٌّ المصرح لهم بالوصول إليه، ولكنها تضمن عدم تمكنهم من فهم ما يرونه. غالبًا ما يطلَق على المعلومات المراد إخفاؤها اسم «النص الأصلي»، فيما يطلق على عملية إخفائها اسم «التشفير». ويطلق على النص الأصلي المشفر اسم «النص المشفر» أو «بيان التشفير»، كما يطلق على مجموعة القواعد المستخدمة في تشفير معلومات النص الأصلي «خوارزمية التشفير». عادةً، تعتمد هذه الخوارزمية على «مفتاح التشفير»؛ وهوً يمثل مدخلا لها بالإضافة إلى الرسالة. وحتى يتمكن المتلقي من استرجاع الرسالة من خلال النص المشفر، يجب أن تتوافر «خوارزمية فك التشفير» التي، عند استخدامها مع «مفتاح فك التشفير» المناسب، تسترجع النص الأصلي من النص المشفر.

تعتمد خوارزمية تبادل المفاتيح Diffie Hellman لتوليد المفاتيح على تشفير المنحنى الإهليلجي، وهي طريقة للقيام بتشفير المفتاح العام استنادا إلى بنية الجبر للمنحنيات الإهليلجية على الحقول المحدودة. يستخدم DH أيضا وظيفة trapdoor، تماما مثل العديد من الطرق الأخرى للقيام بتشفير المفتاح العام. الفكرة البسيطة لفهم خوارزمية DH هي التالية.

1. يختار الطرف الأول عددين أوليين هما g و p ويخبرهما للطرف الثاني.
2. ثم يختار الطرف الثاني رقما سريا نسميه a ، ثم يحسب $g^a \bmod p$.

   ويرسل النتيجة مرة أخرى إلى الطرف الأول؛ دعونا نسمي النتيجة A. ضع في اعتبارك أن الرقم السري لا يتم إرساله إلى أي شخص، بل النتيجة فقط.

3. ثم يفعل الطرف الأول الشيء نفسه؛ يختار رقما سريا b ويحسب النتيجة.
4. الخطوة 2. ثم يتم إرسال هذه النتيجة إلى الطرف الثاني.
5. يأخذ الطرف الثاني الرقم B المستلم ويحسب B^a
6. يأخذ الطرف الأول الرقم المستلم A ويحسب A^b.

الإجابة في الخطوة 5 هي نفس الإجابة في الخطوة 4. هذا يعني أن كلا الطرفين سيحصلان على نفس الإجابة بغض النظر عن ترتيب الأس.

2. مثال تطبيقي:

