CS101 PROJECT REPORT

***Project title: Understanding Diffie Hellman key exchange algorithm and finding potential loopholes*.**

NAME	ENTRY NO
TEJAS WAGH	2022CSB1144
PRANAV BHOLE	2022CSB1103
SHIVAM ZAMPLE	2022MCB1280
WALDE VISMAY	2022MCB1283

CS101 PROJECT REPORT

Project title: Understanding Diffie Hellman key exchange algorithm and finding potential loopholes.

S no	project subparts
1	Introduction to Diffie Hellman algorithm
2	Implementation: Analogous example
3	Implementation: Mathematical aspects
4	Discrete logarithm problem: Backbone of DH algorithm
5*	Potential loopholes: vulnerabilities
6	Applications of DH algorithm
7	Observations
8	Conclusions

Note: Abbreviations used [DH – Diffie Hellman]

Introduction to Diffie Hellman algorithm

DH algorithm belongs to the branch of cryptography known as Asymmetric key cryptography .but what is it?
Lets understand it first

Asymmetric Key Cryptography: In this type we use a pair of keys with specific function either to lock or to unlock. A receiver's public key is used for encryption and a receiver's private key is used for decryption. Both the keys are different. In this method only the intended person can see the message as it can be decrypted by private key only.

In this branch of cryptography key exchange is the primary step and to make the key exchange secure, comes in the DH algorithm

Note: Historical aspects of algorithm are intentionally omitted to make report concise and to the point

Definition: It is the method of exchanging the keys over an insecure communication channel using the mathematical properties of modular exponentiation and the computational difficulty of solving discrete logarithm problem.

.

Implementation: Analogous example (source – Wikipedia)

- The process begins by having the two parties, Alice and Bob, publicly agree on an arbitrary starting color that does not need to be kept secret. In this example, the color is yellow. Each person also selects a secret color that they keep to themselves – in this case, red and cyan. The crucial part of the process is that Alice and Bob each mix their own secret color together with their mutually shared color, resulting in orange-tan and light-blue mixtures respectively, and then publicly exchange the two mixed colors. Finally, each of them mixes the color they received from the partner with their own private color. The result is a final color mixture (yellow-brown in this case) that is identical to their partner's final color mixture.

- If a third party listened to the exchange, they would only know the common color (yellow) and the first mixed colors (orange-tan and light-blue), but it would be very hard for them to find out the final secret color (yellow-brown). Bringing the analogy back to a real-life exchange using large numbers rather than colors, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern supercomputers.

Note: It is the best example to understand analogy of DH algorithm so we took help of wikipedia here

What did we observe from this example ?

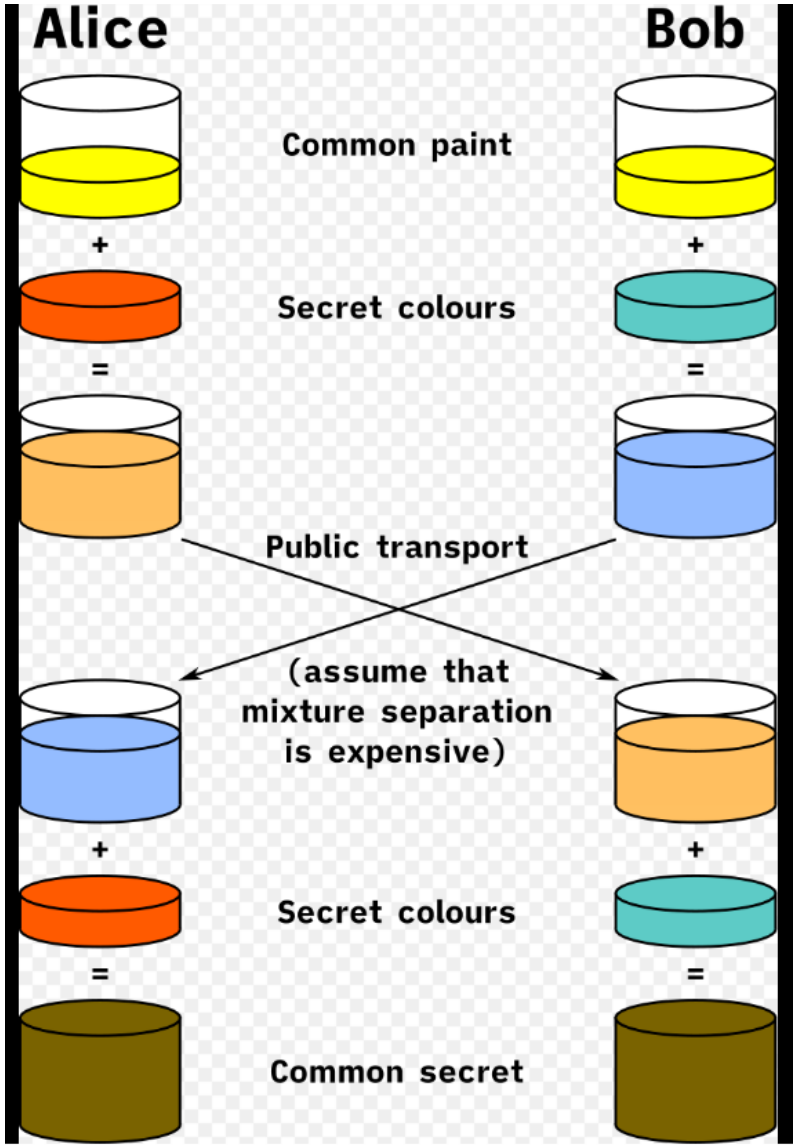
Our observations:

