IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

SECURE AUTHENTICATION METHOD USING HONEY KEY PASSWORD IN ONLINE SHOPPING

[1]Mrs.P. Swarajya Lakshmi, [2]Bashyakarla Sucharitha, [3]Araganlapally Manidhar,[4]Lakavath Sravanthi
[1]Assistant Professor , [2]Student, [3]Student,[4]Student
[1]Computer Science and Engineering ,
[1]Anurag University, Hyderabad, India

Abstract : As online applications relying predominantly on password-only authentication, the vulnerability to password leaks from both internal and external adversaries remains a critical concern. This paper introduces an innovative solution, Honey PAKE (HPAKE), designed to overcome the limitations of existing methods, including honeywords for external attacks and augmented password-authentication key exchange (aPAKE) for insider threats. HPAKE implements a robust authentication mechanism during the registration process within online shopping platforms.Our unique registration protocol mandates users to upload a single text file and provide a secret key pair. Any modifications to these elements result in login restrictions, thereby reinforcing the authentication process. This protocol not only enhances the security of online shopping applications but also empowers the authentication server to proactively identify and address potential password leaks.Moreover, to mitigate fraudulent activities, specifically erroneous payments, users involved in such transactions will face account suspension. The integration of HPAKE signifies a substantial progression in securing user credentials and establishing a formidable defense against both internal and external threats prevalent in the online shopping domain.

***IndexTerms* - aPAKE,HPAKE.**

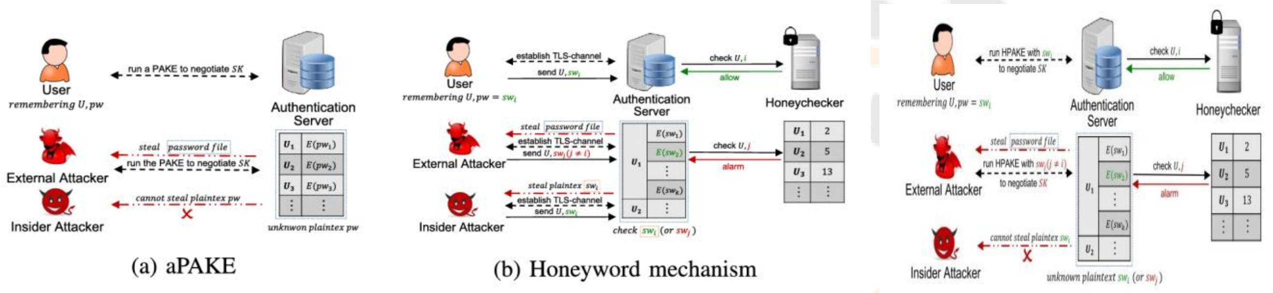
I. Introduction

INTRODUCTION

The most often used authentication method in safety-related applications is the username/password paradigm [3]. Tokens and biometrics are two examples of alternative authentication elements that necessitate the purchase of additional hardware, which is sometimes deemed too costly for a given application. Passwords, however, are vulnerable to dictionary attacks and are low-entropy secrets [3]. They must therefore be safeguarded throughout transmission. Passwords are sent using SSL/TLS, which is the most commonly used protocol [36]. However, Public Key Infrastructure (PKI) must be set up for this; PKI maintenance is costly. Furthermore, there is a risk of man-in-the-middle attacks while utilizing SSL/TLS [3]. Even though the session is fully encrypted, if a user authenticates himself to a phishing website by sharing his password, the password will be stolen. One logical approach, given the inherent weakness of passwords, would seem to be to replace them with powerful secrets, such as cryptographically secure private keys. The UK National Grid Service (NGS) used this method for user authentication [4].This has made it much more difficult for grid computing technologies to become widely used. Therefore, we have to accept the fact that weak passwords are a reality. The Password-Authenticated Key Exchange (PAKE) is a research topic that has researchers actively investigating methods to accomplish password-based authentication without utilizing PKIs or certificates [9]. The EKE protocol was introduced by Beloin and Merrit in 1992, marking the first significant event [13]. The PAKE problem was shown to be at least solvable by the EKE protocol, despite certain documented flaws [22, 26, 29, 32].Numerous protocols have been suggested since then. Many of them are just EKE variations that implement the "symmetric cipher" in different ways [9]. The majority of the few methods that assert to be resistant to known attacks are patent-protected; two prominent examples are Lucent Technologies' EKE [15] and Phoenix Technologies' SPEKE [24]. Because of this, applying these strategies cannot immediately benefit the scientific community or the larger security business [16].The EKE and SPEKE protocols only provide heuristic security. Formal security proofs seem unlikely without introducing new assumptions or relaxing criteria, given the way the two methodologies were constructed; we shall describe the specifics in Section 4. In the section that follows, we will present an alternative method for resolving the CAE problem and demonstrate how it is free from the security flaws associated with the EKE and SPEKE protocols.