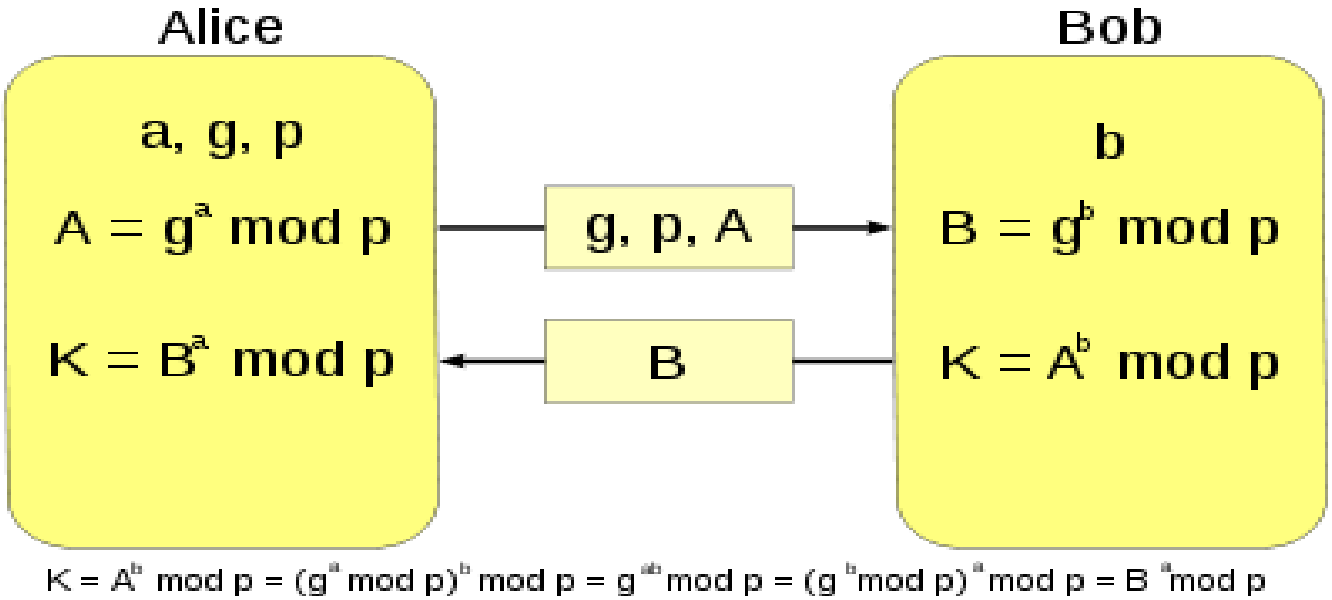
يستخدم التطبيق الأبسط والأصلي من البروتوكول مجموعة مضاعفة من معاملات الاعداد الصحيحة p، حيث يكون p أولياً، و g هو معامل الجذر الأولي p . يتم اختيار هاتين القيمتين بهذه الطريقة لضمان أن السر المشترك الناتج يمكن أن يأخذ أي قيمة من 1 إلى $p–1$. فيما يلي مثال على البروتوكول.

يوافق أليس وبوب علناً على استخدام المعامل $p = 23$ و $g = 5$.

1. تختار أليس عدداً صحيحاً سرياً $\boldsymbol{a} = 4$, وترسل لبوب $A = g^{a} \bmod p$
   - $A = 5^{4} \bmod 23 = 4$
2. يختار بوب عدداً صحيحاً سرياً $\boldsymbol{b} = 3$, ويرسل لأليس $B = g^{b} \bmod p$
   - $B = 5^{3} \bmod 23 = 10$
3. تقوم أليس بحساب $\boldsymbol{s} = B^{a} \bmod p$
   - $\boldsymbol{s} = 10^{4} \bmod 23 = 18$
4. يقوم بوب بحساب $\boldsymbol{s} = A^{b} \bmod p$
   - $\boldsymbol{s} = 4^{3} \bmod 23 = 18$
5. يتشارك أليس وبوب الآن سراً (الرقم 18).

وصل كل من أليس وبوب إلى نفس القيم لأنه في ظل mod p،

ونلخص ما سبق بالشكل التالي:



3. استخدامات تبادل مفاتيح Diffie-Hellman

الغرض الرئيسي من تبادل مفاتيح Diffie-Hellman هو تمويرِ أسرار مشتركة بشكل آمن يمكن استخدامها لاشتقاق المفاتيح. يمكن بعد ذلك استخدام هذه المفاتيح مع خوارزميات المفاتيح المتماثلة لنقل المعلومات بطريقة محمية. تميل الخوارزميات المتماثلة إلى استخدامها لتشفير الجزء الأكبر من البيانات لأنها أكثر كفاءة من خوارزميات المفتاح العام.

من الناحية الفنية، يمكن استخدام تبادل مفاتيح Diffie-Hellman لإنشاء مفاتيح عامة وخاصة. ومع ذلك، في الممارسة العملية، يميل RSA إلى استخدامه بدلا من ذلك. وذلك لأن خوارزمية RSA قادرة أيضا على توقيع شهادات المفتاح العام، في حين أن تبادل مفاتيح Diffie-Hellman ليس كذلك.