1) 2 things were publically declared And 2 things were kept secret

2) the 2nd PKC was obtained using 1st PKC

3) Once ones private colour was mixed with others public we got shared secret and which is known to both the parties but not the middleman

4*) Once we mixed the secret colour with the 1st PKC it is very difficult to determine what was the secret colour and that's what is the backbone of Diffie Hellman algorithm.



Source :Wikipedia

Note: PKC stands for 'publically known colour'

- From the previous slide we observed that there are some actions which once performed in one direction cannot be reversed back!

(give glass breaking example in video explanation)

and we take its benefit in cryptography !

This was an example of one way function,

Wait,…. But what is it ?

One way function: It is simply the function which is easy to compute for every value in domain but it is hard to get back from range to domain i.e it is hard to get the input from the given the output

And here hard and easy are in the sense of computational complexity theory.

We observed that DH algorithm also relies on one way function popularly known as the discrete logarithm problem.

Discrete logarithm problem: Backbone of DH algorithm

- Given a finite cyclic group G, a generator g of the group, and an element h in the group, the discrete logarithm problem involves finding an integer x such that g^x ≡ h (mod modulus), where g^x denotes the modular exponentiation of g to the power of x.

- In other words, given g, h, and modulus, we need to find the exponent x such that raising g to the power of x and taking the remainder modulo modulus gives us h.

- The difficulty of the discrete logarithm problem lies in the fact that there is no known efficient algorithm to solve it for all cases.

- **And that's the key for DH algorithm**
- If something is there which is computationally difficult cannot be solved using brute force in less time
- And that's what DH algorithm uses…

Implementation :Mathematical aspects

- Lets see how exactly we are going to use discrete log problem in DH algorithm
- The algorithm is as follows :-

a)Key Generation: Both parties, let's call them Alice and Bob, agree on a large prime number, "p," and a primitive root modulo "p," denoted as "g.“

b)Public Key Exchange: Alice and Bob independently generate their private keys, "a" and "b," respectively. They then compute their public keys as follows:

- Alice: Alice calculates her public key, "A," by computing A = g^a mod p.
- Bob: Bob calculates his public key, "B," by computing B = g^b mod p.
- Secret Key Calculation: Alice and Bob exchange their public keys with each other.

c)Shared Secret Key: Alice and Bob each independently calculate the shared secret key using the other party's public key and their own private key:

- Alice: Alice computes the shared secret key as S = B^a mod p.
- Bob: Bob computes the shared secret key as S = A^b mod p.

d)Both Alice and Bob now have the same shared secret key, which can be used for symmetric encryption to secure their communication.