NEED OF THE STUDY.

check Against peacemakers. In Figure 1a, augmented password authentication key exchange(aPAKE)(9) is designed to allow a client and a garçon to establish a session crucial predicated on a word, where the client has the word plaintext and the garçon only holds the verifier. This fashion prevents the garçon from knowing the word, and therefore resists the bigwig attacks. Since Bellovin and Merrit(9) introduced this notion, multitudinous researchers proposed various aPAKE schemes(10),(11),(12),(13) in order to meliorate the security and effectiveness performance. Among them, OPAQUE(12) is the most well- studied scheme with the strongest security and thus, it recently is homogenized by the Crypto Forum Research Group of the Internet Engineering Task Force(IETF)(14). Against outlanders. Honeyword fashion(15)(see Figure 1b) is proposed to descry the word leakage for the most common word-only authentication systems, passwordover- TLS. This approach associates t −1 bait and plausiblelooking watchwords(i.e., honeywords) to each account. The honeywords and the real word are collectively calledsweetwords.However, she can't tell the real one and presumably(with 1 −1/ t probability) log in with a honeyword, If an attacker steals the word train. also, the garçon can descry the word leakage from the “ wrong ” login. The follow- up works focus on the honeyword generation algorithms(16),(17) so as to produce farther plausible- looking baits and the discovery styles(18) to meliorate responsibility. Others. word less authentication(19) ormulti- factor authentication systems(20),(21) make good use of other factors,e.g., smartphone and point. They significantly reduce the trouble ofwordleakage.However, she still needs fresh factors to compromise account, If an attacker steals the word. either, in some of these designs, authentication garçon does not need to store the word- related data, so that indeed if the attacker compromises the storage train on garçon, she can't carry out offline word guessing as long as other factors are secure. A typical design can be seen in(21),(22),(23) that a smart device(as an authentication factor) is used to store the word- related data, making systems repel offline guessing in the case of garçon concession. shortcomings. The ways over, unfortunately, have the following shortcomings. The honeyword medium requires the client to shoot the word plaintext to the garçon(via a garçon- authenticated secure channel), differently the garçon can't tell if the login word is real. thus an bigwig can directly steal the plaintext of the login word without any guessing attacks. In aPAKE, the garçon has to store the verifiers in the word train for authentication. But an external attacker may steal the train and carry out guessing attacks(24) to recover the word. This vulnerability is essential in aPAKE. And neither of these styles can give a result maintaining security against both peacemakers and outlanders. As for other(passwordless ormulti- factor) approaches, they may give stronger security counting on spare factors, which may bring disadvantages to deployability and usability. In this paper, we do not consider them and only concentrate on passwordonly authentication. According to the below discussion, we thus raise a question “ How could one design a fast and secure word-only authentication scheme that can repel both the bigwig and external attackers.



