DIFFIE-HELLMAN KEY EXCHANGE VIA REST

PETRO KOLOSOV

Abstract. Discussion on Diffie-Hellman Key Exchange and its implementation via REST.

Contents

1. Introduction

Diffie–Hellman (DH) protocol is a method of asymmetric exchange of the cryptographic keys for a group of two or more participants, developed in 1976 by cryptographers Ralph Merkle, Whitfield Diffie and Martin Hellman. In contrast to the symmetric key exchange, the Diffie–Hellman protocol eliminates the direct transfer of the shared secret between the participants so that each participant computes a shared secret with his own private-public key pair. The Diffie–Hellman protocol is based on a one-way function of the form

$$A = G^a \bmod P \tag{1.1}$$

where A is the user's public key, a is the user's private key, $P = 2Q + 1$ is modulus, such that 2048 bits safe-prime because Q is also prime, G is generator such that G is primitive root modulo P. We say that G is primitive root modulo P if for each $1 \leq a \leq P - 1$ the $A = G^a \bmod P$ is unique and belong to the set $\{1, 2, \ldots, P - 1\}$. The period of such cyclic group $\mathbb{Z}_P$ is $P - 1$ then.

Thus, the safety of the Diffie–Hellman protocol is based on the **discrete logarithm problem** which is unsolvable in polynomial time if the constants G and P are chosen correctly. Graphically the flow of the Diffie–Hellman protocol can be expressed through the analogy with mixing paints as the picture below shows

Date: March 16, 2022.

2010 *Mathematics Subject Classification.* 26E70, 05A30.

Key words and phrases. Diffie-Hellman Key Exchange, DH key exchange, REST .