- What we did : random prime number generation code
-

```python
import random

def prime_numbers():
    j=0
    while j<100:
        odd_number = generate_odd_number()
        i=0
        while i<=10000:
            (variable) flag: Literal[0]
                                    ber-1)
            if pow(a,odd_number-1,odd_number) !=1:    # primality test
                break
            flag=1
            i=i+1
        if flag==1:
            #print(odd_number)
            return odd_number


        j=j+1

prime1=prime_numbers()
print(prime1)
```

✓ 1.1s

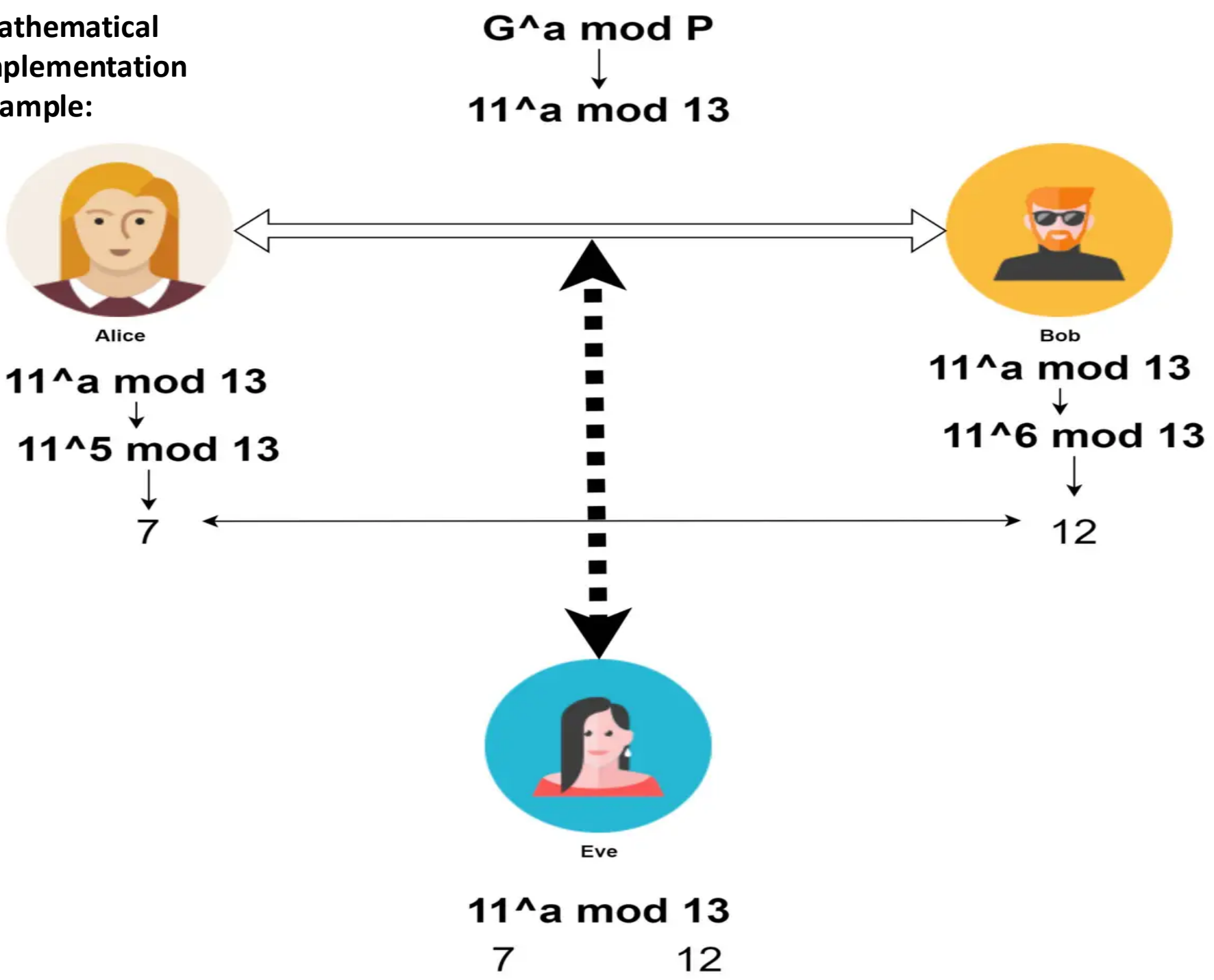
Python

```
1616216095655779561294301738797021874641711220493258890586255448490070862922803 3
```

the above code generates a 80 digit prime number using primality test. as the probability of finding a prime is roughly 1/logn it tries. the prime number can be expected in logn turns

Markdown

Mathematical implementation example:



- We too implemented and verified whether the time required is feasible or not:

now as any one lets try to brute force the key lets see if we can

Markdown

We found our code lagging and we interrupted it

```
print(common1)
count=0
while count<=common1:
    count=count+1
print("found")
```

