Internet Engineering Task Force
INTERNET-DRAFT
Intended Status: Standards Track
Expires: November 4, 2013

T.Kwon, Sejong University
H. Yoon, Samsung SDS
S. Kim, Samsung SDS
May 3, 2013

# I-PAKE: Identity-Based Password Authenticated Key Exchange
## draft-kwon-yoon-kim-ipake-01

Abstract

Although password authentication is the most widespread user authentication method today, cryptographic protocols for mutual authentication and key agreement, i.e., password authenticated key exchange (PAKE), in particular authenticated key exchange (AKE) based on a password only, are not actively used in the real world. This document introduces a quite novel form of PAKE protocols that employ a particular concept of ID-based encryption (IBE). The resulting cryptographic protocol is the ID-based password authenticated key exchange (I-PAKE) protocol which is a secure and efficient PAKE protocol in both soft- and hard-augmented models. I-PAKE achieves the security goals of AKE, PAKE, and hard-augmented PAKE. I-PAKE also achieves the great efficiency by allowing the whole pre-computation of the ephemeral Diffie-Hellman public keys by both server and client.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

## 1 Introduction

The most widespread user authentication method today is obviously password-based authentication; a user memorizes a textual and/or numerical password and makes its direct entry for authentication.

There have been various attempts to enhance security of password-based authentication. Password authenticated key exchange (PAKE) is a cryptographic protocol to achieve both mutual authentication and secure key agreement based on a password only, i.e., without requiring a public key certificate [IEEE P1363.2]. That is, PAKE is an authenticated key exchange (AKE) protocol based on the user-memorable low-entropy password, so that both authenticated entities can only agree on a fresh session key. Augmented PAKE protocols allow a server to store a one-way function of password instead of the plain text, so as to mitigate the server compromise [RFC2945].

PAKE protocols, however, have not been deployed actively despite of their merits. Instead, for example, Web applications employ the SSL/TLS suite for secure transport of password information at the cost of manipulating and relying on server's public key certificates with regard to long-term security. This is in part due to that the password transport over SSL/TLS is easier for migration of existing password-based systems on the Web and also that PAKE protocols require a large amount of computation for security enhancement, e.g., for encrypting ephemeral Diffie-Hellman public keys under the low-entropy password. There still remains a security concern as for the augmented PAKE protocols. The protocols which said to resist a server compromise are prone to off-line guessing attacks if the server is really compromised. The password information stored in the server's storage for verification purposes can be exploited by adversaries to derive real passwords. Consequently, users are still in great danger.

This document raises the fundamental questions as above to the existing form of PAKE protocols and attempts to provide a novel design to resolve these problems. We depart from the previous way of encrypting the ephemeral Diffie-Hellman public keys under the low-entropy password. Instead we map <ID, password> to <salt, verifier> for authentication. To the protocol, the former pair is the user input derived from the user's memory, while the latter is the server input fetched from the server's storage. We also introduce the hard-augmented model which enhances the previous, soft-augmented model for sever compromise security, based on the hardware security module (HSM). For the purposes, a particular concept of identity-based encryption (IBE) is employed. At the cost of pre-computation of the salt alphabet as for the ID alphabet, we can reduce the amount of computation in real time, i.e., at registration and authentication.

The resulting cryptographic protocol is the ID-based password authenticated key exchange (IPAKE) protocol which is a secure and efficient PAKE protocol in both soft- and hard-augmented models. IPAKE achieves the security goals of AKE, PAKE, and hard-augmented PAKE. IPAKE also achieves the great efficiency by allowing the whole pre-computation of the ephemeral Diffie-Hellman public keys by both server and client.

### 1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. Requirements Notation

### 2.1 Definitions

Identity Based Encryption (IBE): Identity-based encryption (IBE) is a public-key encryption technology that allows a public key to be calculated from an identity and a set of public mathematical parameters and that allows for the corresponding private key to be calculated from an identity, a set of public mathematical parameters, and a domain-wide secret value [RFC5408]. The IBE framework is defined in [RFC5091], [RFC5408], and [RFC5409].

