DEFEATING MITM ATTACKS ON CRYPTOCURRENCY EXCHANGE ACCOUNTS WITH INDIVIDUAL USER KEYS

Cheman Shaik

VISH Consulting Services Inc, 6242 N Hoyne Avenue, Chicago IL 60659, USA

Email: cheman_shaik@rediffmail.com

ABSTRACT

Presented herein is a User-SpecificKey Scheme based on Elliptic Curve Cryptography that defeats man-in-the-middle attacks on cryptocurrency exchange accounts. In this scheme, a separate public and private key pair is assigned to every account and the public key is shifted either forward or backward on the elliptic curve by a difference of the account user's password. When a user logs into his account, the server sends the shifted public key of his account. The user computes the actual public key of his account by reverse shifting the shifted public key exactly by a difference of his password. Alternatively, shifting can be applied to the user's generator instead of the public key. Described in detail is as to how aman-in-the-middle attack takes place and how the proposed scheme defeats the attack.

Provided detailed security analysis in both the cases of publickey shifting and generator shifting. Further, compared the effectiveness of another three authentication schemes in defending passwords against MITM attacks.

KEYWORDS

Cryptocurrency Exchange, Elliptic Curve Cryptography, Man-in-the-middle Attack, MITM Attack, Public Key, Private Key, Key Spoofing, Shifting.

1. INTRODUCTION

Man-in-the-middle (MITM) attack is a very serious concern for cryptocurrency exchanges as their user accounts are deemed huge honey pots for hackers. In a MITM attack, a hacker positions himself between an account user's computer and the exchange server [1]. One viable technique of MITM attack is key spoofing wherein the attacker intercepts the public key of the server and replaces it with his public key. Unaware of the attack, the user encrypts his account password during login with the attacker's fraudulent public key and submits the ciphertext to the exchange server. As the attacker has already positioned himself between the user and the server, he intercepts the ciphertext and decrypts it with his matching private key that will exactly decrypt the password to its original plain text [2]. The stolen password is used later for stealing crypto assets from the account.

During a login session, a user's password is encrypted by the server's public key which is certified by a certifying authority. Standard browsers check the key certificate and verify the authenticity of the public key and alert the user on any mismatch on ownership of the key [3]. However, only a few astute users that are technically knowledgeable of the attack understand its

complicacies. Most layman users ignore the alert and proceed further to submit their login credentials.

Even in case a hybrid encryption approach is used wherein the user sends the server a symmetric key encrypted by its public key, which is used for symmetric encryption of the remaining communication, the same MITM attack becomes successful. In this case, the attacker will compromise the symmetric key instead of the user's password which will subsequently lead to password compromise.

Every exchange server obtains a public and private key pair from a certificate authority to secure all its communication with its account users. The same key pair is used to encrypt and decrypt all communications, irrespective of the user the server is communicating with, and this practice gives way to MITM attacks through key spoofing.

Fig. 1 below illustrates how encryption and decryption of user credentials take place using the server's public and private keys.

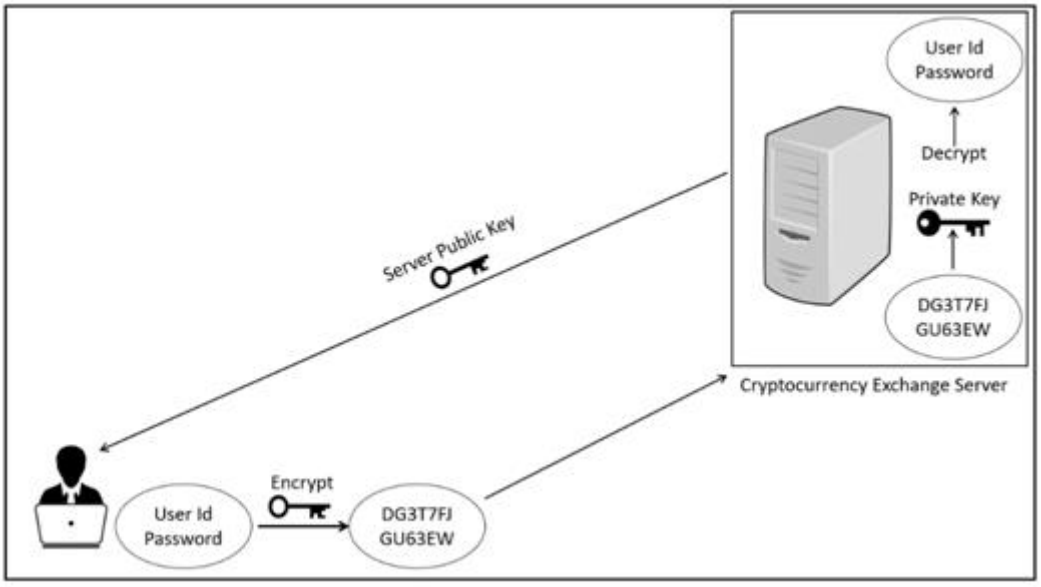

Fig. 1 Encryption and decryption of user login credentials

As shown in the above figure, the server's public key is passed to the user machine during login. The user's login credentials are encrypted with the public key to generate a ciphertext which is passed to the server where it is decrypted by the server's private key to generate the plain password. The public and private key pair remains the same for every communication with the server, irrespective of the user. The public key needs to be certified by a certifying authority.