⊗ 6m 31.5s

Python

```
41564316165723610385023446927698423769422097779484194935722276197656043185217924
```

```
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[9], line 3
      1 print(common1)
      2 count=0
----> 3 while count<=common1:
      4     count=count+1

KeyboardInterrupt:
```

as you can see the time to brute force is massive and its just a 80 digit prime number imagine using a 600 digit prime number. this is the security!!!!!!! the size of the number grows exponentially thus impossible

Markdown

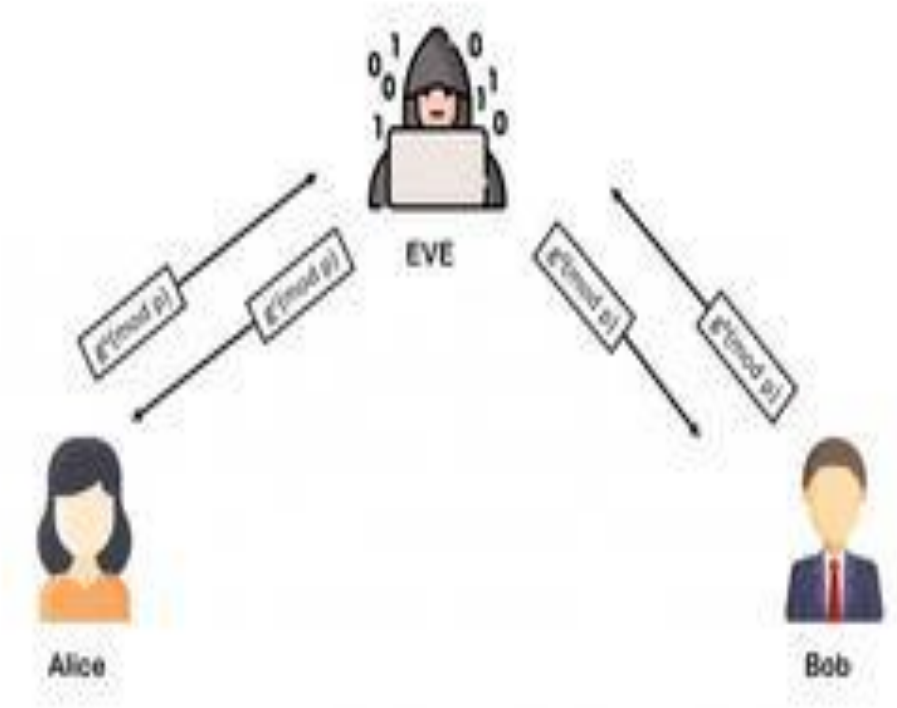
Man in the middle attack:

The basis of this attack is the step when two parties Alice and Bob exchange their public keys. There is lack of trust factor that the private key I receive is from the right person and not someone else.

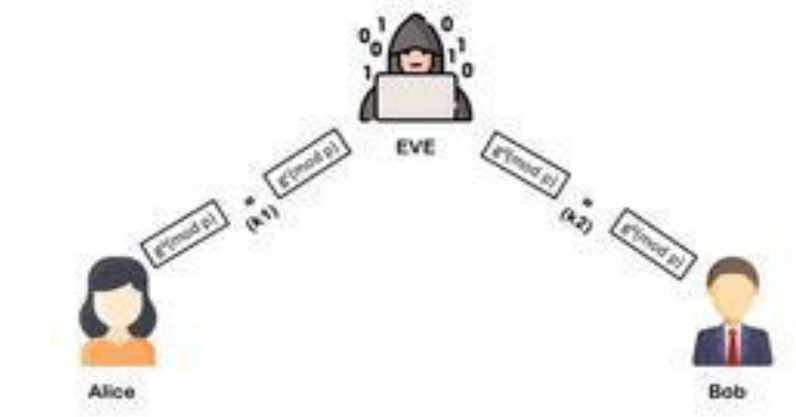
Man in the Middle (MITM) against Diffie-Hellman:
A malicious Malory, that has a MitM (man in the middle) position, can manipulate the communications between Alice and Bob, and break the security of the key exchange.

Step by Step explanation of this process:
Step 1: Selected public numbers p and g, p is a prime number, called the “modulus” and g is called the base.