(a) aPAKE (b) Honeyword mechanism

C) Hpake

RESEARCH METHODOLOGY

In order to provide security beyond the conventional bounds of aPAKE, we introduce the concept of honey PAKE (HPAKE), which enables the authentication server to detect password leakage. On top of the honeyword method, honey encryption, and OPAQUE, a standardized aPAKE, we also design an HPAKE structure. Our design's security is properly examined, with insider resistance and password breach detection achieved.

HPAKE offers both honeyword tactics and password leaking to **external attackers.** Through offline guessing attacks, the attacker will obtain a password list with one sweetword if the authentication service is compromised. Because the attacker can't distinguish which is real, it's likely that they use a honeyword together with HPAKE to compromise the account. As a result, a honey session key will be generated, and any use of the key will trigger an alarm.

For the **insiders**: HPAKE achieves the same level of security as aPAKE by guaranteeing that the password plaintext is never disclosed to the client. The key exchange specifically completes the authentication. Additionally, using the actual password to execute the program will produce a real session key, and only instructions that have been encrypted with that key will be permitted. Consequently, the plaintext password cannot be stolen by the server or insider.

IMPLEMENTATION (Our Contribution):

MODULES:

Online Shopping Website:

Using this module web application is developed which has online shopping features where seller can use admin module to upload products and buyer can view products and purchase. This application provides option for payment, add products to cart, view products, search products get conformation from admin on purchase, use attacker module to show internal attacks. Show security methods to secure authentication process.

Admin Module:

This module is part on online shopping website where admin and login to application add products with cost and product details and verify users as attackers or normal users and block users who are attackers. Admin can verify users for purchasing products and get confirmation.

User Module:

This module is part of online shopping website where users can register with application by entering valid user name and password along with text file data which is used for every time login. User must give same text fiile every time and apply Honey Password-Authenticated Key Exchange when he login to application which will encrypt and send key to authentication server who will verity and validate user . If Password-Authenticated Key Exchange is success then only user is considered as normal user else he is considered as attacker.

Authentication Server Module:

This module is used as middle layer between user registration process and login process verification for verifying Honey Password-Authenticated Key Exchange process. Every time new user registers this server will store a security key which is unique based on user input data . If same data is uploaded by user while login then only authentication server will give validation else authentication exchange will be failed.

HONEY KEY MECHANISM:

Honeyword Mechanism In order to identify password leakage for password-over-TLS, the honeyword technique [15] has been proposed. The honeyword technique, as illustrated in Figure 1b, produces multiple honeywords directly and saves them (as hash values) on the authentication server with the actual password. On a different server known as honeychecker, it keeps the index of the actual password in the list.
The authentication server verifies whether a password is a sweetword when a user comes in with a username of U and a password of pw (pw being supplied to the authentication server via the TLS channel):
1) Reject this login if it isn't.
2) The authentication server communicates (U, i) to the honeychecker (via a secure connection) if it is the i-th sweetword.
The honeychecker then determines whether index I is accurate for U:
a) Accept this login if it is.
b) If not, report a password leak and take the appropriate action in accordance with the security policy that has been predetermined.
This technique retains its deployability advantages because it just slightly modifies the password-over-TLS server side. Additionally, the honeychecker's interface is quite simple, making it easy to modify in order to prevent compromise.