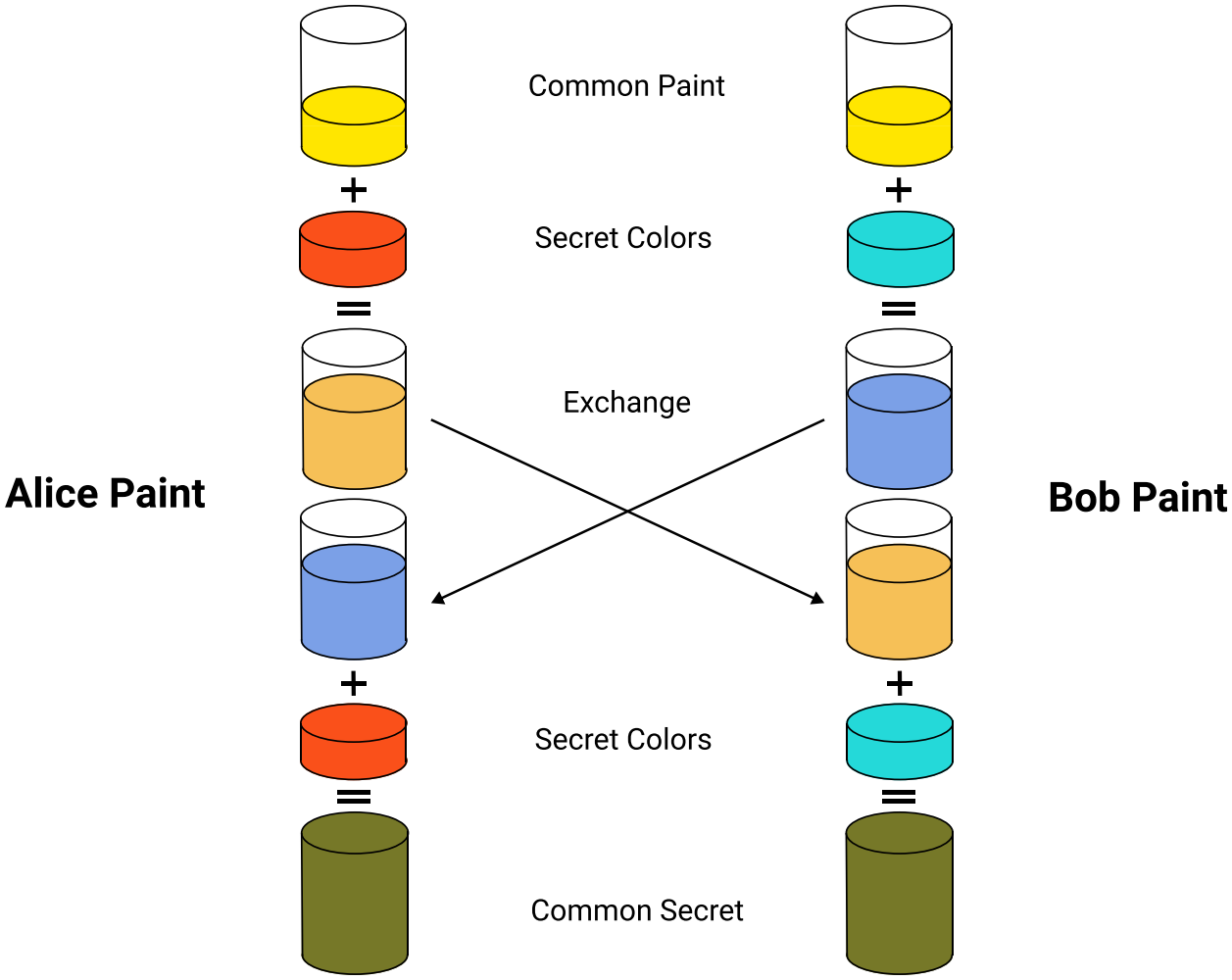

Figure 1. Diffie–Hellman key exchange concept diagram.

In contrast to the Diffie–Hellman based on discrete logarithm problem, there is an Elliptic Curve Diffie–Hellman key exchange which is based on the elliptic curve discrete logarithm problem. Although, the idea is quite similar, the difference only in that Elliptic Curve Diffie–Hellman ensures the same safety as discrete logarithm Diffie–Hellman with lower number of bits of the prime modulus P. For instance, 521 bit modulus used in Elliptic Curve Diffie–Hellman is equally safe as 2048 bit modulus in discrete logarithm Diffie–Hellman. To summarize, the flow of Diffie–Hellman key exchange is as follows. Given 2048 bits public prime modulus P and generator G such that G is primitive root modulo P then

(1) Alice chooses her secret a.
(2) Alice sends to Bob her public key $A = G^a \bmod P$.
(3) Bob chooses his secret b.
(4) Bob sends to Alice his public key $B = G^b \bmod P$.
(5) Alice computes common secret $s = B^a \bmod P$.

(6) Bob computes common secret $s = A^b \bmod P$.
(7) Alice and Bob have arrived to the same value

$$s = A^b \bmod P = G^{ab} \bmod P \tag{1.2}$$

$$s = B^a \bmod P = G^{ba} \bmod P \tag{1.3}$$

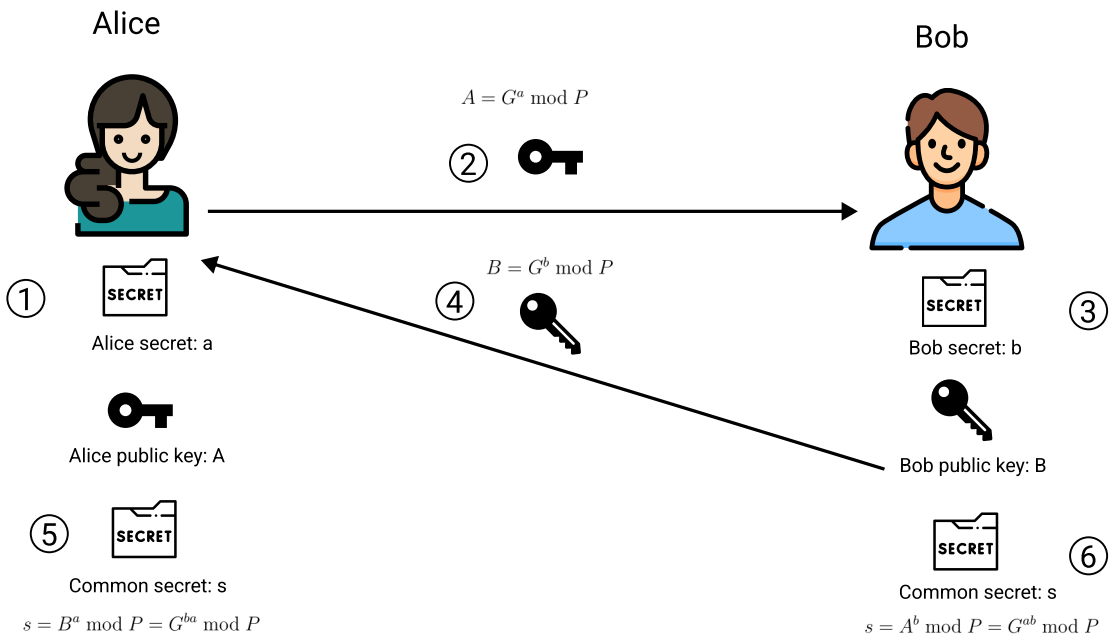

Figure 2. Diffie–Hellman key exchange concept diagram.

