IBE Encrypted Password-Authenticated Key Exchange Protocols

Sruthy K V, S.Soundharya, N.Sudha
Dept of CSE
CMS COLLEGE OF ENGINEERING AND TECHNOLOGY

***Abstract**- In two-server password-authenticated key exchange (PAKE) protocol, a client splits its password and stores two shares of its password in the two servers, respectively, and the two servers then cooperate to authenticate the client without knowing the password of the client. In case one server is compromised by an adversary, the password of the client is required to remain secure. In this paper, we present two compilers that transform any two-party PAKE protocol to a two-server PAKE protocol on the basis of the identity-based cryptography, called ID2S PAKE protocol. By the compilers, we can construct ID2S PAKE protocols which achieve implicit authentication. As long as the underlying two-party PAKE protocol and identity-based encryption or signature scheme have provable security without random oracles, the ID2S PAKE protocols constructed by the compilers can be proven to be secure without random oracles.And also when we upload a file we use IBE encryption is used dwhen download the file decrypt it,do Compared with the Katz et al.'s two-server PAKE protocol with provable security without random oracles, our ID2S PAKE protocol can save from 22% to 66% of computation in each server.*

Keywords- IBE, password-authenticated key exchange

I. INTRODUCTION

To secure communications between two parties, an authenticated encryption key is required to agree on in advance. So far, two models have existed for authenticated key exchange. One model assumes that two parties already share some cryptographically-strong information: either a secret key which can be used for encryption/authentication of messages, or a public key which can be used for encryption/ signing of messages. These keys are random and hard to remember. In practice, a user often keeps his keys in a personal device protected by a password/PIN. Another model assumes that users, without help of personal devices, are only capable of storing "human-memorable" password. Here added IBE encryption and decryption added as enhancement while doing upload and download files.

In an attribute-based encryption (ABE) scheme, if the attributes of users satisfy the access policy (also called access structure) which is decided by other users, then they can decrypt the cipher text. The first ABE scheme was proposed by Sahai and Waters, which is an extended concept from identity-based encryption (IBE). In such a scheme, an encryption can send the cipher text to many users by indicating the attributes about the expected receivers, and those users who possess the attributes matching the attributes assigned by the encryption can successfully decrypt the cipher text.

II. RELATED WORK

Bellovin and Merritt were the first to introduce password-based authenticated key exchange (PAKE), where two parties, based only on their knowledge of a password, establish a cryptographic key by exchange of messages. A PAKE protocol has to be immune to on-line and off-line dictionary attacks. In an off-line dictionary attack, an adversary exhaustively tries all possible passwords in a dictionary in order to determine the password of the client on the basis of the exchanged messages. In on-line dictionary attack, an adversary simply attempts to login repeatedly, trying each possible password. By cryptographic means only, none of PAKE protocols can prevent on-line dictionary attacks. But on-line attacks can be stopped simply by setting a threshold to the number of login failures. Since Bellovin and Merritt introduced the idea of PAKE, numerous PAKE protocols have been proposed. In general, there exist two kinds of PAKE settings, one assumes that the password of the client is stored in a single server and another assumes that the password of the client is distributed in multiple servers. PAKE protocols in the single-server setting can be classified into three categories as follows.

Password-only PAKE: Typical examples are the "encrypted key exchange" (EKE) protocols given by Bellovin and Merritt [4], where two parties, who share a password, exchange messages encrypted by the password, and establish a common secret key. The formal model of security for PAKE was firstly given in [3], [8]. Based on the security model, PAKE protocols have been proposed and **proved** to be secure. **PKI-based PAKE**: PKI-based PAKE protocol was first given by Gong et al. [17], where the client stores the server's public key in addition to share a password with the server. Halevi and Krawczyk [18] were the first to provide formal definitions and rigorous proofs of security for PKI-based PAKE.