Fig. 2 below illustrates how a MITM attacker positions himself between a user computer and the cryptocurrency exchange server.

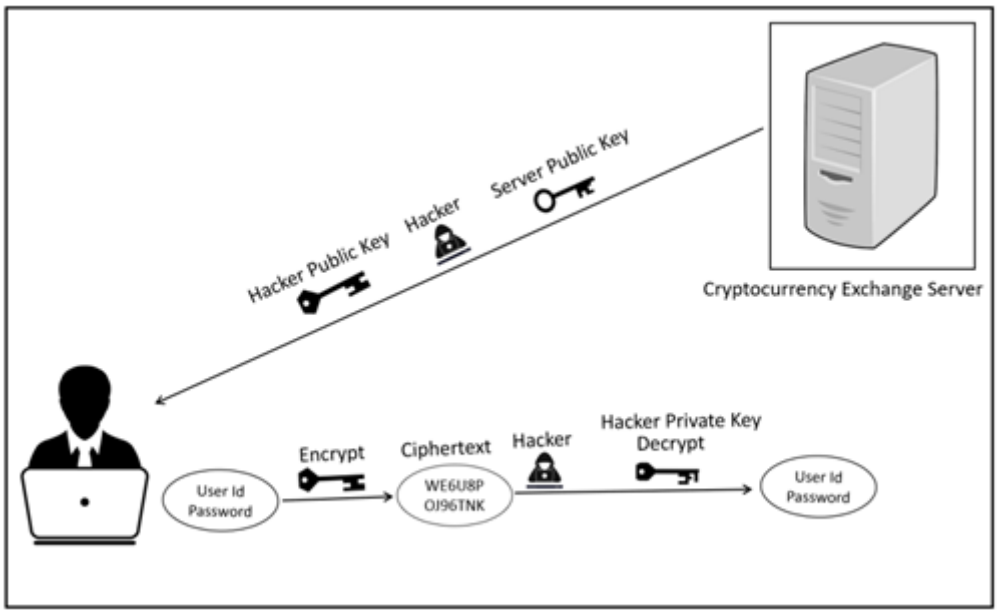

Fig. 2 MITM Attack by hacker on user login credentials

As shown in the above figure, a MITM attacker positions himself between a user computer and the cryptocurrency exchange server. At the time of user login, the attacker intercepts and replaces the server's public key with his own public key for which he has already computed a matching private key. Unaware of the attack, the user encrypts his credentials withthe attacker's fraudulent public key, creates a ciphertext and submits it. On its way to the server, the attacker intercepts the ciphertext and decrypts it with his fraudulent private key which will exactly result in the original plaintext login credentials [4].

One viable way of defeating such MITM attacks is to create and assign a separate and exclusive key to each user, which is related to the user's password itself some way or the other.

2. RELATED WORK

In 2007, Trabelsi et al proposed a solution to ARP poisoning attacks often used as part of MITM attacks. The proposed solution replaces the traditional stateless ARP cache with a stateful ARP cache and also uses a novel Fuzzy Logic approach to differentiate malicious ARP replies from genuine ones [5]. However, this solution works only for Local Area Networks whereas the Internet is an open network where the user or the application server has no control.

In 2012, Krishna Kumar et al explored MITM attacks and suggested a solution that forces authentication before key exchange. The solution requires that both the sender and receiver have a private key and digitally sign some exponents they exchange during communication [6]. However, in a login scenario, the login user requires no private key and only the server does.

In 2013, Italo Dacosta et al proposed a new protocol, Direct Validation of Certificates (DVCert) that enables domains to directly and securely vouch for their certificates using previously established user authentication credentials instead of depending on third-parties for certificate validation [7]. However, this approach has faced critical challenges due to its cost, complexity and its introduction of new privacy risks.

In 2014, Sounthiraraj et al discovered that many Android apps are vulnerable to SSL/TLS man-in-the-middle attacks. They presented a system called SMV-HUNTER to automatically detect such vulnerabilities by combining both static and dynamic analyses [8]. However, this is only a method of detecting Android apps from the Google Play Store vulnerable to MITM attacks. In fact, they did not present any defeating mechanism to defeat MITM attacks.

In 2015, Aqeel Sahi et al proposed a method to secure the Diffie-Hellman protocol using Geffe generation of binary sequences that can fortify systems against MITM attacks. Geffe generator generates pseudo random sequences with high level randomness [9]. However, this method is designed for communication between two users and it requires that each user compute and possess a private key. A secure user login to a cryptocurrency exchange account does not require a private key, which would be a burden for the user.

Le Wang et al proposed in 2016 an approach to detect MITM attacks based on the received signal strength indicators which indicate the power level being received by the antenna. The received indicators are processed and analyzed to detect any rogue access points using which hackers launch their MITM attacks [10]. However, this approach provides only a mechanism to detect rouge Aps on wireless networks, which are used to as a means to MITM attacks. Itdoe not provide any defeating mechanism to defeat MITM attacks.