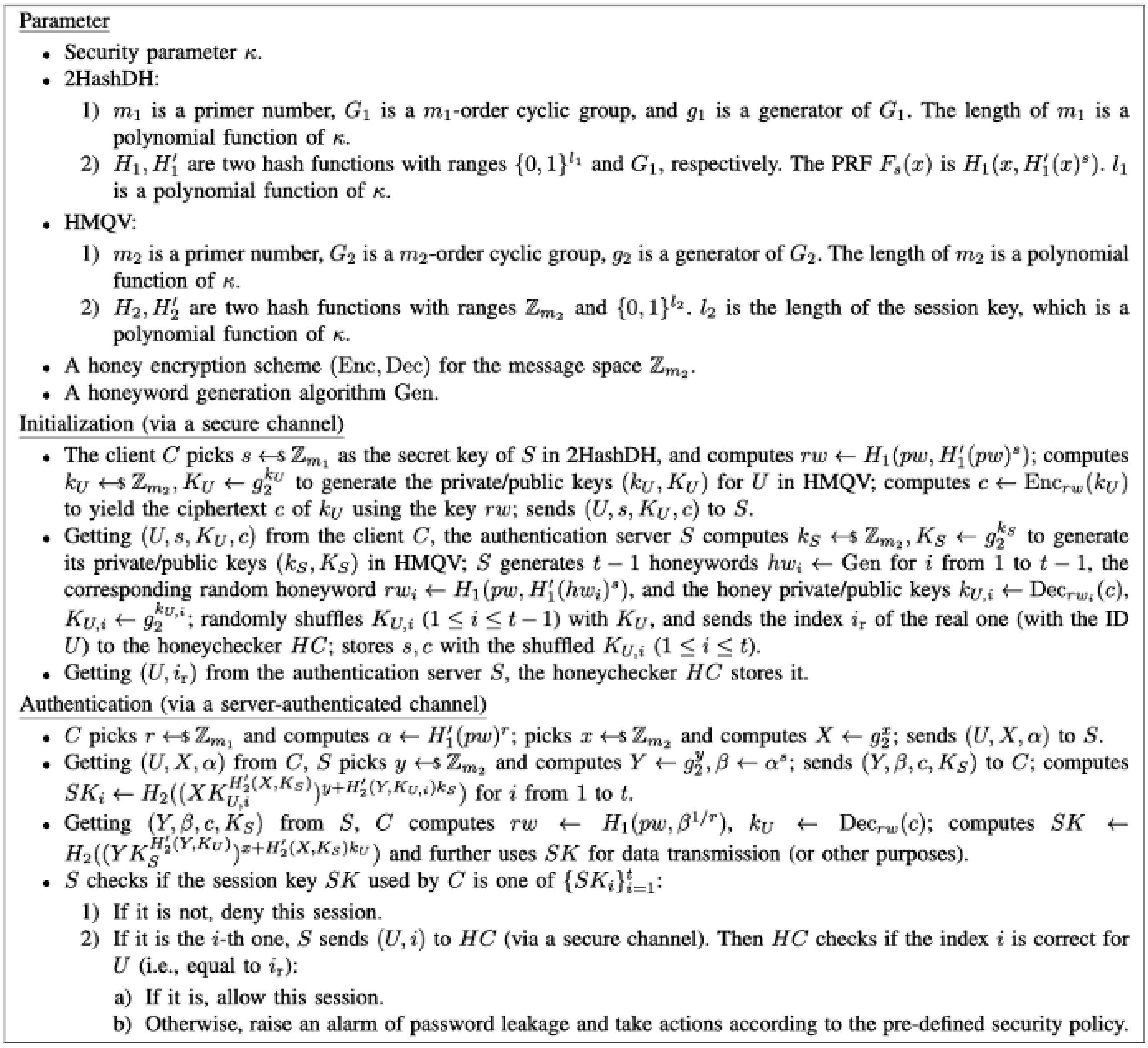
Honey Key encryption:

Encryption using Honey As seen in Figure 4b, honey encryption is a unique encryption technique that can produce decoy messages for erroneous keys [25], [32]. In order to convert the message M into a (fixed-length) uniform bit string S, it first introduces a probabilistic encoder. Next, S is encrypted using a well selected conventional encryption algorithm (see to Figure 5). The message distribution M, which might be uniform or nonuniform (for password vaults, for example, see [33]), is taken into consideration while designing the encoder.The encoder must ensure that a message sampled from M is produced upon decoding a random bit string. Formally, A, (M0, S0), and (M1, S1) are indistinguishable for an arbitrary adversary (possibly possessing infinite computing resources) (we denote (M0, S0) ~ (M1, S1)), where l is the bit string length, S0 ↞$ {0, 1} l (i.e., randomly selecting a l-bit string), M0 ↞ Decode(S0), M1 ↞p M (i.e., sampling a message from M based on the message distribution p), S1 ↞ Encode(M1), and l is the bit string length.To be more precise, |Pr[A(M0, S0) = 1 : S0 ←$ {0, 1} l, M0 □ Decode(S0)] It is insignificant, − Pr[A(M1, S1)

