AIJREAS VOLUME 3, ISSUE 3(2018, MAR) (ISSN-2455-6300) ONLINE
ANVESHANA'S INTERNATIONAL JOURNAL OF RESEARCH IN ENGINEERING AND APPLIED SCIENCES

IDENTITY BASED (ID2S) AUTHENTICATED EXCHANGE PROTOCOLS

E. SUNIL
Assistant Professor, Dept. of CSE,
Holy Mary Institute of Technology and Science, Hyderabad

ABSTRACT:
In the two-server password-authenticated key exchange (PAKE) protocol, a sender splits its password and stores two shares of its identification in the two servers, respectively, and the two servers then forward to authenticate the client without remembering the password of the client. In case one server is discredited by an opponent, the password of the user is required to endure secure. In this paper, we introduce two compilers that convert any two-party PAKE protocol to a two-server PAKE protocol by the identity-based cryptography, called ID2S PAKE protocol. By the compilers, we can construct ID2S PAKE protocols which obtain understood authentication. As long as the underlying two-party PAKE protocol and identity-based encryption or signature scheme have provable security without casual oracles, the ID2S PAKE protocols constructed by the compilers can be proven to be secure without random oracles. 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.

***Index Terms**: Password-authenticated key exchange, identity-based encryption and signature, Diffie-Hellman key exchange, decisional Diffie-Hellman problem*

1. INTRODUCTION

To secure conversations between two parties, an authenticated encryption key is required to agree on in advance. So far, two prototypes 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 managed for encryption/authentication of messages or a public key which can be used for encryption/signing of words. These keys are casual 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 the help of own devices, are only capable of saving "human-memorable" passwords.

Bellovin and Merritt [4] 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 to determine the password of the client by the exchanged messages. In the on-line dictionary attack, an adversary simply attempts to log in 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 merely by setting a threshold to the number of login failures.

Since Bellovin and Merritt [4] 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 believes 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, which share a password, interchange 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, PAKEprotocols [1], [2], [5], [10], [11], [16], [20], [22] have been intended and proved to be secure.

PKI-based PAKE: PKI-based PAKE protocol was beginning given by Gong et al. [17], where the client stores the server's public key, also, to sharing 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 identify a password in addition to the identity of the server, whereas the server keeps the password in increasing to a private key related to its status. ID-based PAKE can be considered as a trade-off between password-only and PKI-based PAKE.

In the single-server setting, all the passwords unavoidable to authenticate clients are stored on 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 [12], where a user authenticates against the authentication server with his username and password and obtains a token to validate against the service server.

To address this problem, the multi-server setting for PAKE was first suggested in [15], [19], 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 [15], where n severs, sharing the password of the client, cooperate to authenticate the client and establish independent gathering keys with the client. As long as $n - 1$ or fewer servers are depreciated, their protocol remains secure. Jablon [19] 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 to authenticate the client. Their protocol remains secure as long as $t - 1$ or some servers are compromised. Di Raimondo and Gennaro [26] 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 [9], where two servers combine 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 in [27]. A two-server password-only PAKE protocol was given by Katz et al. [23], in which two servers symmetrically contribute to the authentication of the client. The protocol in the server side can run in parallel. Effective protocols [21], [29], [30], [31] 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 secures a session key with the client. These protocols are asymmetric in the server side and have to run in sequence. Yi et al. gave asymmetric resolution [34] which is even more efficient than asymmetric protocols [21], [29], [30], [31]. Recently, Yi et al.

constructed an ID2S PAKE protocol with the identity-based encryption system (IBE) [35].

In this paper, we will consider the two-server setting for PAKE only. In two-server PAKE, a client divides 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. [3], which is built upon the two-party PAKE protocol (i.e., the KOY protocol [2]), 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 [3], Katz et al. also showed how to modify their basic protocol 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 approximately six as compared to the underlying protocol [3]).

In [5], a security model for ID2S PAKE protocol was given, and a compiler that transforms any two-party PAKE protocol to an ID2S PAKE protocol was proposed by the Cramer-Shoup public key encryption scheme [13] and any identity-based encryption scheme, such as the Waters' scheme [8].

2. DEFINITIONS

A formal model of security for two-server PAKE was given by Katz et al. [3] (based on the MacKenzie et al.'s model for PKI-based PAKE [4]). Boneh and Franklin [7] defined chosen ciphertext security for IBE under chosen identity attack. Combining the two models, a model for ID2S PAKE protocol was given in [35] and can be described as follows.