In 2017, Robbi Rahim discussed in his research paper a method to defeat MITM attacks using interlock protocol created by Ron Rivest and Adi Shamir [11]. In his method Rahim splits the ciphertext into two parts and sends both the parts to the other party. However, when a man-in-the-middle collects both the parts, the message can be decrypted.

In 2017, Sanjeev Kumar et al presented an identity based authentication method wherein the server creates a unique id as a function of the server serial number, user computer's serial number and the user's government id such as social security number[12]. However, in the United States, social security number is treated as very confidential information and it is not advisable to submit it to any third party servers.

3. THE BASIC ECC SCHEME

Table.1 Notions Used

Description	**Symbol**
Generator	G
Server Public Key	P_s
Server Private Key	n_s
Message	M
Point on Elliptic Curve Encoding Message M	P_M
Random Number	r
Ciphertext	C_M

The basic elliptic curve cryptography scheme generates a public and private key based on some elliptic curve discrete mathematical problem. The key computation, message encryption and decryption will take place through the following steps:

- Server selects an arbitrary discrete Generator G on the elliptic curve
- Server selects an arbitrary number n_s as its private key
- Server computes its public key $P_s = n_s G$
- Server passesG and P_s to login user

- User encodes his message M to a discrete point P_M on the elliptic curve
- User selects a random number r
- User computes his ciphertext $C_M = \{r\ G, P_M + rP_s\}$ and sends C_M to server
- Server computes the original message $P_M = P_M + r\ P_s - n_s r\ G$

Replacing P_s with $n_s\ G$ in the above equation, $P_M + r\ n_s\ G - n_s r\ G = P_M$. Hence, the original message is obtained.

4. USER SPECIFIC ECC KEY SCHEME

Table.2 Additional Notions Used

Description	**Symbol**
Point on Elliptic Curve Encoding Password	P_w
Shifted Public Key on Server	P_{sw}
Shifted Public Key of Hacker	$P_{sw\text{-}hack}$
Hacker's Public Key	$P_{s\text{-}hack}$
Hacker's Private Key	$n_{s\text{-}hack}$
Hacker's Generator	G_{hack}
Shifted Generator	G_w
Hacker's Shifted Generator	$G_{w\text{-}hack}$

In User Specific ECC Key Scheme, the server creates a separate public and private key pair for each user and the user's password is encrypted using that particular public key whenever he logs into his account. The public key or the generator of the user is shifted either backward or forward on the elliptic curve by a difference of the user's password.

4.1. Public Key Shifting by Password

In this case the public key $P_s$is shifted either backward or forward to a new point on the elliptic curve by a difference of the user's password and the shifted public key is passed to the user instead of the original one. In backward shifting, the serverencodesthe user's password to a discrete point P_w on the elliptic curveand computes the shifted public key $P_{sw}= P_s – P_w$.All the three values – the generator G, the private key n_s, and P_{sw} are stored in the database as a user record.

4.1.1. User Sign Up

When a user signs up for a new account on the server, he fills in his user id and password and submits the form which passes the encrypted credentials to the server. Fig.3 below illustrates capturing of the user's password, creating his key pair and shifting the public key backward by the password.

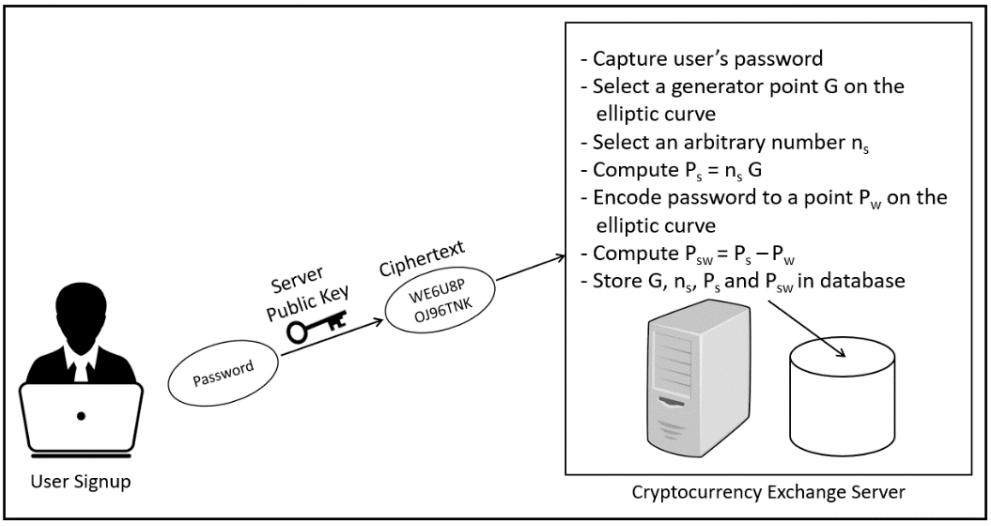

Fig.3 User signup and shifted public key creation

The server public key shown in the figure is not the userspecific key. It is the server's generic public key obtained from a certifying authority. On retrieving the user's password from the user's signup request, the server performs the following steps:

- Decrypt and capture the user's password
- Select an arbitrarygenerator G on the elliptic curve
- Select an arbitrary number $n_s$as the private key
- Compute $P_s = n_s G$
- Encode user's password to a point P_w on the elliptic curve
- Compute the shifted public key $P_{sw}= P_s - P_w$
- Store G, $n_s$andP_{sw}in database

4.1.2. User Login

When a user wants to login to his account, the following steps as shown in Fig. 4 will take place:

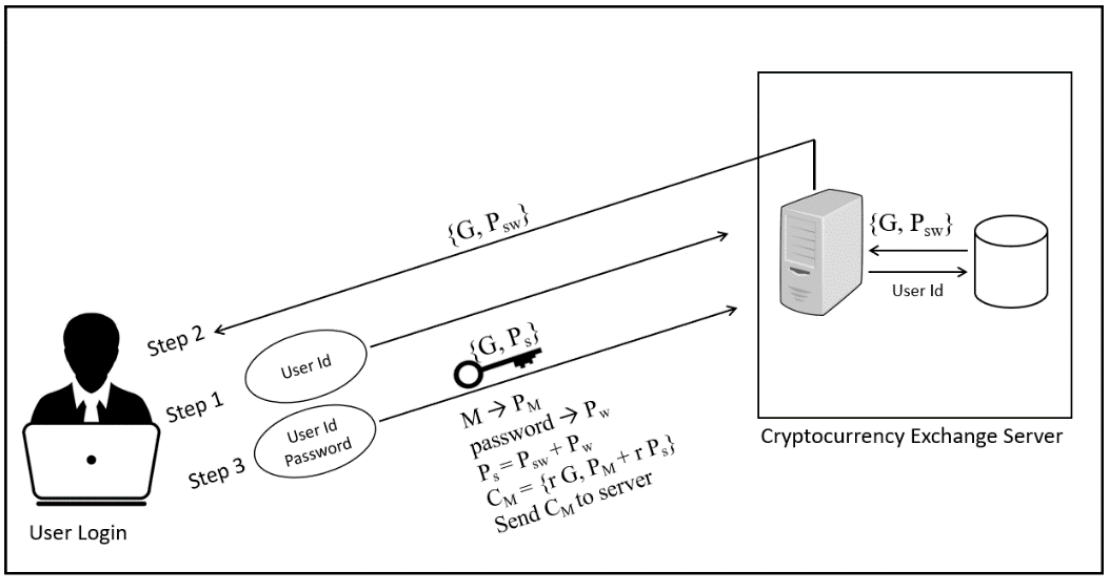

Fig.4 User login with shifted public key passed from server

- User sends only his userid to the server
- Server retrieves the G and P_{sw}values of the user from the database based on his user id
- Server sends G and $P_{sw}$to the login user's machine
- User encodes his message M to a discrete point P_M on the elliptic curve
- User encodes his password to a discrete point $P_w$on the elliptic curve
- User computes $P_s = P_{sw} + P_w$
- User selects a random number r
- User computes his ciphertext $C_M = \{r\ G, P_M + r\ P_s\}$and sends C_M to server
- Server retrieves the private key n_s of the user account
- Server computes the original message by subtracting the first term multiplied by n_s from the second term as below:

$$P_M = P_M + r\ P_s - n_s r\ G$$

Replacing P_s with $n_s\ G$ in the above equation, $P_M + r\ n_s\ G - n_s r\ G = P_M$. Hence, the original message is obtained.

4.1.3. Security Analysis

The additional point P_w involved in the computation of P_s foils MITM attacks. When a MITM attacker positions himself in between the user and server, he needs to replace P_{sw} with his own $P_{sw\text{-}hack}$ which should be computed as $P_{s\text{-}hack} - P_w$ for which he needs the user's password. Without knowing the user's password, if the attacker passes his own G_{hack} and $P_{s\text{-}hack}$, the user computes his ciphertext $C_M = \{r\ G_{hack}, P_M + r\ P_s\}$ where $P_s = P_{sw\text{-}hack} + P_w$and the hacker intercepts and tries to decrypt it as follows:

Multiplying the first term of the ciphertext C_M with the hacker's private key $n_{s\text{-}hack}$ and subtracting it from the second term,

$$\begin{aligned} P_M + r\ P_s - n_{s\text{-}hack}\ r\ G_{hack} &= P_M + r\ (P_{s\text{-}hack} + P_w) - n_{s\text{-}hack}\ r\ G_{hack} \\ &= P_M + r\ (n_{s\text{-}hack}\ G_{hack} + P_w) - n_{s\text{-}hack}\ r\ G_{hack} \\ &= P_M + r\ n_{s\text{-}hack}\ G_{hack} + r\ P_w - n_{s\text{-}hack}\ r\ G_{hack} \end{aligned}$$

Canceling the second and forth terms in the above, the MITM attacker is left with $P_M + r\ P_w$ which is not the original message and eventually it reads junk for the attacker. The attacker can compute the original message P_M by subtracting rP_w from $P_M + r\ P_w$. However, it is not possible for him to guess the random number selected by the user and his password.