= 1 : M1 ←$ M, S1 ← Encode(M1)]|. Please be aware that each wrong key should result in a random bit string according to the conventional encryption method used in honey encryption. Consequently, the honey encryption algorithm will generate a l-bit string S ′ and a subsequent plausible-looking message M′ on M for every wrong key K′. Our approach for HPAKE encrypts the user's private key, kU, a uniformly random value on Zm2, using honey encryption.It is easy to design an encoder for kU. Round is the rounding function. To encode kU, we simply choose an integer number from [round(kU 2 l/m2),round((kU + 1)2l/m2)) (⊆ [0, 2 l)) as S. Then, we identify the relevant interval and decode S to get kU. With the encoder, the honey encryption strategy may generate a believable-looking private key on Zm2 for every invalid key rw.

Honey key technique is proposed to detect the password leakage for the most common password-only authentication systems, password over-TLS. This approach associates t−1 decoy and plausible looking passwords (i.e., honeywords) to each account. The honeywords and the real password are collectively called sweet words. If an attacker steals the password file, she cannot tell the real one and probably (with 1−1/t probability) log in with a honeyword. Then, the server can detect the password leakage from the "wrong" login. The follow-up works focus on the honeyword generation algorithms [16], [17] so as to produce more plausible-looking decoys and the detection methods [18] to improve reliability.

PARAMETERS:



Parameter

- Security parameter κ.
- 2HashDH:
  1) m_1 is a primer number, G_1 is a m_1-order cyclic group, and g_1 is a generator of G_1. The length of m_1 is a polynomial function of κ.
  2) H_1, H_1' are two hash functions with ranges $\{0,1\}^{l_1}$ and G_1, respectively. The PRF $F_s(x)$ is $H_1(x, H_1'(x)^s)$. l_1 is a polynomial function of κ.
- HMQV:
  1) m_2 is a primer number, G_2 is a m_2-order cyclic group, g_2 is a generator of G_2. The length of m_2 is a polynomial function of κ.
  2) H_2, H_2' are two hash functions with ranges $\mathbb{Z}_{m_2}$ and $\{0,1\}^{l_2}$. l_2 is the length of the session key, which is a polynomial function of κ.
- A honey encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$ for the message space $\mathbb{Z}_{m_2}$.
- A honeyword generation algorithm Gen.

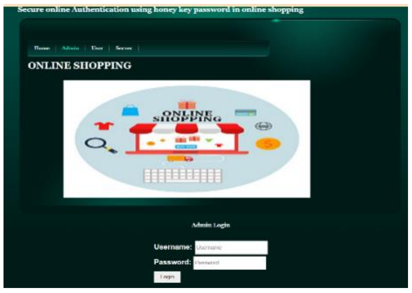
Initialization (via a secure channel)

- The client C picks $s \leftarrow\!\!\$\, \mathbb{Z}_{m_1}$ as the secret key of S in 2HashDH, and computes $rw \leftarrow H_1(pw, H_1'(pw)^s)$; computes $k_U \leftarrow\!\!\$\, \mathbb{Z}_{m_2}, K_U \leftarrow g_2^{k_U}$ to generate the private/public keys (k_U, K_U) for U in HMQV; computes $c \leftarrow \mathrm{Enc}_{rw}(k_U)$ to yield the ciphertext c of k_U using the key rw; sends (U, s, K_U, c) to S.
- Getting (U, s, K_U, c) from the client C, the authentication server S computes $k_S \leftarrow\!\!\$\, \mathbb{Z}_{m_2}, K_S \leftarrow g_2^{k_S}$ to generate its private/public keys (k_S, K_S) in HMQV; S generates $t-1$ honeywords $hw_i \leftarrow$ Gen for i from 1 to $t-1$, the corresponding random honeyword $rw_i \leftarrow H_1(pw, H_1'(hw_i)^s)$, and the honey private/public keys $k_{U,i} \leftarrow \mathrm{Dec}_{rw_i}(c)$, $K_{U,i} \leftarrow g_2^{k_{U,i}}$; randomly shuffles $K_{U,i}$ $(1 \leq i \leq t-1)$ with K_U, and sends the index i_r of the real one (with the ID U) to the honeychecker HC; stores s, c with the shuffled $K_{U,i}$ $(1 \leq i \leq t)$.
- Getting (U, i_r) from the authentication server S, the honeychecker HC stores it.