2. Diffie–Hellman key exchange implementation via REST

Although, the idea of Diffie–Hellman key exchange looks quite simple, some remarks on the concrete implementation should be added. Firstly, it is necessary to implement the mechanism of key exchange request between two or more parties. As it discussed previously, each user has his own private-public key pair so that in order to perform request between parties it should be implemented dedicated REST web–service endpoint. For instance, the `POST: api/key-exchange-requests` which takes the request body of the form

```
{
    "requestedUserId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "publicKey": "RUNLMSAAAAC2lkqYcTGhutQPxcjvoqUELKoy0"
}
```

So that request sender generates a private-public key pair, keeps the private key in the file system and shares the public in request to receiver. Therefore, the second party has received the key exchange request. In order to display all the key exchange requests awaiting

the confirmation of decline decisions, it is worth to implement another REST endpoint such that `GET: api/key-exchange-requests`, so that requested party will have the list of requests to proceed. This endpoint may return the data structure like follows


```
{
    "keyExchangeRequests": [
        {
            "requestId": "81d314c1-913f-4686-827e-ef2a65ccc370",
            "senderId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
            "senderPublicKey": "RUNLMSAAAAC2lkqYcTGhutQPxcjvoqUELKoy0"
        }
    ],
    "message": "SUCCESS",
    "success": true
}
```


Finally, requested party should be able to confirm or decline the key exchange request, the `DELETE: api/key-exchange-requests` endpoint should be implemented then. The server is able to fetch the request thanks to the body endpoint takes


```
{
    "requestId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "confirmed": true,
    "publicKey": "string"
}
```


Therefore, an identifier of awaiting request is passed to the server among with boolean value indicating the confirmation. Under the roof of this operation are also generation of private-public keys pair for the requested party and generation of common secret stored in client's file system. As result, the initial request sender receives a public key as confirmation from requested party. Requested side may get all his public keys via the REST web–service using the resource `GET: api/public-keys`


```
{
    "publicKeys": [
        {
            "partnerId": "ae9e10a4-0c7e-4911-8450-4139d4a114a7",
            "partnerPublicKey": "RUNLMSAAAAAbc49wfaZ+QF9J2cu1S66bkp0"
        }
    ],
    "message": "SUCCESS",
    "success": true
```

```
}
```

Now requested participant is able to derive the common secret. In order to provide an example, a simple command line interface is implemented. We have used an Elliptic Curve Diffie–Hellman implementation `ECDiffieHellmanCng Class` from the namespace `System.Security.Cryptography` of the .NET base class library. The `P-256` curve is used.

More precisely, the following CLI commands are implemented

- `MangoAPI.DiffieHellmanConsole login SENDER_EMAIL SENDER_PASSWORD`
- `MangoAPI.DiffieHellmanConsole key-exchange RECEIVER_ID`
- `MangoAPI.DiffieHellmanConsole key-exchange-requests`
- `MangoAPI.DiffieHellmanConsole confirm-key-exchange REQUEST_ID`
- `MangoAPI.DiffieHellmanConsole print-public-keys`
- `MangoAPI.DiffieHellmanConsole create-common-secret RECEIVER_ID`

Commands are self-explanatory, therefore we skip the detailed documentation on them. An example of console output straightforward

```
PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole login $env:SENDER_EMAIL $env:SENDER_PASSWORD
Attempting to login ...
Writing tokens to file ...
Login operation success.

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole key-exchange $env:RECEIVER_ID
Key exchange request with an ID 5810fb94-e3ce-4d2a-a033-95366a7c2b30 created successfully.
Writing private key to file...
Writing public key to file ...
Key exchange request sent successfully.

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole login $env:RECEIVER_EMAIL $env:RECEIVER_PASSWORD
Attempting to login ...
Writing tokens to file ...
Login operation success.

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole key-exchange-requests
RequestId: 5810fb94-e3ce-4d2a-a033-95366a7c2b30
SenderId: fd3c67c5-c6ff-4a5d-a166-98ece1b7752b
Sender Public Key: RUNLMSAAAADBFEUGJcwSeMlwDi4Jf4s6IAbCvP5w0TiYB/G/iKB3IhbSgSNeG0PUje2NBdjp534psQbKf6Gup7JfM5zwxiVy

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole confirm-key-exchange 5810fb94-e3ce-4d2a-a033-95366a7c2b30
Writing private key to file...
Writing public key to file ...
Writing common secret to file...
Key exchange request confirmed successfully.

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole print-public-keys
PartnerId: fd3c67c5-c6ff-4a5d-a166-98ece1b7752b
Public Key: RUNLMSAAAADBFEUGJcwSeMlwDi4Jf4s6IAbCvP5w0TiYB/G/iKB3IhbSgSNeG0PUje2NBdjp534psQbKf6Gup7JfM5zwxiVy

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole login $env:SENDER_EMAIL $env:SENDER_PASSWORD
Attempting to login ...
Writing tokens to file ...
Login operation success.

PS C:\Users\pkolosov> MangoAPI.DiffieHellmanConsole create-common-secret $env:RECEIVER_ID
Writing common secret to file...
Common secret generated successfully.
```

Figure 3. Diffie—Hellman key exchange console output.