Participants, Initialization and Passwords. An ID2S PAKE protocol involves three kinds of protocol participants: (1) A set of clients (denoted as Client), each of which requests services from servers on the network; (2) A set of servers (denoted as Server), each of which provides services to clients on the network; (3) A group of Private Key Generators (PKGs), which generate public parameters and corresponding private keys for servers.

We assume that Client Server Triple is the set of triples of the client and two servers, where the client is authorized to use services provided by the two servers, Client T Server = ∅, User = Client S Server, any PKG 6∈ User, and ClientServerTriple ⊆ Client × Server × Server.

Before any execution of the protocol, we assume that an initialization phase occurs. During initialization, the PKGs cooperate to generate public parameters for the contract, which are available to all participants, and private keys for servers, which are given to the appropriate servers. The user may keep the public parameter in a personal device, such as a smart card or a USB flash drive. When the PKGs generate the private key for a server, each PKG generates and sends a private key component to the server via a secure channel. The server then derives its private key by combining all individual key components from all PKGs. We assume that at least one of PKGs is honest to follow the protocol. Therefore, the private key of the server is known to the server only.

For any triple (C, A, B) ∈ ClientServerTriple, we assume that the client C chooses its password PwC independently and uniformly at random from a "dictionary" D = {pw1, pw2, · · · , pwN } of size N, where D ⊂ Zq, N is a fixed constant which is independent of any security parameter, and q is a large prime. The password is then split into two shares PwC, A and PwC,B and stored at the two servers A and B, respectively, for authentication. We assume that the two servers never collude to determine the password of the client. The client C needs to remember pwC to log into the servers A and B.

For simplicity, we assume that each client C shares its password PwC with exactly two servers A and B. In this case, we say that servers A and B are associated with C. A server may be associated with multiple clients.

Execution of the Protocol: 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 modeled by allowing each user to have unlimited number of instances (please refer to [3]) with which to execute the protocol. We denote instance i of user U as Ui. A given instance may be used only once. The adversary is given oracle access to these different instances. Furthermore, each instance maintains (local) state which is updated during the course of the experiment. In particular, each instance Ui is associated with the following variables, initialized as NULL or FALSE (as appropriate) during the initialization phase.

— sidi U , pidi U and ski U are variables containing the session identity, partner identity, and session key for an instance Ui , respectively. Computation of the session key is, of course, the ultimate goal of the protocol. The session identity is simply a way to keep track of the different executions of a particular user U. Without loss of generality, we simply let this be the (ordered) concatenation of all messages sent and received by instance Ui. The partner identity denotes the identity of the user with whom Ui believes it is interacting. For a client C, ski C consists of a pair (ski C, A, ski C, B), which are the two keys shared with servers A and B, respectively.

— Acci U and termi U are boolean variables denoting whether a given instance UI has been accepted or terminated, respectively. Termination means that the given instance has done receiving and sending messages, acceptance indicates successful completion. In our case, acceptance means that the instance is sure that it has established a session key with its intended partner; thus, when an instance UI has been accepted.

3 ID2S PAKE PROTOCOLS

In this section, we present two compilers transforming any two-party PAKE protocol P to an ID2S PAKE protocol P0 with identity-based cryptography. The first compiler is built on identity-based signature (IBS) and the second compiler is based on identity-based encryption (IBE).

3.1 ID2S PAKE Based on IBS

3.1.1 Protocol Description

We need an identity-based signature scheme (IBS) as our cryptographic building block. A high-level description of our compiler is given in Fig. 1, in which the client C and two servers A and B establish two authenticated keys, respectively. If we remove authentication elements from our compiler, our key exchange protocol is essentially the Diffie-Hellman key exchange protocol [14]. We present the protocol by describing initialization and execution.

Initialization: Given a security parameter k ∈ N (the set of all natural number), the initialization includes:

Parameter Generation: On input k, (1) m PKGs cooperate to run SetupP of the two-party PAKE protocol P to generate system parameters, denoted as paramsP. (2) m PKGs cooperate to run SetupIBS of the IBS scheme to generate public system parameters for the IBS scheme, denoted as paramsIBS (including a subgroup G of the additive group of points of an elliptic curve), and the secret master-keyIBS. (3) m PKGs choose a public key encryption scheme E, e.g., [13], whose plaintext group is a large cyclic group G with a prime order q and a generator g and select two

hash functions, H1 : {0, 1} ∗ → Z ∗ n, where n is the order of G, and H2 : {0, 1} ∗ → Z∗q, from a collision-resistant hash family. The public system parameters for the protocol P0 is params = params, IBS,E S{(G, q, g),(H1, H2)} and the secret master-key IBS is secretly shared by the PKGs in a manner that any coalition of PKGs cannot determine master-key IBS as long as one of the PKGs is honest to follow the protocol.