Password Authenticated Key Exchange (PAKE): Password authenticated key exchange (PAKE) is a cryptographic protocol to achieve both mutual authentication and key agreement securely based on a password only, i.e., without requiring a public key certificate. That is, PAKE is a form of authenticated key exchange (AKE) which is in particular based on the password for authentication. The PAKE framework is defined in [IEEE P1363.2], [ISO/IEC 11770-4], and [RFC2945].

### 2.2 Abbreviations

| | |
|---|---|
| TDL | Trapdoor Discrete Logarithm |
| IBE | Identity Based Encryption |
| I-PAKE | Identity-Based Password Authenticated Key Exchange |
| AKE | Authenticated Key Exchange |

PKG Private Key Generator

HSM Hardware Security Module

MAC Message Authentication Code

SSL/TLS Secure Socket Layer/Transport Layer Security

CDH problem Computational Diffie-Hellman problem

## 2.3 Underlying Group

The I-PAKE protocol can be implemented over the following group.

Let N =p*q where p and q are sufficiently large B-smooth prime such that p = 3 (mod 4), q = 3 (mod 4), gcd(p - 1, q - 1) = 2. Let G be a multiplicative cyclic subgroup of Z_N^* with a generator g = g_m^2 where g_m is a generator of a maximal cyclic subgroup of Z_N^* of order phi(N)/2. The order of g is phi(N)/4.

## 2.4 Notations

a||b
: A concatenation of a and b.

A
: The alphabet of ID. An alphanumeric set is commonly used.

T
: The alphabet of salt. A set of discrete logarithms of the A's elements in G. This set is maintained privately by the server's HSM_s.

P
: The alphabet of password. An alphanumeric set with special characters is commonly used.

User
: A human user who actually remembers a pair of ID and password for authentication.

ID
: User's identity which is represented ID = ID_1||ID_2|| ... || ID_alpha where the ID_i is an element of Set A for all i in {1,2, ..., alpha}.

Client or C

A protocol entity providing a user interface to User for ID and password entry, and a system interface to a remote server for running the protocol. Client represents User in the protocol.

Server or S
: A protocol entity providing a system interface to Client for running the protocol, and storing a 3-tuple of ID, salt, and password-verifier for authentication.

HSM_t and HSM_v
: Private hardware attached to Server. HSM_t stores Set T and is used for deriving salt given ID. HSM_v is only used in the hard-augmented model for computing a password-verifier.

h
: A known random hash function which takes an arbitrary finite bit string and assigns it to an element of Z_N^*, where Z_N^* is a set of positive integers modulo N which are not zero divisors. In practice, a cryptographically secure one-way hash function is used.

H
: It is defined as H(ID) = (h(ID))^2 for an ID. By definition, H takes an ID and assigns it to an elements of G. That is H(ID) = I is in G. In fact, ID = ID_1||ID_2|| ... || ID_alpha, H(ID) is calculated as H(ID_1) * H(ID_2) * ... * H(ID_alpha).

h_i (i = 0, 1, 2, 3, 4, 5, 6)
: Known random hash functions. In practice, a cryptographically secure one-way hash function is used with distinct indices i. Each h_i takes an arbitrary finite bit string and assigns it to a finite string. The bit size of output of each h_i can be different from each other.

t_ID
: The secret key of the client whose identity is ID. t_ID is a discrete logarithm of I = H(ID) based on g. That is, t_ID = log_g H(ID) = log_g H(ID_1) * H(ID_2) * ... * H(ID_alpha) = log_g H(ID_1) + log_g H(ID_2) + ... + log_g H(ID_alpha) = t_ID_1 + t_ID_2 + ... + t_ID_alpha.

pw and PW
: "pw" is the password maintained by Client. Importantly, pw can be a plaintext of password, exactly remembered by User, or a one-way hash function of it, depending on the protocol and Server implementation. "PW" is the password maintained by Server. Importantly, PW can be equal to pw, exactly maintained by Client, or a one-way hash function of it, or a MAC of it, depending on the

protocol and Server implementation.

V_PW
: A password-verifier maintained by Server, such that V_PW = h(ID,t_ID,PW). This value is derived from PW by hashing in the soft-augmented model or MAC in the hard-augmented model. That is PW = h'(pw) or PW = MAC_hk(pw), where h' is a cryptographically secure one-way hash function, MAC_hk is a message authentication code that takes an element in P and assigns it to a finite bit string where hk is a secret key of HSM_v.