تعتمد خوارزمية ElGamal، التي تم استخدامها بكثافة في PGP، على تبادل مفاتيح Diffie-Hellman، لذا فإن أي بروتوكول يستخدمه ينفذ بشكل فعال نوعا من Diffie-Hellman.

باعتبارها واحدة من أكثر الطرق شيوعا لتوزيع المفاتيح بأمان، يتم تنفيذ تبادل مفاتيح Diffie-Hellman بشكل متكرر في بروتوكولات الأمان مثل TLS وIPsec وSSH وPGP وغيرها الكثير. وهذا يجعلها جزءا لا يتجزأ من اتصالاتنا الآمنة.

كجزء من هذه البروتوكولات، غالبا ما يتم استخدام تبادل مفاتيح Diffie-Hellman للمساعدة في تأمين اتصالك بموقع ويب، والوصول عن بعد إلى جهاز كمبيوتر آخر، وإرسال رسائل بريد.

استخدامات خوارزمية ديفي هيلمان:

بصرف النظر عن استخدام الخوارزمية لإنشاء مفاتيح عامة، هناك بعض الأماكن الأخرى التي يمكن فيها استخدام خوارزمية DH:

- التشفير: يمكن استخدام خوارزمية تبادل المفاتيح Diffie Hellman للتشفير. واحدة من المخططات الأولى التي يجب القيام بها هي تشفير الجمل. أحد الأمثلة الحديثة على ذلك يسمى نظام التشفير المتكامل، والذي يوفر الأمان ضد النص العادي المختار وهجمات الحافظة المختارة.

- اتفاقية مصادق عليها بكلمة المرور: عندما يتشارك طرفـان كلمـة مـرور، يمكن اسـتخدام اتفاقية مفتاح مصادق عليها بكلمة مرور لمنـع هجـوم الرجـل في الوسط. يمكن أن يكـون هذا الاتفاق الرئيسي في شكل Diffie-Hellman. يعد بروتوكول كلمة المـرور الآمنـة عن بعد مثالا جيدا يعتمد على هذه التقنية.
- السرية الأمامية: يمكن للبروتوكولات المستندة إلى السرية إلى الأمام إنشاء أزواج مفاتيح جديدة لكل جلسة عمل جديدة، ويمكنها تجاهلها تلقائيا عند انتهاء الجلسة. في بروتوكولات السرية الأمامية هذه، في أكثر الأحيان، يتم استخدام تبادل مفاتيح Diffie Hellman..

4. مزايا خوارزمية ديفي هيلمان:

1. لا يحتاج المرسل والمستقبل إلى أي معرفة مسبقة ببعضهما البعض.
2. بمجرد تبادل المفاتيح، يمكن توصيل البيانات من خلال قناة غير آمنة.
3. مشاركة المفتاح السري آمنة.

5. عيوب خوارزمية Diffie Hellman

1. لا يمكن تبادل مفاتيح غير متماثل.
2. لا يمكن استخدامه لتوقيع التوقيعات الرقمية.

القسم العملي

سنقوم ببناء تطبيق TCP Client/Server يقوم بتطبيق خوارزمية ديفي هلمان لتبادل المفاتيح بشكل آمن قبل التبادل الفعلي للبيانات. وبعد تبادل المفاتيح يتم استخدام التشفير المتناظر في تبادل البيانات الفعلية.

يتم الاتفاق على G, P مسبقاً وتخزن بكلا تطبيقي العميل والسيرفر. ويتم توليد قيم a و b عشوائياً (واحدة في كل طرف) وتطبيق الخوارزمية المعروفة للحصول على المفتاح المشترك k.

ولكن لاستخدام المفتاح في التشفير المتناظر (تتطلب الخوارزمية المستخدمة مفتاح بطول محدد) نستخدم خوارزمية أخرى تقوم بتوليد مفتاح انطلاقاً من قيمتين هما password سرية متفق عليها بين الطرفين (k في مثالنا) و salt قيمة عشوائية يتم توليدها بأحد الأطراف (السيرفر في مثالنا) وإرسالها للطرف الآخر عند كل عملية اتصال. عندها نحصل على مفتاح key يتم استخدامه في التشفير وفك التشفير وذلك باستخدام الحزمة Fernet.