Remark: Taking the Paterson-Schuldt IBS scheme [25] for example, m PKGs agree on randomly chosen G, G2 ∈ G and each PKG randomly chooses αi ∈ Zp and broadcast Gαi with a zero-knowledge proof of knowing αi and a signature. Then we can set G1 = GPi αi as the public master key and the secret master-key IBS = GPi αi2. The secret master key is privately shared among m PKGs and unknown to anyone even if m − 1 PKGs maliciously collude.

Key Generation: On input the identity S of a server S ∈ Server, paramsIBS, and the secret sharing master-key IBS, PKGs cooperate to run extracts of the IBS scheme and generate a private (signing) key for S, denoted as dS, in a manner that any coalition of PKGs cannot determine dS as long as one of the PKGs is honest to follow the protocol. Remark. In the Paterson-Schuldt IBS scheme with m PKGs, each PKG computes one component of the private key for a server S, i.e., (Gαi2 H(S) or, Gri), where H is the Waters' hash function and sends it to the server via a secure channel. Combining all components, the server can construct its private key dS = (GPi αi2 H(S)Piri Piri), which is known to the server only even if m−1 PKGs maliciously collude.

3.2 ID2S PAKE Based on IBE

3.2.1 Protocol Description

A high-level description of our compiler based on identity-based encryption (IBE) is given in Fig. 2. We present the protocol by describing initialization and execution.

Initialization: Given a security parameter k ∈ N, the initialization includes:

Parameter Generation: On input k, (1) m PKGs cooperate to run SetupP of the two-party PAKE protocol P to generate system parameters, denoted as paramsP. (2) m PKGs cooperate to run SetupIBE of the IBE scheme to generate public system parameters for the IBE scheme, denoted as params IBE, and the secret master-key. Assume that G is a generator of IBE plaintext group G with an order n. (3) M PKGs choose a public key encryption scheme E, e.g., [13], whose plaintext group is a large cyclic group G with a prime order q and a generator g and select two hash functions, H1 : {0, 1} ∗ →Z∗n and H2 : {0, 1} ∗ → Z∗q, from a collision-resistant hash family. The public system parameters for the protocol P0 is params = params, IBE, E S{(G, G, n),(G, q, g),(H1, H2)} and the secret master-key is secretly shared by the PKGs in a manner that any coalition of PKGs cannot determine master-key as long as one of the PKGs is honest to follow the protocol.

4. PROOF OF SECURITY

Based on the security model defined in Section 2, we provide a rigorous proof of security for our compilers in this section.

4.1 Security of ID2S PAKE Protocol Based on IBE

Theorem 2. Assuming that (1) the identity-based encryption (IBE) scheme is secure against the chosen-ciphertext attack; (2) the public key encryption scheme E is protected against the chosen-ciphertext attack; (3) the decisional Diffie-Hellman problem is hard over (G, g, q); (4) the protocol P is a secure two-party PAKE protocol with explicit authentication; (5) H1, H2 are collision-resistant hash functions, then the protocol P0 illustrated in Fig. 2 is a secure ID2S PAKE protocol according to Definition 1.

Proof. Given an adversary A attacking the protocol, a simulator S runs the protocol for A. First of all, the simulator S initializes the system by generating params = paramsP, IBE, E S{(G, G, n),(G, q, g),(H1, H2)} and the secret master-key IBE. Next, Client, Server, and Client Server Triple sets are determined. Passwords for clients are chosen at random and split, and then stored at corresponding servers. Private keys for servers are computed using master-key IBE. The public information is provided to the adversary. Considering (C, A, B) ∈ Clinet Server Triple, we assume that the adversary A chooses the server B to corrupt and the simulator S gives the adversary A the information held by the corrupted server B, including the private key of the server B, i.e., dB, and one share of the password of the client C, G pwC, B and gpw∗ C,B . After computing the appropriate answer to any oracle query, the simulator S provides the adversary A with the internal state of the corrupted server B involved in the query.

We view the adversary's queries to its Send oracles as queries to four different oracles as follows:

— Send(C, i, A, B) represents a request for instance Ci of client C to initiate the protocol. The output of this query is msg1 = hC, Wc, pk, Eai and msg2 = hC, Wc, pk, Ebi.

— Send(A, j, C, msg1) represents sending message msg1 to instance Aj of the server A. The output of this query is either msgA = hA, Wa, E1i or ⊥.

— Send(C, i, A, B, msgA|msgB) represents sending the message msgA|msgB to instance Ci of the client C. The output is either acci C = TRUE or ⊥.