Similarly, in forward shifting of the public key, the server computes the shifted public key as $P_{sw} = P_s + P_w$ and the user computes the original public key as $P_s = P_{sw} - P_w$. Rest all steps of computation remain the same on both the user and server side.

Requiring the password to hack a password is a paradoxical situation defeating MITM attacks.

4.2. Generator Shifting by Password

In this case, the Generator G is shifted either backward or forward to a new point on the elliptic curve by a difference of the user's password, and the shifted Generator G_w is passed to the user instead of the original Generator. In backward shifting, the server encodes the user's password to a discrete point P_w on the elliptic curve and computes the shifted Generator $G_w = G - P_w$. All the three values G_w, the private key n_s and the public key P_s are stored in the database as a user record.

4.2.1. User Sign Up

When a user signs up for a new account on the server, he fills in his user id and password and submits the form which passes the encrypted credentials to the server. Fig.5 below illustrates capturing the user's password, creating his key pair and shifting the generator backward by the password.

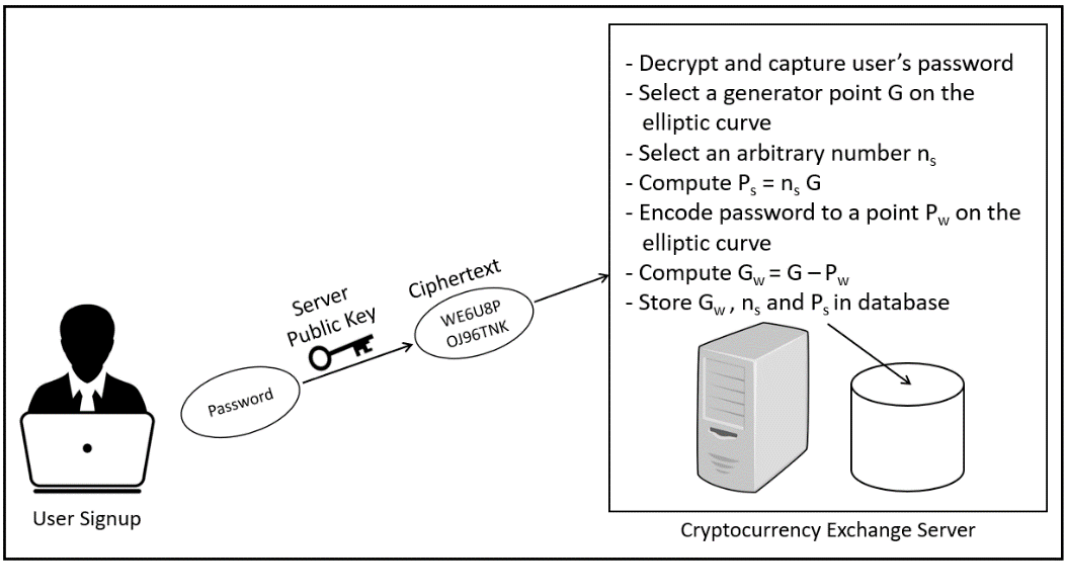

Fig.5 User signup and public key creation with shifted generator point

The server public key shown in the figure is not the userspecific key. It is the server's generic public key obtained from a certifying authority. On retrieving the user's password from the user's signup request, the server performs the following steps:

- Decrypt and capture user's password
- Select an arbitrarygenerator G on the elliptic curve
- Select an arbitrary number $n_s$as the private key
- Compute $P_s = n_s G$
- Encode user's password to a point P_w on the elliptic curve
- Compute the shifted generator $G_w = G - P_w$
- Store G_w, $n_s$and P_s in database

4.2.2. User Login

When a user wants to login to his account, the following steps as shown in Fig. 6 will take place:

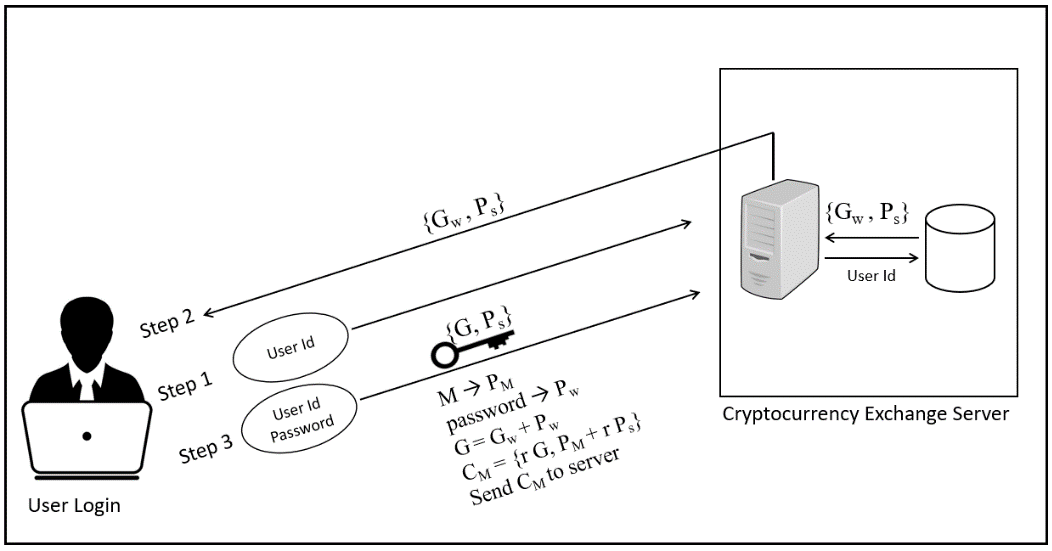

Fig.6 User login with shifted generator point passed from server

- User sends only his user id to the server
- Server retrieves the G_w, $n_s$and P_s values of the user from the database based on his user id
- Server sends G_w and P_s to login user's machine
- User encodes his message M to a discrete point in the elliptic curve P_M
- User encodes his password to $P_w$on the elliptic curve
- User computes $G = G_w + P_w$
- User selects a random number r
- User computes his ciphertext $C_M = \{r\ G, P_M + r\ P_s\}$and sends C_M to server
- Server retrieves the private key n_s of the user account
- Server computes the original message $P_M = P_M + r\ P_s - n_s r\ G$

Replacing P_s with $n_s\ G$ in the above equation, $P_M + r\ n_s\ G - n_s r\ G = P_M$. Hence, the original message is obtained.

4.2.3. Security Analysis

The additional point P_w involved in the computation of G_w foils MITM attacks. When a MITM attacker positions himself in between the user and server, he needs to replace $G_w$with his own $G_{w\text{-}hack}$ which should be computed as $G_{hack} - P_w$ for which he needs the user' password. Without knowing the user's password, if the attacker passes his own G_{hack} and $P_{s\text{-}hack}$, the user computes his ciphertext $C_M = \{r\ G, P_M + r\ P_{s\text{-}hack}\}$ where $G = G_{hack} + G_w$, and the hacker intercepts and tries to decrypts it as below:

Multiplying the first term of the ciphertext $C_M$with the hacker's private key $n_{s\text{-}hack}$ and subtracting it from the second term,

$$P_M + r\ P_{s\text{-}hack} - n_{s\text{-}hack}\ r\ G = P_M + r\ P_{s\text{-}hack} - n_{s\text{-}hack}\ r\ G$$
$$= P_M + r\ P_{s\text{-}hack} - n_{s\text{-}hack}\ r\ G_{hack} - n_{s\text{-}hack}\ r\ G_w$$
$$= P_M + r\ n_{s\text{-}hack}\ G_{hack} - n_{s\text{-}hack}\ r\ G_{hack} - n_{s\text{-}hack}\ r\ G_w$$

Substituting $P_{s\text{-}hack} = n_{s\text{-}hack}\ G_{hack}$the above expression becomes

$$P_M + r\ n_{s\text{-}hack}\ G_{hack} - n_{s\text{-}hack}\ r\ G_{hack} - n_{s\text{-}hack}\ r\ G_w$$

Cancelling the second and third terms in the above expression, the MITM attacker is left with P_M - $n_{s\text{-}hack}$ r G_w which is not equal to the original message P_M and eventually it reads junk for the attacker.

To remove the effect of $n_{s\text{-}hack}$ r G_w the attacker needs to know the random value r and the user's password.

Similarly, in forward shifting of the generator, the server computes the shifted generator G_w = $G+P_w$ and the user computes the original generator $G= G_w- P_w$. Rest all steps of computation remain the same on both the user and serverside.

Requiring thepassword to hack a password is again a paradoxical situation defeating MITM attacks.

Fig. 7 shows the authentication process taking place on the server when it receives the ciphertext C_M from the user.

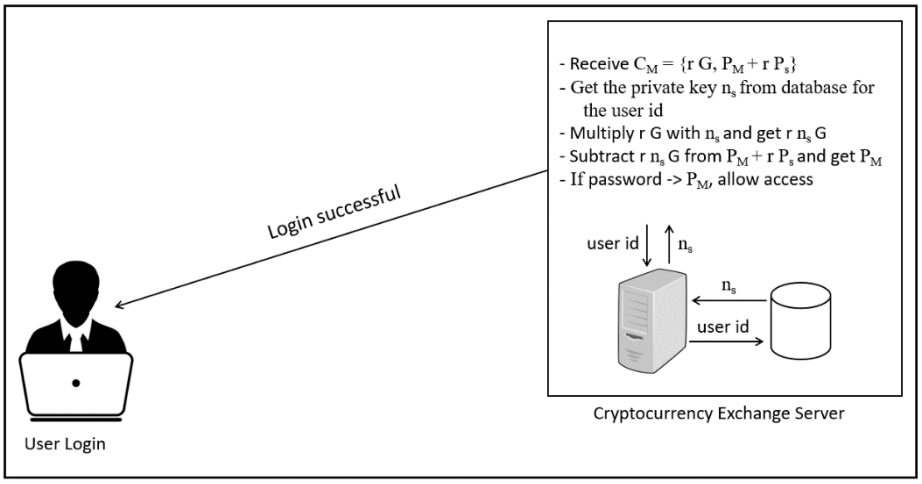

Fig.7 Validation of ciphertext on cryptocurrency exchange server for successful login

5. FORGOT PASSWORD AND UPDATE PASSWORD IMPLEMENTATION

When a user forgets his password and creates a new password, the same procedure as discussed in section 4 will take place. The existing values P_{sw} or G_w are cleared from the database and the newly computed values are updated in the user record.