كود السيرفر:

```
import random
import string
from socket import *
from threading import Thread
from generate_key import generate_key
from cryptography.fernet import Fernet

server = socket(AF_INET,SOCK_STREAM)

server.bind(("127.0.0.1", 2222))

server.listen()

print('Starting server..')

def worker(client):
    G = 9
    P = 23
```

```
    a = random.randint(10,100)
    print("a=", a)
    x = int(pow(G, a, P))
    print("x=", x)
    salt = ''.join(random.choices(string.ascii_uppercase
+ string.digits, k=10)).encode()
    print("salt=", salt)
    client.send(str(x).encode())
    client.send(salt)

    y = int(client.recv(1024).decode())
    print("y=", y)

    k = int(pow(y, a, P))
    print("k=", k)
    password = str(k).encode()

    key = generate_key(password, salt)
    print("key=", key)
    fernet = Fernet(key)

    while True:
        enc_message = client.recv(1024)
        message = fernet.decrypt(enc_message).decode()
        print("<< {} [{}]".format(message,
enc_message.decode()))
        message=message[::-1]
        enc_message = fernet.encrypt(message.encode())
        print(">> {} [{}]".format(message,
enc_message.decode()))
        client.send(enc_message)

while True:
    client, remote_address = server.accept()
```

```
    print('Receive connection from address: ',
remote_address)

    thread = Thread(target=worker, args=(client,))
    thread.start()
```

كود العميل:

```
import random
from socket import *
from generate_key import generate_key
from cryptography.fernet import Fernet

client = socket(AF_INET,SOCK_STREAM)

client.connect(("127.0.0.1", 2222))

while True:
    G = 9
    P = 23

    b = random.randint(10,100)
    print("b=", b)
    y = int(pow(G, b, P))
    print("y=", y)
    x = int(client.recv(1024).decode())
    print("x=", x)
    salt = client.recv(1024)
    print("salt=", salt)

    client.send(str(y).encode())
    k = int(pow(x, b, P))
    print("k=", k)
    password = str(k).encode()
```

```
    key = generate_key(password, salt)
    print("key=", key)
    fernet = Fernet(key)

    while True:
        message = input("Enter a message: ")
        enc_message = fernet.encrypt(message.encode())
        print(">> {} [{}]".format(message,
enc_message.decode()))
        client.send(enc_message)

        r_enc_message = client.recv(1024)
        r_message =
fernet.decrypt(r_enc_message).decode()
        print("<< {} [{}]".format(r_message,
r_enc_message.decode()))
```

كود التابع generate_key:

```
import base64
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import
PBKDF2HMAC

def generate_key(password, salt):
    kdf = PBKDF2HMAC(algorithm=hashes.SHA256(),
length=32, salt=salt, iterations=390000)
    key = base64.urlsafe_b64encode(kdf.derive(password))
    return key
```

اختبار التطبيق:

السيناريو المطبق: يتم تشغيل السيرفر ثم تشغيل العميل عندها تلقائياً يتم تبادل المفاتيح (يتم طباعة تفاصيل العملية للتوضيح).

يدخل العميل رسالة ثم يتم تشفيرها وإرسالها إلى السيرفر ثم يستقبل الرد من السيرفر ويفك تشفير الرسالة وإظهارها.

يقوم السيرفر باستقبال الرسالة وفك تشفيرها ثم يقوم بعكس الرسالة وإعادة تشفيرها وإرسالها للعميل.

6.المراجع:

[1] https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/

[2] https://djangostars.com/blog/how-to-create-and-deploy-a-telegram-bot/

[3] Aryan, Chaithanya Kumar "Enhanced diffie-hellman algorithm for reliable key exchange" IOP Conference Series: Materials Science and Engineering ,2017.

[4] https://mathstats.uncg.edu/sites/pauli/112/HTML/secdiffiehellman.html