ID-based PAKE: ID-based PAKE protocols were proposed by Yi et al. [32], [33], where the client needs to remember a password in addition to the identity of the server, whereas the server keeps the password in addition to a private key related to its identity. ID-based PAKE can be thought as a trade-off between password-only and PKI-based PAKE. In the single-server setting, all the passwords necessary to authenticate clients are stored in a single server. If the server is compromised, due to, for example, hacking or even insider attacks, passwords stored in the server are all disclosed. This is also true to Kerberos, where a user authenticates against the authentication server with his username and password and obtains a token to authenticate against the service server. To address this problem, the multi-server setting for PAKE, where the password of the client is distributed in n servers. PAKE protocols in the multi-server setting can be classified into two categories as follows

Threshold PAKE: The first PKI-based threshold PAKE protocol was given by Ford and Kaliski , where n severs, sharing the password of the client, cooperate to authenticate the client and establish independent session keys with the Client. As long as n - 1 or fewer servers are compromised, their protocol remains secure. Jablon gave a protocol with similar functionality in the password-only setting. MacKenzie et al. proposed a PKI-based threshold PAKE protocol which requires only t out of n servers to cooperate in order to authenticate the client. Their protocol remains secure as long as t - 1 or fewer servers are compromised. Di Raimondo and Gennaro suggested a password-only threshold PAKE protocol which requires fewer than 1/3 of the servers to be compromised.

Two-server PAKE: Two-server PKI-based PAKE was first given by Brainard , where two servers cooperate to authenticate the client and the password remains secure if one server is compromised. A variant of the protocol was later proved to be secure. A two-server password-only PAKE protocol was given by Katz , in which two servers symmetrically contribute to the authentication of the client. The protocol in the server side can run in parallel. Efficient protocols were later proposed, where the front-end server authenticates the client with the help of the back-end server and only the front-end server establishes a session key with the client. These protocols are asymmetric in the server side and have to run in sequence. Yi et al. gave a symmetric solution which is even more efficient than asymmetric protocols. Recently, Yi et al. constructed an ID2S PAKE protocol with the identity-based encryption scheme (IBE). In this paper, we will consider the two-server setting for PAKE only. In two-server PAKE, a client splits its password and stores two shares of its password in the two servers, respectively, and the two servers then cooperate to authenticate the client without knowing the password of the client. Even if one server is compromised, the attacker is still unable to pretend any client to authenticate against another server. A typical example is the two-server PAKE protocol given by Katz et al. [23], which is built upon the two-party PAKE protocol (i.e., the KOY protocol [22]), where two parties, who share a password, exchange messages to establish a Common secret key. Their basic two-server protocol is secure against a passive (i.e., “honest-but-curious”) adversary who has access to one of the servers throughout the protocol execution, but cannot cause this server to deviate from its prescribed behaviour. In , Katz et al. also showed how to modify their basic protocol so as to achieve security against an active adversary who may cause a corrupted server to deviate arbitrarily from the protocol. The core of their protocol is the KOY protocol. The client looks like running two KOY protocols with two servers in parallel. However, each server must perform a total of roughly 80 exponentiations (i.e., each server’s work is increased by a factor of roughly 6 as compared to the basic protocol).

III. PROPOSED METHOD

In this paper, we propose a new compiler for ID2S PAKE protocol based on any identity-based signature scheme (IBS), such as the Paterson et al.’s scheme.The basic idea is: The client splits its password into two shares and each server keeps one share of the password in addition to a private key related to its identity for signing.

In key exchange, each server sends the client its public key for encryption with its identity-based signature on it. The signature can be verified by the client on the basis of the identity of the server. If the signature is genuine, the client submits to the server one share of the password encrypted with the public key of the server. With the decryption keys, both servers can derive the same one-time password, by which the two servers can run a two-party PAKE protocol to authenticate the client. We have implemented our ID2S PAKE protocols; it shows that our protocols save from 22% to 66% of computation in each server, compared with the Katz et al.’s protocol.The server performance is critical to the performance of the whole protocol when the servers provide services to a great number of clients concurrently. Our Protocol shows that less than one second is needed for the client to execute our protocols. In the real world, a protocol determines how users behave in response to input from their environments. In the formal model, these inputs are provided by the adversary. Each user is assumed to be able to execute the protocol multiple times (possibly concurrently) with different partners.This is modelled by allowing each user to have unlimited number of instances with which to execute the protocol.