X and Y
: Ephemeral Diffie-Hellman public keys. X=g^x mod N and Y=g^y mod N, respectively, for randomly chosen x and y in Z_N^*.

E_K(pw)
: Encryption of pw under the ephemeral high-entropy key K. The encryption function E is a secure symmetric key encryption function, for example, AES.

C_1 and C_2
: Confirmation messages in the protocol.

'
: An indicator that discriminates the values computed by the server from the same values computed by the client. For instance, sk and sk' might be the same values but computed by respective parties.

## 3 Identity-Based Password Authenticated Key Exchange

I-PAKE is a two-party protocol where Client and Server authenticate each other and generate a session key together, based on a human-memorable password and ID information. For the purpose, the client and the server exchange messages involving ephemeral keying information and agree on a fresh session key.

### 3.1 Initialization

The initialization SHOULD be finished before the I-PAKE protocol execution.

#### 3.1.1 System Initialization

Server MAY negotiate with PKG or play the role of PKG by itself, so as to generate system parameters including underlying group parameters, ID and salt alphabets, and required functions.

Server MUST keep the salt alphabet, T, secure, e.g., in HSM_t, after the initialization.

In the hard-augmented model, Server MUST initialize HSM_v as well.

#### 3.1.2 Registration

User SHOULD select ID and a memorable password, and register them to Server through a secure channel. The secure channel at registration is out of scope in this document.

Client MAY receive the User's input and set the password as pw, so as to submit ID and pw to Server.

Server MUST fetch the discrete logarithms of H(ID_i) from T for ID, and set their sum as User's salt, t_ID. Server MAY set the received pw as PW, so as to compute a password-verifier.

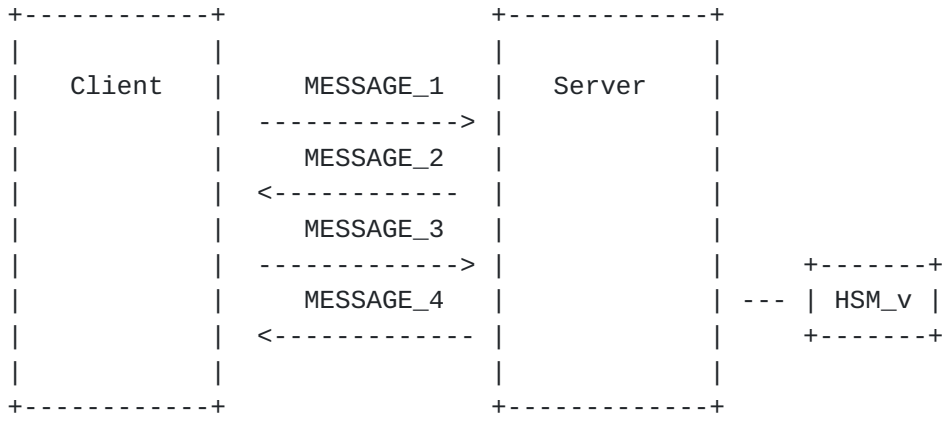
In the hard-augmented model, Server MUST negotiate with HSM_v for obtaining PW such that PW = MAC_hk(pw). In the soft-augmented model, Server MAY set PW as a function of pw.

Server SHOULD store ID, t_ID, and V_PW in its storage after computing the password-verifier such that V_PW = h(ID,t_ID,PW)

User MAY update the memorable password with a new one, and register it to Server again.

### 3.2 Protocol Execution

User MUST input ID and password at the user interface provided by Client. The protocol SHOULD be executed between Server and Client.




Client ----> Server

Client SHALL choose a random x from Z_N^* and computes its ephemeral Diffie-Hellman public key X = g^x mod N. It is an OPTION that this computation MAY be done as pre-computation.

MESSAGE_1 = ID, X

Server ----> Client

If the received value X is 1, 0, or -1, then Server MUST terminate this protocol execution. Upon receiving MESSAGE_1, Server SHALL select a random y from Z_N^* and compute Y=g^y mod N. It is an OPTION that this computation MAY be done as pre-computation.