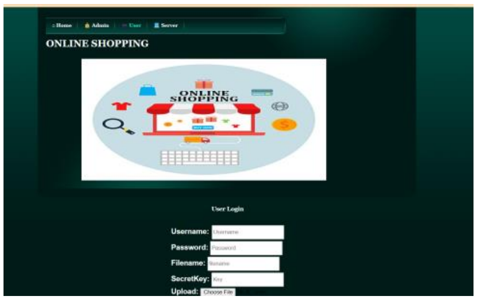
Authentication (via a server-authenticated channel)

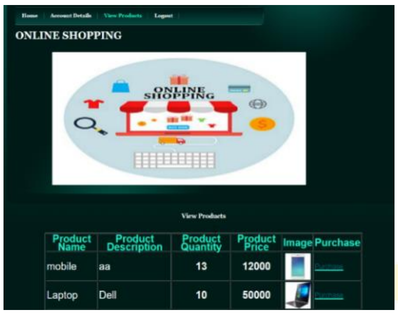
- C picks $r \leftarrow\!\!\$\, \mathbb{Z}_{m_1}$ and computes $\alpha \leftarrow H_1'(pw)^r$; picks $x \leftarrow\!\!\$\, \mathbb{Z}_{m_2}$ and computes $X \leftarrow g_2^x$; sends (U, X, α) to S.
- Getting (U, X, α) from C, S picks $y \leftarrow\!\!\$\, \mathbb{Z}_{m_2}$ and computes $Y \leftarrow g_2^y, \beta \leftarrow \alpha^s$; sends (Y, β, c, K_S) to C; computes $SK_i \leftarrow H_2((XK_{U,i}^{H_2'(X,K_S)})^{y+H_2'(Y,K_{U,i})k_S})$ for i from 1 to t.
- Getting (Y, β, c, K_S) from S, C computes $rw \leftarrow H_1(pw, \beta^{1/r})$, $k_U \leftarrow \mathrm{Dec}_{rw}(c)$; computes $SK \leftarrow H_2((YK_S^{H_2'(Y,K_U)})^{x+H_2'(X,K_S)k_U})$ and further uses SK for data transmission (or other purposes).
- S checks if the session key SK used by C is one of $\{SK_i\}_{i=1}^{t}$:
  1) If it is not, deny this session.
  2) If it is the i-th one, S sends (U, i) to HC (via a secure channel). Then HC checks if the index i is correct for U (i.e., equal to i_r):
     a) If it is, allow this session.
     b) Otherwise, raise an alarm of password leakage and take actions according to the pre-defined security policy.

IV. RESULTS AND DISCUSSION


Secure online Authentication using honey key password in online shopping
Home | Admin | User | Server
ONLINE SHOPPING
ONLINE SHOPPING
Admin Login
Username:
Password:
Login

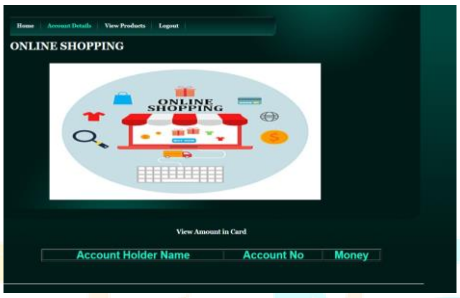
Secure online Authentication using honey key password in online shopping
Home | Admin | User | Server
ONLINE SHOPPING
ONLINE SHOPPING