- **Step 2:** Selecting private numbers.
- let Alice pick a private random number a and let Bob pick a private random number b, Malory picks 2 random numbers c and d.
- Malory intercepts Alice's public value (g^a(mod p)), block it from reaching Bob, and instead sends Bob her own public value (g^c(modp)) and Malory intercepts Bob's public value (g^b(mod p)), block it from reaching Alice, and instead sends Alice her own public value (g^d (modp))



- **Step 4:** Computing secret key
- Alice will compute a key $S_1=g^{da}(\text{mod } p)$, and Bob will compute a different key, $S_2=g^{cb}(\text{mod } p)$



- **Step 5:** If Alice uses S_1 as a key to encrypt a later message to Bob, Malory can decrypt it, re-encrypt it using S_2, and send it to Bob. Bob and Alice won't notice any problem and may assume their communication is encrypted, but in reality, Malory can decrypt, read, modify, and then re-encrypt all their conversations.

- The possible solution to this attack is introducing the authentication protocol.

A third party certifying authority can be appointed, confirming that the received key is from the authorised sender using some advanced techniques such as digital signature

- In the proposed system design, proposed Variable Round Hash (VRH) function is added to verify whether the public key has not been altered by an attacker. In the system, a private key is chosen and the public key is calculated by using prime (p), generator (g) and private key like the same process of original DHKE algorithm. Before exchanging the public key to each other, the hash codes of public keys in both sites are generated with the help of proposed VRH function and interchanged with each other. In the verification process, the hash code is produced from receiving public key again in both sites. If both receiving hash code and calculating hash code is matched, it is ensured that the key has not been changed. Therefore, the secret key is calculated using this public key. Otherwise, the receiving public key is rejected and the key exchange process is repeated. The design of the proposed system is shown in Figure 1.