System Initialization

The most creative and challenging phase of the life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementations of the candidate system. The design may be defined as “the process of applying various techniques and principles for the purpose of defining a device, a process or a system with sufficient details to permit its physical realization”. The designer’s goal is how the output is to be produced and in what format. Samples of the output and input are also presented. Second input data and database files have to be designed to meet the requirements of the proposed output. The processing phases are handled through the program Construction and Testing. Finally, details related to justification of the system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step toward implementation. The importance of software design can be stated in a single word “Quality”. Design provides us with representations of software that can be assessed for quality. Design is the only way where we can accurately translate a customer’s requirements into a complete software product or system. Without design we risk building an unstable system that might fail if small changes are made. It may as well be difficult to test, or could be one who’s quality can’t be tested. So it is an essential phase in the development of a software product.

Design process

The design phase focuses on the detailed implementation of the system recommended in the feasibility study. The design phase is a transition from a user-oriented document to document oriented to the programmers or database personnel. System design goes through to phase of development:

- Logical Design
- Physical Design

The dataflow diagram shows the logical flow of the system and defines the boundaries of the system. For a candidate system, it describe the inputs(source), output(destination), database(file) and procedures(dataflow), all in a format that meets the users requirement in logical design we specifies the user’s needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources.

Following logical design is physical design. This produces the working system by defining specification that tell programmers exactly what the candidate system must do, in turn we write the necessary programs or modifies the software package that accept input from the user, then perform the necessary operation through logical system design is one important phase of system design, the dataflow diagram is the logical flow of a system and defines the boundaries of the system, for a candidate system it describes the inputs or source, outputs or destination, database or data stores and procedures all in a format that meets the user needs. When analysts prepare the logical system design, they specify the user needs at level of detail that virtually. It determines the information flow into and out of the system and the required data resources. The logical system design covers: Reviews the current physical system its dataflow, file content, volumes, frequency etc.

Preparing output specification that determines the format, content and frequency of reports including terminal specification and location. Prepares input specification format, content and the most if the input functions. This includes determining the flow of the document from the input data source to the detailed output location. Prepares edit security and control specification , this includes specifying the rules for edit correction backup procedures and the controls that ensure processing file integrity specifies the implementation plan. Prepare the logical design walks through the information flow output, input and controls and implementation plan reviews benefits, costs, target rates and system constraints the existing file and procedure reports.

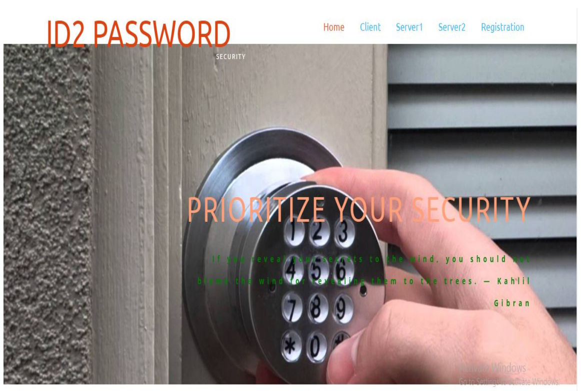
System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. For any software that is newly developed, primary importance is given to testing the system .It is the last opportunity for the developer over to the customers. Testing is the process by which a developer will generate a set of test data, which gives maximum probability of finding all types of errors that can occur in the software. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests: online response, volume, stress, recovery & security and usability tests. A series of testing are performed for the proposed system before the system is ready for user acceptance testing.