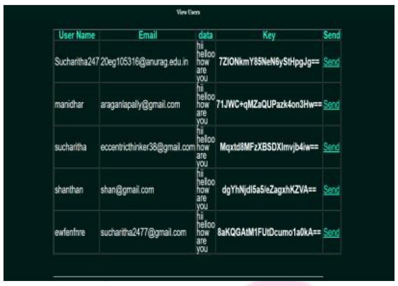
View Products

Product Name	Product Description	Product Quantity	Product Price	Image	Purchase
mobile	aa	13	12000		Purchase
Laptop	Dell	10	50000		Purchase



View Amount in Card

Account Holder Name	Account No	Money


View Users

User Name	Email	data	Key	Send
Sucharitha247	20eg105316@anurag.edu.in	hii helloo how are you	7ZIONkmY85NeN6yStHpgJg==	Send
manidhar	araganlapally@gmail.com	hii helloo how are you	71JWC+qMZaQUPazk4on3Hw==	Send
sucharitha	eccentricthinker38@gmail.com	hii helloo how are you	Mqxtd8MFzXBSDXImvjb4iw==	Send
shanthan	shan@gmail.com	hii helloo how are you	dgYhNjdl5a5/eZagxhKZVA==	Send
ewfenfnre	sucharitha2477@gmail.com	hii helloo how are you	8aKQGAtM1FUtDcumo1a0kA==	Send

CONCLUSION:

our proposal introduces HPAKE, a pioneering approach that combines the strengths of honeywords and aPAKE techniques to enhance password security. By leveraging OPAQUE, honeyword mechanism, and honey encryption, we construct a robust HPAKE framework that effectively detects password leakage from external attackers and prevents insider threats. Through rigorous analysis using a game-based security model, we formally prove the security of our design. Furthermore, our real-world implementation demonstrates the efficiency of HPAKE for practical applications. Looking ahead, we envision expanding HPAKE's utility by integrating it with UPI payments, thereby ensuring secure transactions between users and merchants. Additionally, we plan to enhance security measures within HPAKE, including implementing multi-factor authentication and developing a real-time alert system to promptly notify users of any suspicious activities related to their financial transactions. By continuously improving and adapting HPAKE, we aim to provide users with a reliable and secure solution for safeguarding their sensitive information in an ever-evolving digital landscape.

REFERENCES

[1] J. Bonneau, C. Herley, P. C. Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” in Proc. IEEE S&P 2012, pp. 553–567.

[2] N. Huaman, S. Amft, M. Oltrogge, Y. Acar, and S. Fahl, “They would do better if they worked together: The case of interaction problems between password managers and websites,” in Proc. IEEE S&P 2021, pp. 1367–1381.

[3] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, “Improving password guessing via representation learning,” in Proc. IEEE S&P 2021, pp. 265–282.

[4] W. Li and J. Zeng, “Leet usage and its effect on password security,” IEEE Trans. Inform. Foren. Secur., vol. 16, pp. 2130–2143, 2021.

[5] “Have i been pwned?” [Online]. Available: https://haveibeenpwned.com

[6] “Yahoo! data breaches.” [Online]. Available: https://en.wikipedia.org/w iki/Yahoo! data breaches

[7] “Yahoo tries to settle 3-billion-account data breach with $118 million payout.” [Online]. Available: https://arstechnica.com/tech-policy/2019/0 4/yahoo-tries-to-settle-3-billion-account-data-breach-with-118-million -payout/

[8] Z. Whittaker, “Github says bug exposed some plaintext passwords,” ht tps://www.zdnet.com/article/github-says-bug-exposed-account-passwor ds/, 2018.

[9] S. M. Bellovin and M. Merritt, “Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise,” in Proc. ACM CCS 1993, pp. 244–250.