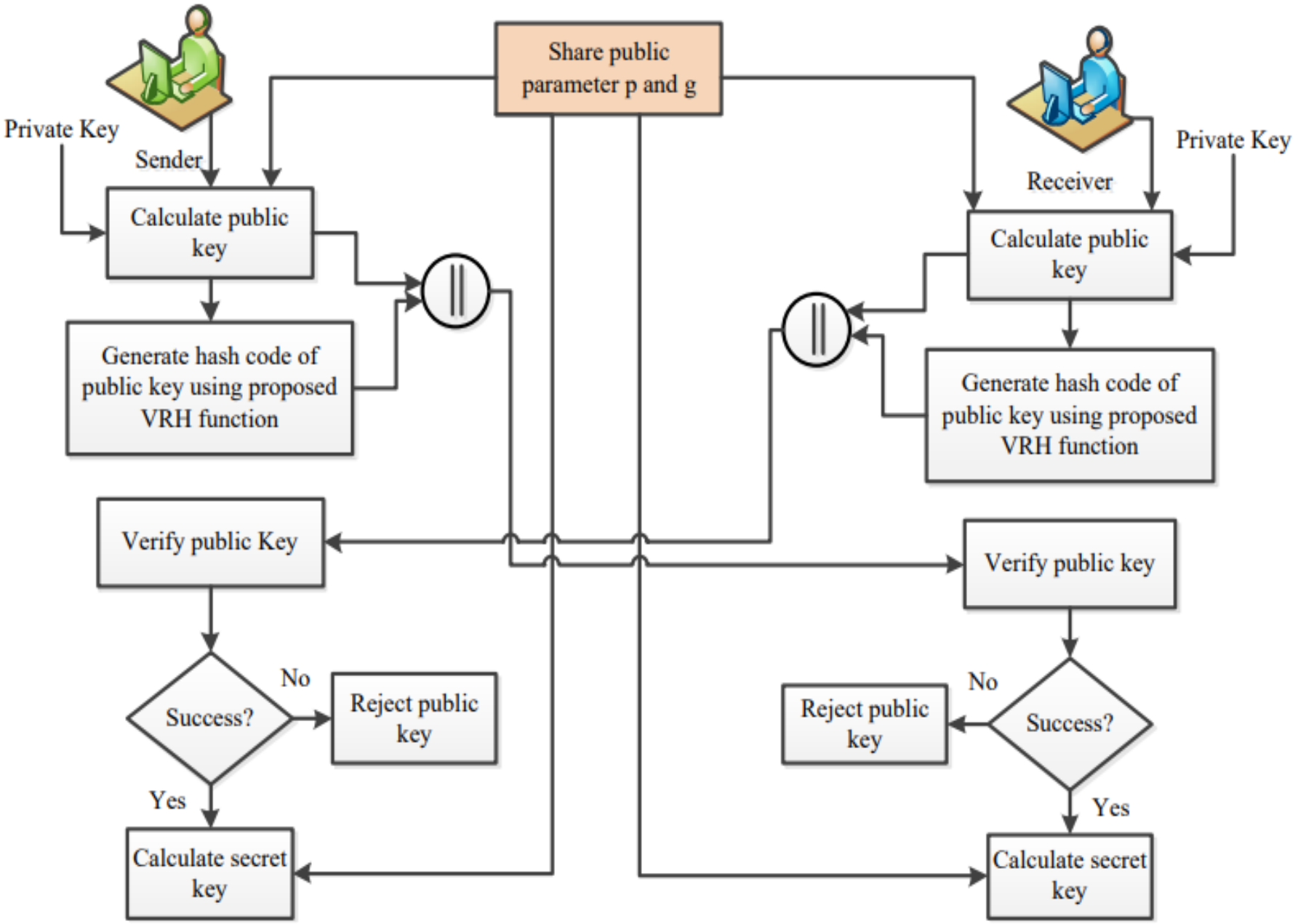


Figure1. Design of the proposed system

begins with the standard protocol of generating p and g and the repective public keys but the exchange of public keys gets interrupted

```
m= random.randint(1, prime1-2)
#false key
falsekey=pow(g,m,prime1)
#he exchanges this key with alice and bob
alicekey=pow(falsekey,private1,prime1)
bobkey=pow(falsekey,private2,prime1)
#thus hacker has
hack1=pow(r1,m,prime1)
hack2=pow(r2,m,prime1)
print(hack1,alicekey)
print(hack2,bobkey)
```

✓ 0.0s

```
29823434906657668069655829894810077727883246697543340011217202364527546155694255 29823434906657668069655829894810077727883246697543340011217202364527546155694255
71435922639362701969058843889889159878452442803855322185247480138553471823214310 71435922639362701969058843889889159878452442803855322185247480138553471823214310
```

- precomputation attack also known as dictionary attack:

- The basis of this attack is the public nature of the prime number and the generator number. In this attack the attacker calculates all the values of a for the given remainder by (g raise to a mod n) as the remainder is periodic it is quite intuitive that when he tries all the values of a there would be more probability of hitting the answer than simple bruteforcing as the attacker can skip n-2 numbers at one time.
- To prevent as the entire calcualtions are yet dependent on the p and g just change them. Or take very large values of a and b which are just unreachable for the hacker. Changing the values of p and q is a more optimum solution as the attacker would have to redo the entire calculation which is not time feasible
- Reference taken from:

International Journal of
Advances in Scientific Research and Engineering (ijasre)

DOI: 10.31695/IJASRE.2019.33560

October - 2019

Prevention of Man-In-The-Middle Attack in Diffie-Hellman Key Exchange Algorithm using Proposed Hash Function

Phyu Phyu Thwe[1] and May Htet[2]

Research Scholar [1] and Associate Professor[2]

[1-2]Department of Computer Engineering and Information Technology,

[1-2]Mandalay Technological University, Mandalay

Myanmar

We also verified precomputation attack:

```
prime=23
proot=5
private_key=2716053
#secret key
l=[]
i=0
while i<=prime-2:
    l.append(pow(proot,i,prime))
    i=i+1
# given public key
remain= 14  #remainder
#challenge to find the a
i=0
while i<=prime-2:
    if l[i]==remain:
        break
    i=i+1
print(i)
#periodicity
p=1
test=21+(22*p)
while test!= private_key:
    p=p+1
    test=21+(22*p)
print(test)
```

✓ 0.0s

```
21
2716053
```

Precomputation attack:

- In a precomputation attack, an attacker performs extensive calculations before the actual key exchange takes place. The attacker precomputes values based on known parameters and stores them for later use. These precomputed values can significantly reduce the computational effort required during the key exchange phase.
- By carefully selecting parameters and performing precomputation, an attacker can derive information that allows them to break the security of the Diffie-Hellman key exchange more efficiently. They can leverage these precomputed values to quickly compute the shared secret key or discover the private key of one of the communicating parties.
- To mitigate precomputation attacks, various countermeasures can be employed. One common technique is to use ephemeral (temporary) keys in the Diffie-Hellman protocol, which are generated for each session. This prevents the reuse of precomputed values across different key exchanges, making it harder for attackers to exploit their precomputed data.
- Additionally, using larger key sizes and employing elliptic curve-based Diffie-Hellman (ECDH) variants can enhance the security against precomputation attacks by increasing the computational complexity required to perform the precomputations and breaking the underlying mathematical problem.
- It's worth noting that the Diffie-Hellman protocol itself is not inherently vulnerable to precomputation attacks, but the implementation and usage of the protocol can introduce potential vulnerabilities if not properly secured.