MESSAGE_2 = Y

Client ----> Server

If the received value Y is 1, 0, or -1, then Client MUST terminate this protocol execution. Upon receiving MESSAGE_2, Client SHALL perform the following:

o Action 1: Calculate I = H(ID).

o Action 2: Calculate e = h_0(ID,C,S,X,Y,I).

o Action 3: Calculate Z = (YI^e)^x mod N.

o Action 4: Calculate the secret key K = h_1(ID,C,S,X,Y,I,Z).

o Action 5: Calculate the session key sk = h_2(ID,C,S,X,Y,I,Z).

o Action 6: Encrypt the password pw using the secret key K, E_K(pw).

o Action 7: Calculate C_1 = h_3(ID,C,S,X,Y,I,sk).

MESSAGE_3 = E_K(pw), C_1

Server ----> Client

Upon receiving MESSAGE_3, Server SHALL perform the following:

o Action 1: Calculate I = H(ID).

o Action 2: Calculate e = h_0(ID,C,S,X,Y,I).

o Action 3: Calculate Z' = X^(y+t_ID*e) mod N.

o Action 4: Calculate the secret key K' = h_1(ID,C,S,X, Y, I, Z').

o Action 5: Calculate the session key sk' = h_2(ID,C,S,X, Y, I, Z').

o Action 6: Decrypt the received E_K(pw) using the secret key K'.

o Action 7: Calculate the password verifier V_PW = h(ID,t_ID,PW), where PW is MAC_hk(pw) in the hard-augmented model or h(pw) in the soft-augmented model.

If the calculated V_PW is not equal to the saved V_PW then Server MUST terminate this protocol execution. Otherwise, Server SHALL perform the following:

o Action 8: Calculate C_1' = h_3(ID,C,S,X,Y,I,sk).

If the received C_1 in MESSAGE_3 is not equal to C_1' then Server MUST terminate this protocol execution. Otherwise, Server shall perform the following:

o Action 9: Calculate C_2 = h_4(ID,C,S,X,Y,I,sk').

MESSAGE_4 = C_2

Since (YI^e)^x = (g^y * (g^sID)^e)^x = g^xy * g^((s_ID)ex) = X^y * X^(s_ID * e) = X^(y + s_ID*e), K = K' and sk = sk'. That is, Server calculates the same secret key with Client and can decrypt the MESSAGE_3 and obtain the same password pw. Server can confirm that Client is now operated by authenticated User and agree on the session key sk to be shared with Client.

Client

Upon receiving MESSAGE_4, Client SHALL perform the following:

o Action 1: Calculate C_2' = h_4(ID,C,S,X,Y,I,sk').

If the received C_2 does not equal to C_2' then Client MUST terminate this protocol execution. Otherwise, Client can confirm that Server is authenticated and agree on the session key sk to be shared with Server.

# 4 Security Considerations

## 4.1 General - Completeness

The I-PAKE protocol is a cryptographic protocol that achieves mutual authentication and key agreement, based on a human-memorable password and ID information, between two entities.

### 4.1.1 Mutual Authentication

A user is authenticated by a server through a pair of ID and memorable password pw in the I-PAKE protocol. For the purpose, a client of the user should submit the encryption of pw under a correct fresh encryption key K, along with the user's ID, in the protocol. Note that pw can be a secure one-way hash function of real memorable password.

The server is authenticated by the client through the salt, that is, the user's ID information based on IBE, in the I-PAKE protocol. For the purpose, the server must show the confirmation that the same session key has been derived based on the salt.

### 4.1.2 Key Agreement

A client and a server both exchange ephemeral Diffie-Hellman public keys, X and Y, and agree on the same key incorporating the correct Diffie-Hellman key based on them.

## 4.2 I-PAKE - AKE Security

The I-PAKE protocol is an AKE protocol based on a memorable password and ID information. Thus, it fulfills the requirements of AKE security.

### 4.2.1 Passive Attacks

We say that an AKE protocol is secure against passive attacks if an adversary who merely observes honest entities carrying out the protocol, fails to derive a session key, which was authenticated and agreed by the honest entities.

IPAKE is a secure AKE protocol because the messages, (U, X), (S, Y), C1, and C2, eavesdropped by a passive attacker, do not reveal the corresponding session key, sk, due to the CDH problem and the secure one-way hash function.