[10] V. Boyko, P. MacKenzie, and S. Patel, “Provably secure passwordauthenticated key exchange using diffie-hellman,” in Proc. EUROCRYPT 2000. Springer, pp. 156–171.

[11] C. Gentry, P. MacKenzie, and Z. Ramzan, “A method for making password-based key exchange resilient to server compromise,” in Proc. CRYPTO 2006. Springer, pp. 142–159.

[12] S. Jarecki, H. Krawczyk, and J. Xu, “OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks,” in Proc. EUROCRYPT 2018. Springer, pp. 456–486.

[13] M. Abdalla, M. Barbosa, T. Bradley, S. Jarecki, J. Katz, and J. Xu, “Universally composable relaxed password authenticated key exchange,” in Proc. CRYPTO 2020. Springer, pp. 278–307.

[14] S. Smyshlyaev, N. Sullivan, and A. Melnikov, “[cfrg] results of the PAKE selection process,” 2020. [Online]. Available: https://mailarchiv e.ietf.org/arch/msg/cfrg/LKbwodpa5yXo6VuNDU66vt Aca8/

[15] A. Juels and R. L. Rivest, “Honeywords: Making password-cracking detectable,” in Proc. ACM CCS 2013, pp. 145–160.

[16] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, “A security analysis of honeywords,” in Proc. NDSS 2018, pp. 1–18.

[17] Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, “Generation of secure and reliable honeywords, preventing false detection,” IEEE Trans. Depend. Secur. Comput., vol. 16, no. 5, pp. 757–769, 2019.

[18] K. C. Wang and M. K. Reiter, "Using amnesia to detect credential database breaches," in Proc. USENIX Secur, 2021, pp. 839-855.

[19] Passwordless Authentication Duo Security. Accessed: Aug. 15, 2021. [Online]. Available: https://duo.com/solutions/passwordless

[20] W. Li, H. Cheng, P. Wang, and K. Liang. "Practical threshold multi- factor authentication," IEEE Trans. Inf. Forensics Security, vol. 16.pp. 3573-3588, 2021.

[21] J. Zhang, H. Zhong, J. Cui, Y. Xu, and L. Liu, "SMAKA: Secure many-to-many authentication and key agreement scheme for vehicular networks." IEEE Trans. Inf. Forensics Security, vol. 16, pp. 1810-1824. 2021.

[22] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial Internet of Things." IEEE Trans. Depend. Secure Comput.. vol. 17, no. 6, pp. 1133-1146, Nov. 2020.

[23] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "Cloud centric authentication for wearable healthcare monitoring system," IEEE Trans. Depend. Secure Comput., vol. 17, no. 5, pp. 942-956, Sep. 2020.

[24] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in Proc. IEEE Symp. Secur. Privacy, May 2014, pp. 538-552.

[25] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," in Proc. EUROCRYPT Cham, Switzerland: Springer, 2014, pp. 293-310.

[26] Github CFRG/Draft-IRTF-CFRG-Opaque/the Opaque Asymmetric Pake Protocol. Accessed: Aug. 15, 2021. [Online]. Available: https://github.com/cfrg/draft-irtf-cfrg-opaque

[27] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password- based protocols secure against dictionary attacks," in Proc. IEEE Com- put. Soc. Symp. Res. Secur. Privacy, 1992, pp. 72-84

[28] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu. "Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online)," in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), Mar. 2016, pp. 276-291.

[29] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in Proc. CRYPTO, 2005, pp. 546-566

[30] T. Wu, "The secure remote password protocol," in Proc. NDSS, vol. 98. 1998. pp. 97-111.

[31] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin, Using the Secure Remote Password (SRP) Protocol for TLS Authentication, document RFC5054, 2007.

[32] H. Cheng. Z. Zheng, W. Li, P. Wang, and C.-H. Chu, "Probability model transforming encoders against encoding attacks," in Proc. USENIX Secur, 2019, pp. 1573-1590.