- "In the context of cryptographic protocols like Diffie-Hellman, a subgroup attack is a type of attack that takes advantage of the properties of the underlying mathematical group used in the protocol to compromise the security of the key exchange.

- The Diffie-Hellman key exchange relies on the computational difficulty of computing discrete logarithms in a specific mathematical group. This group is typically a finite field or an elliptic curve group. The security of the protocol depends on the properties and size of the group's elements.

- A subgroup attack occurs when an attacker manipulates the group parameters, such as the generator or the prime modulus, to force the key exchange to occur within a small and insecure subgroup of the larger group. This smaller subgroup has fewer elements, making it easier to compute discrete logarithms within that subgroup.

- By selecting a small subgroup, an attacker can significantly reduce the computational effort required to break the protocol's security. They can compute the discrete logarithm within the small subgroup and derive the secret key used for encryption or authentication.

- To mitigate subgroup attacks, it is crucial to use well-selected and validated group parameters. These parameters should ensure that the protocol operates within a large and secure subgroup with a sufficiently large prime modulus or a strong elliptic curve group. Careful implementation and validation of these parameters help prevent the possibility of subgroup attacks.

- Regularly reviewing and updating cryptographic standards and recommendations is essential to ensure that the group parameters used in protocols like Diffie-Hellman remain secure against potential attacks, including subgroup attacks.“

A**pplications:**

- The Diffie-Hellman (DH) algorithm finds applications in diverse areas of cryptography. Its usage spans various domains, ensuring secure communication, key exchange, and other cryptographic operations.
- The algorithm enables the establishment of shared secret keys without the need for pre-shared secrets, providing confidentiality in exchanged messages. It is employed in protocols like TLS and SSH to establish secure communication channels.
- DH can also be used for creating digital signatures, ensuring message integrity and non-repudiation. Furthermore, it supports password-based key derivation, enabling encryption key generation from passwords. The algorithm extends to group key establishment for secure group communication or multicast encryption. Additionally, DH aids in anonymous communication, allowing for secure and private connections. Overall, the Diffie-Hellman algorithm serves as a versatile and essential tool in cryptography.

Observations:

1)We observe that taking base as primitive root modulo p(prime modulus) increases complexity, if not taken may face subgroup attack.

2)we observe that as Diffie Hellman protocol is based on discrete logarithm problem , which is computationally difficult , applying brute force would lead to exhaustive search through the entire keyspace.

3)It does not provide authentication to either of the parties. Without authentication, the Diffie-Hellman protocol is vulnerable to various attacks, such as man-in-the-middle attacks as discussed earlier.(solution- use ephemeral keys).

4) Prime number p should be very large in order to make it computationally difficult.

RISE OF RSA:

While the Diffie-Hellman protocol enables secure key exchange, it does not inherently provide encryption or digital signature capabilities, The computational complexity of Diffie Hellman is much more , as it involves multiple modular exponentiations. Taking a large prime number would make it more computationally expensive.

This lead to rise of RSA algorithm, which is versatile and supports both encryption and digital signatures.
It primarily relies on modular exponentiation with the RSA public and private keys, which can be more efficient.

It's important to note that the rise of RSA does not diminish the significance or security of the Diffie-Hellman protocol. Both algorithms have their specific use cases and strengths. In fact, modern cryptographic protocols often combine the strengths of different algorithms to provide secure and efficient solutions, leveraging Diffie-Hellman for key exchange and RSA for encryption and authentication.

- **Conclusion:**

Diffie-Hellman works on the principles one way function , and it comes under asymmetric key cryptography, where each party assumes a private key and declares a public key. The public keys can be openly shared, while the private keys remain secret. This enables secure communication without the need for a pre-shared secret key.

The Diffie-Hellman key exchange algorithm allows two parties to establish a shared secret key over an insecure channel, providing secure communication and perfect forward secrecy. However, it is important to consider additional security measures to protect against potential attacks and vulnerabilities associated with the algorithm.

The Diffie Hellman key exchange is a one way function, as we can get public key from private key , but getting private key(which we don't know) back from public key is not impossible but rather difficult.

So we can conclude ignorance is the key, whatever we humans can't solve will be a part of cryptography.