— SendP (A, j, B, M) represents sending message M to instance Aj of the server A, supposedly by the server B, in the two-party PAKE protocol P. The input and output of this query depends on the protocol P.

We refer to the real execution of the experiment, as described above, as P0.

5. PERFORMANCE ANALYSIS

The efficiency of the compiled protocols using our compilers depends on performance of the underlying protocols. In our IBS-based protocol, if we use the KOY two- party PAKE protocol [2], the Paterson et al.'s IBS scheme [5] and the Cramer-Shoup public key encryption scheme [13] as cryptographic building blocks, the performance of our IBS-based protocol can be shown in TABLE 1. In our IBE-based protocol, if we use the KOY two-party PAKE protocol [12], the Waters IBE scheme [8] and the Cramer Shoup public key encryption scheme [13] as cryptographic building blocks, the performance of our IBE-based protocol can also be shown in TABLE 1. Also, we compare our protocols with the Katz et al. two-server PAKE protocol (secure against active adversary).

In Exp.,exp. Sign. And Pair of computation represent the computation complexities of modular exponentiation over an elliptic curve, modular exponentiation over Zp, a signature generation and a pairing, respectively, and Exp., exp. And Sign. in communication denotes the size of the modulus and the size of the signature, and KOY stands for the computation or communication complexity of the KOY protocol.

In Different operations are computed in different protocols. For example, some modular exponentiations in our protocols are over an elliptic curve group, while the modular exponentiations in the Katz et al.'s protocol are over Zp only. Our protocols need to compute pairings while the Katz et al.'s protocol does not. To further compare their performance, we implement our two protocols.

To realize the modular exponentiation Gx over an elliptic curve group G and the pairing map e: $G \times G \rightarrow GT$ in our protocols, we build our implementation on top of the PBC pairing-based cryptography library1, whereas the multiplicative group over the prime integer p is based on the GNU MP library2. Moreover, the elliptic curve we use is the

A512 ECC in which the first two groups are the same, i.e., asymmetric pairing. Another library embed TLS3 is adopted due to the invocations of AES and SHA-512 for the one time signature in KOY. All the experiments were conducted in Ubuntu 14.04 running on a computer equipped with an Intel i7-4770HQ CPU and 16 GBytes of memory. When implementing our protocols, we also performed optimization when applicable. For example, we compute the Waters' hash function by parallel computation.

The execution time of our two protocols compared with the Katz et al.'s protocol. From we can see that the client performance in Katz et al.'s protocol is better than our protocols, but the execution times for client in the three protocols are all less than 10 ms. The server performance in our protocols is better than the Katz et al.'s protocol, saving from 22% to 66% of computation. When the servers provide services to a great number of clients concurrently, the server performance is critical to the performance of the whole protocol. For example, assume that Servers A and B provide services to 100 clients concurrently and there is no communication delay, the longest waiting time with respect to a client for our IBE based protocol is around 7.08+208+176=391.08 ms while the Katz et al.'s protocol takes about 1.26+531+531=1,063.26 ms. The difference is 672.18 ms. In terms of communication complexity, the size of a group element over elliptic curve (denoted as Exp.) in our protocols can be 512 bits, while the size of a group element over Zp in the Katz et al.'s protocol [23] has to be 1024 bits. we can see that the communication complexity of our protocols is about a half of the Katz et al.'s protocol.

6. 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] M. Abdalla, P. A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In Proc. PKC'05, pages 65-84, 2005.

1. https://crypto.stanford.edu/pbc/download.html

2. https://gmplib.org/

3. https://tls.mbed.org/

[2]M. Abdalla and D. Pointcheval. 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: Password-based protocol secure against the 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 PACE|CA protocol for machine-readable travel documents. In INTRUST'13, pages 17-35, 2013.

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

[8] V. Boyko, P. Mackenzie, and S. Patel. Provably secure password authenticated key exchange using Diffie-Hellman. In Proc. Euro- crypt'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. In Proc.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 241-250, 2003.

[11] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on an encrypted key exchange. In Proc. PKC'04, pages 145-158, 2004.

[12] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. IEEE Communications, 32 (9):33-38, 1994.

[13] R. Cramer and V. Shoup. A practical public key cryptosystem was provably secure against adaptively chosen ciphertext attack. In Proc.Crypto'98, pages 13-25, 1998.

[14] W. Diffie and M. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, 32(2): 644-654, 1976.

[15] W. Ford and B. S. Kaliski. Server-assisted generation of a strong secret from a password. In Proc. 5th IEEE Intl. Workshop on Enterprise Security, 2000.