Most online applications require that their users change their passwords once every month to secure them from brute force attacks. Even in such a case, the existing values are replaced with the newly computed values.

6. APPLICABILITY TO PASSWORD HASH

Today, most servers store their user passwords only in hash form and not in encrypted form. Hashed password prevents their compromise by internal adversary elements as hash functions are irreversible. The industry standard hash function SHA256 offers good security to passwords stored in database. SHA512 and higher bit hash functions also can be used to hash passwords for accounts that store very huge crypto assets and perform high value transactions.

User Specific Key scheme can also be implemented with hashed passwords. In this case, the public key or the generator of the user is shifted either backward or forward by a difference of the password hash as below:

Backward shifting of public key: $P_{sw\text{-}hash} = P_s - P_{w\text{-}hash}$
Forward shifting of public key: $P_{sw\text{-}hash} = P_s + P_{w\text{-}hash}$

When a user tries to login to his account, the server retrieves the shifted public key of the account $P_{sw\text{-}hash}$ and passes it to the user. The user runs the same hash function on his password and computes the actual public key $P_s = P_{sw\text{-}hash} + P_{w\text{-}hash}$ for backward shifting and $P_s = P_{sw\text{-}hash} - P_{w\text{-}hash}$ for forward shifting.

If the hash length is higher than the ECC key length, the password hash may be truncated after N bits before computing the shifted public key or generator, where N is the ECC key length, and the same truncation is performed over the hash by the user.

7. TECHNICAL IMPLEMENTATION

On the serverside, computation of the public key, private key, shifted public key or shifted generator and decryption can be achieved through any standard serverside programming languages such as Java, Python, Scala, C++ and C#. Alternatively, it can be achieved using any standard cryptography libraries.

On clientside, computation of the actual public key of the user account and encryption can be achieved through any standard browser side scripting languages such as JavaScript. Alternatively, any standard JavaScript cryptography libraries can be used.

With the User Specific ECC scheme, authentication takes two steps:user submitting his user id to the server and getting from it his shifted public key and generator or vice versa in the first step, and encrypting his password and submitting it to the server in the second step. This can be achieved with a single button click without any page refresh using AJAX (Asynchronous JavaScript and XML) programming.

User Specific ECC scheme can be wrapped as an additional layer under the standard SSL/TLS layer applied in all secure communications on the web. With this implementation, the user password or its hash encrypted with his specific public key is in turn encrypted by the SSL/TLS public key of the server which is common for all users. On serverside the SSL/TLS ciphertext is decrypted first followed by the user specific decryption with the account specific private key on the server.

8. PERFORMANCE ANALYSIS

At the time of registration, the User Specific Key Scheme computes a key pair for the user. It is a one-time computation only. Thereafter, during every login the server and user need to add or subtract the shifted public key or generator which adds only a negligible computational overhead.

When encryption or decryption is performed, the ECC algorithm performs several addition operations over the selected elliptic curve depending upon the key size. One more addition or subtraction of the shifted key or generator causes very negligible computational overhead, which at the same time defeats MITM attacks.

9. ADVANTAGES OF THE KEY SCHEME

User Specific Key scheme defeats MITM attacks through key spoofing as each user has a specific key assigned to his account on the server. Actually, the userspecific public and private keys are stored on serverside and the user is not required to store or remember any of these keys. At the time of login what the user gets from the server is his public key and generator where one of them is shifted by a difference of his password or password hash in backward or forward direction. As the actual public key or generator is computed using the user's password which he enters in his login form, the MITM attacker fails to successfully decrypt the password ciphertext even though he intercepts and replaces the key parameters sent from the server.

If a user registration went fine without any MITM attack, the MITM attacker will never be able to compromise his account. Most of the banks provide mobile apps to enable their customers to perform transactions. These apps do not provide any public and private key pair to their customers. Similarly, customers performing online banking transactions through the banks' websites do not possess personal keys for transaction security. Even in case banks and cryptocurrency exchanges provide public and private keys to their account holders, it can not stop the MITM attacker from reading through the original plain text information signed by the user's private key and the attacker can steal any confidential information in the message. This is because the user encrypts the message with the attacker's public key for which the attacker is already in possession of the matching private key.

User Specific Key scheme individually fortifies the security of each user account. Even if an attacker compromises a particular user's password by brute force attack, all the remaining accounts are still secured. The attacker needs to compromise each account individually which is practically infeasible. On the other hand, if an attacker is able to spoof the SSL/TLS public key or compute the private key from the public key through an extensive computing effort, he can compromise all user accounts saved on the server.

10. COMPARISON WITH OTHER AUTHENTICATION SCHEMES

The following table shows the effectiveness of different authentication methods, including the User Specific Key Scheme, in defending user accounts against MITM attacks.