In order to repeat the outputs on the screenshot the user may reference to the resources

- API: https://back.mangomessenger.company/swagger
- Source: https://github.com/MangoInstantMessenger/MangoMessengerAPI

Finally, both test accounts reached the same base 64 common secret.

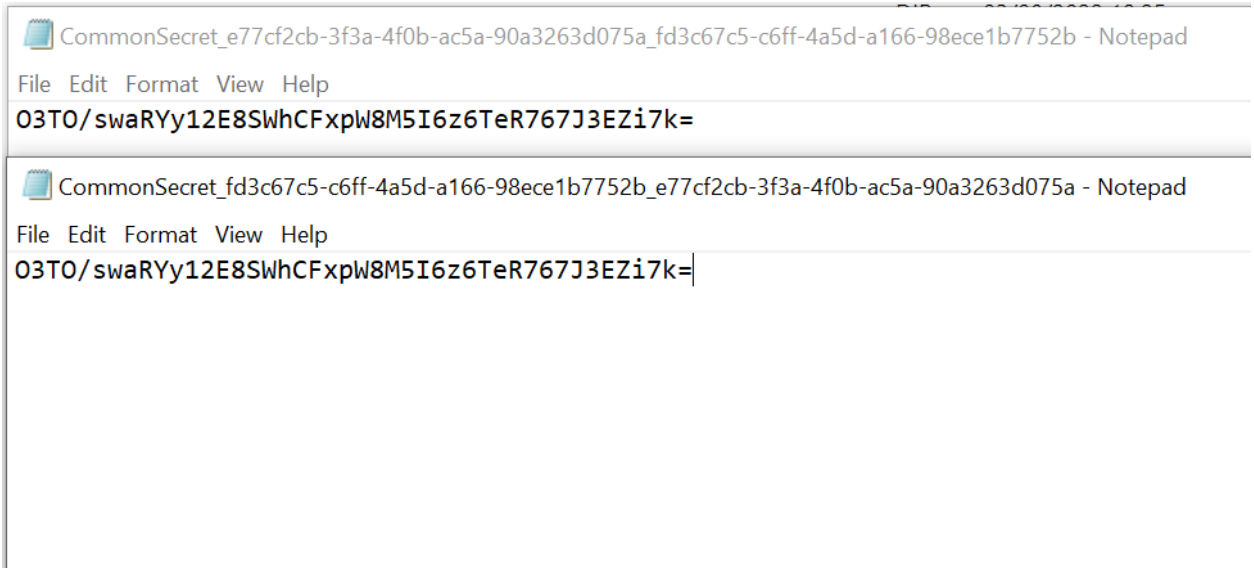

Figure 4. Common secrets.

3. Applying an `openssl`

It is very important to note that `ECDiffieHellmanCng Class` performs the native calls to Windows API, so that it is not compatible with any UNIX based operating systems. Therefore, such problem may be solved by means of `openssl` the cross-platform library. Thus, the key exchange is to be done as follows

- Generate the Diffie-Hellman global public parameters, saving them in the file dhp.pem:
  ```
  openssl genpkey -genparam -algorithm DH -out dhp.pem
  ```
- Each user now uses the public parameters to generate their own private and public key, saving them in the file dhkey1.pem (for user 1) and dhkey2.pem (for user 2):
  ```
  openssl genpkey -paramfile dhp.pem -out dhkey1.pem
  openssl genpkey -paramfile dhp.pem -out dhkey2.pem
  ```

- The users must exchange their public keys. First extract the public key into the file dhpub1.pem (and similar user 2 creates dh2pub.pem - this step is not shown below):
  ```
  openssl pkey -in dhkey1.pem -pubout -out dhpub1.pem
  openssl pkey -in dhkey1.pem -pubout -out dhpub2.pem
  ```

- Derive the common secrets:

```
openssl pkeyutl -derive -inkey dhkey1.pem -peerkey dhpub2.pem -out secret1.bin
openssl pkeyutl -derive -inkey dhkey2.pem -peerkey dhpub1.pem -out secret2.bin
```

References

[OCJ+15] Shyue Ping Ong, Shreyas Cholia, Anubhav Jain, Miriam Brafman, Dan Gunter, Gerbrand Ceder, and Kristin A Persson. The materials application programming interface (api): A simple, flexible and efficient api for materials data based on representational state transfer (rest) principles. *Computational Materials Science*, 97:209–215, 2015.

Email address: kolosovp94@gmail.com
URL: https://razumovsky.me