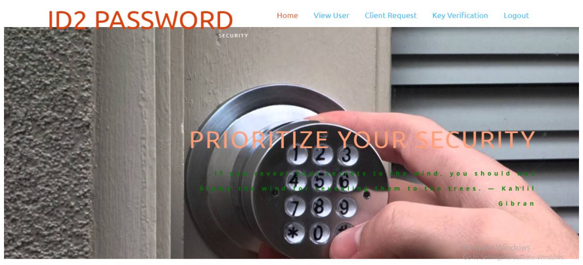
It is the process of exercising or evaluating a system by manual or automatic means to verify that it satisfies the specified requirements or to identify the difference between expected and actual results. The testing activities are aimed at convincing the customer through demonstration and actual use that the software is a solution to the original problem and that both the product and the process that created it are of high quality. It is also used to find and eliminate any residual errors from previous stages and the operational reliability of the system.

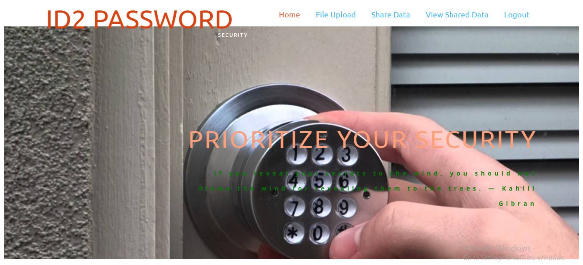
Preparation of Test Data

Software testing is a crucial element of software quality assurance and represents the ultimate review of specification, design and coding. Testing represents an interesting anomaly for the software. During earlier definition and development phases, it was attempted to build software from abstract concepts to tangible implementation. The testing responsible for ensure that the product that has built performs the way that the detailed design documentation specifies.The main purpose of testing an information system is to find the errors and correct them. The scope of system testing should include both manual and computerized operations. System testing is comprehensive evaluation of the programs, manual procedures, computer operations and controls. System testing is the process of checking whether the developed system is working according to the objective and requirement. All testing is to be conducted in accordance to the test conditions specified earlier. This will ensure that the test coverage meets the requirements and that testing is done in a systematic manner.

IV. RESULTS







.

V. CONCLUSION

In this paper, we present two efficient compilers to transform any two-party PAKE protocol to an ID2S PAKE protocol with identity-based cryptography. In addition, we have provided a rigorous proof of security for our compilers without random oracle. Our compilers are in particular suitable for the applications of password-based authentication where an identity-based system has already established. Our future work is to construct an identity-based multiple server PAKE protocol with any two-party PAKE protocol.

REFERENCES

[1] Xun Yi, Fang-Yu Rao, Zahir Tari, Feng Hao, Elisa Bertino, Ibrahim Khalil and Albert Y. Zomaya, “ID2S Password-Authenticated Key Exchange Protocols”, **IEEE Transactions on Computers, 2016.**

[2] M. Abdullah and D. Point cheval. Simple password-based encrypted key exchange protocols. In Proc. CT-RSA 2005, pages 191-208, 2005.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In Proc. Eurocrypt’00, pages 139-155, 2000.

[4] S. M. Bellovin and M. Merritt. Encrypted key exchange: Passwordbased protocol secure against dictionary attack. In Proc. 1992 IEEE Symposium on Research in Security and Privacy, pages 72-84, 1992.

[5] J. Bender, M. Fischlin, and D. Kugler. Security analysis of the PACE key-agreement protocol. In Proc. ISC’09, pages 33-48, 2009.

[6] J. Bender, M. Fischlin, and D. Kugler. The PACEjCA protocol for machine readable travel documents. In INTRUST’13, pages 17-35, 2013.

[7] D. Boneh and M. Franklin. Identity based encryption from theWeil pairing. In Proc. Crypto’01, pages 213-229, 2001.

[8] V. Boyko, P. Mackenzie, and S. Patel. Provably secure passwordauthenticated key exchange using Diffie-Hellman. In Proc. Eurocrypt’ 00, pages 156-171, 2000.

[9] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo. Nightingale: A new two-server approach for authentication with short secrets. InProc. 12th USENIX Security Symp., pages 201-213, 2003.

[10] E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In Proc. CCS’03, pages 241-25