Authentication Scheme	**Effectiveness in Defending MITM Attacks**
User Specific Key Scheme	Secures passwords and messages from compromise.
Digital Signatures	Ensures message integrity but can't stop attacker from viewing the original message text.
Identity Based Authentication	Provides authentication without passwords but may lead to stealing and misuse of identity information such as SSN, Aadhar Card etc.
Interloc Protocol	Splits password ciphertext into two parts. However, attacker can intercept both parts and can decrypt them.

11. CONCLUSION

Man-in-the-middle (MITM) attack is a very serious attack that has been causing huge financial losses to cryptocurrency buyers, traders as well as exchanges. Usually, hackers launch MITM

attacks to target account passwords using which they steal crypto coins from accounts on the exchange server. MITM attacks have been successful despite the implementation of SSL/TLS protocol over online communications.

A user specific ECC key scheme is proposed that defeats MITM attacks through key spoofing. The proposed scheme creates a separate public and private key pair for each account holder at the time of sign up. The public key or the generator of the user is shifted on the elliptic curve backward or forward by a difference of the user's password or its hash. At the time of login, the server passes the shifted parameter to the user using which the user reverse shifts his parameter before encryption to its original value by adding or subtracting his password point on the elliptic curve. The additional password parameter involved in the encryption foils MITM attacks.

The key generation process, public key or generator shifting, encryption and decryption are explained with detailed steps and also proved as to how the encryption thwarts MITM attacks, thereby saving individuals and financial organizations from heavy losses.

The proposed scheme works with plain, encrypted and also hashed passwords stored in the database. It also offers another advantage of individual security to each user account. Even if the attacker compromises one particular account, rest all accounts remain intact, which is not the case with the general SSL/TLS encryption protocol implemented on the server.

A recommendation for future work is that security product developers conduct a proof of concept of the proposed scheme and test it with ethical MITM attacks spoofing the public key.

REFERENCES

[1] Imperva.com, "Man in the middle (MITM) attack", https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/

[2] Stackexchange.com, "What's the actual danger of public key spoofing?", https://security.stackexchange.com/questions/113347/whats-the-actual-danger-of-public-key-spoofing Xxx

[3] Digicert.com, "Name Mismatch in Web Browser", https://www.digicert.com/kb/ssl-support/certificate-name-mismatch-error.htm

[4] AvijitMallike, Zaglul Shahadat, Abid Ahsan, Jia-Chi Tsou, "Man-in-the-middle-attack: Understanding in simple words", International Journal of Data and Network Science 3 (2019)

[5] Z. Trabelsi and W. El-Hajj, "Preventing ARP Attacks Using a Fuzzy-Based Stateful ARP Cache," 2007 IEEE International Conference on Communications, Glasgow, 2007, pp. 1355-1360, doi: 10.1109/ICC.2007.228.

[6] C. Krishna Kumar, G. Jai Arul Jose, C. Sajeev, C. Suyambulingom, "SAFETY MEASURES AGAINST MAN-IN-THE-MIDDLE ATTACK IN KEY EXCHANGE", ARPN Journal of Engineering and Applied Sciences. VOL. 7, NO. 2, FEBRUARY 2012

[7] I. Dacosta, M.Ahamad, P.Traynor, Trust No One Else: DetectingMITM Attacks Against SSL/TLSWithout Third-Parties, ConvergingInfrastructure Security (CISEC) Laboratory, Georgia Tech Information Security Center (GTISC), Georgia 2013.

[8] David Sounthiraraj, Justin Sahs, Garret Greenwood,Zhiqiang Lin,Latifur Khan, "SMV-HUNTER: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps", Proceedings of Network and Distributed System Security Symposium 2014

[9] Aqeel Sahi Khader, David Lai, "Preventing Man-In-The-Middle Attack in Diffie-Hellman Key Exchange Protocol", 22nd International Conference on Telecommunications (ICT 2015)

[10] Le Wang and Alexander M. Wyglinski, "Detection of man-in-the-middle attacks using physicallayer wireless security techniques", Wirel. Commun. Mob. Comput.2016; 16:408–426

[11] Robbi Rahim, "MAN-IN-THE-MIDDLE-ATTACK PREVENTION USING INTERLOCKPROTOCOL METHOD", ARPN Journal of Engineering and Applied Sciences - VOL. 12, NO. 22, NOVEMBER 2017

[12] Sanjeev Kumar Mandal, A R Deepti, "An Efficient Identity Based Authentication Protocol by Using Password", International Journal of Network Security & Its Applications (IJNSA) Vol.9, No.6, November 2017

AUTHOR

Cheman Shaik is a Research & Development professional in Computer Science and Information Technology for the last twenty years. He has been an inventor in these areas of technology with eight U.S Patents for his inventions in Cryptography, Password Security, Codeless Dynamic Websites, Text Generation in Foreign Languages, Anti-phishing Techniques and 3D Mouse for Computers. He is the pioneer of the Absolute Public Key Cryptography in 1999. He is well known for his Password Self Encryption Method which has earned him three U.S Patents. He has published research papers inthe international journals – IJCSEA, IJCIS and the proceedings of EC2ND 2006 and CSC 2008.