More specifically, sk and K are secure one-way hash functions of

protocol messages involving the seed key, zk, denoted as follows.

zk=Z*Z'^e mod N

Note that Z is a Diffie-Hellman key of X and Y, i.e., Z=g^(xy) mod N; Z' can be seen as another form of the Diffie-Hellman key of X and I, i.e., Z=g^(xt) mod N; and e is a secure one-way hash function of protocol messages involving X, Y, and I, i.e., e=h0(U,S,X,Y,I).

The confirmation messages, C1 and C2, are secure one-way hash functions of protocol messages involving sk.

The I-PAKE protocol is secure against the passive attacks.

### 4.2.2 Active Attacks

We say that an AKE protocol is secure against active attacks if an adversary who controls the protocol messages, e.g., by injection, interception, replay, and/or modification, fails to subvert the communications of the honest entities.

o Impersonation of client: In order to impersonate a client of a target user, an adversary should obtain the encryption of pw under a fresh encryption key K and its confirmation message C1. (1) Although the adversary can inject a new message X to the protocol and derive K from x and Y, the probability of constructing the correct encryption of pw is bounded by the password space. (2) Although the adversary can replay the old message X' generated previously by the honest client, the new encryption key K must be different from the old encryption key K' due to the server's new ephemeral key Y that is different from the old ephemeral key Y'. In both cases, the adversary cannot construct the correct encryption of pw and its confirmation message C1, and thus fails to impersonate the client.

o Impersonation of server: In order to impersonate a server, an adversary should obtain a correct encryption key K and its confirmation message C2. (1) Although the adversary can inject a new message Y to the protocol, the probability of computing K is bounded by the salt space. (2) Although the adversary can replay the old message Y' generated previously by the honest server, the new decryption key K must be different from the old encryption key K' due to the client's new ephemeral key X that is also distinct from the old ephemeral key X'. (3) Although the adversary can modify a new message Y, e.g., Y=g^y*I^(1/e') mod

N, for the purpose of canceling out I and so the slat s when computing a fresh key K from X, Y, and the secure one-way hash function e, the probability of obtaining e' such that e'=e must be bounded by the collision resistance of the hash function. In all cases, the adversary cannot obtain a correct encryption key K and its confirmation message C2, and thus fails to impersonate the server.

o Man-in-the-middle attack: In order to reside as a middle man in the protocol, an adversary should enforce a fresh encryption key K, encryption of pw, and confirmation messages C1 and C2, according to her own key X' and Y', respectively. Although the adversary can intercept X and Y, and inject her own key X' and Y' to replace them, respectively, into the protocol, the probability of computing K against a client is bounded by the salt space while that of constructing the encryption of pw against the server is bounded by the password space. The adversary fails to reside as a middle man in the protocol.

The I-PAKE protocol is secure against the active attacks.

#### 4.2.3 Forward Secrecy

We say AKE provides forward secrecy when the secrecy of previous session keys is not affected even if long-term secrets, such as passwords and salt in the I-PAKE protocol, of one or more entities are compromised.

If the password is compromised, an adversary should inspect the protocol messages of the previous sessions that incorporate the password but only is the encryption of pw from a conventional block cipher system. Due to the security assumption of the block cipher, the ephemeral encryption key K is not derivable. Even if K is also compromised, the previous session key is not derivable due to the security assumption of one-way hash function.

If the salt is compromised, the adversary should inspect the protocol messages of the previous sessions that incorporate the salt but only is the ID information. Due to the hardness assumption of the Diffie-Hellman problem, the previous session key is not derivable from X and Y.

The I-PAKE protocol provides the forward secrecy.

#### 4.2.4 Known Session Key

We say AKE is secure against known session key attacks if the protocol achieves its goal even if an adversary learned some

previous session keys.

If the previous session key sk is compromised, an adversary should inspect the protocol messages of the previous and/or future sessions but only is the confirmation message. Due to the hardness assumption of the Diffie-Hellman problem and the security assumption of one-way hash function, the old session keys neither reveal the password and salt information nor enforce forged key agreement.

The I-PAKE protocol is secure against the known session key attacks.

#### 4.2.5 Key Control

We say AKE is secure against key control attacks if neither entity is able to force the session key to be a function of a pre-selected value, such as either X or Y.

Due to the the hardness assumption of the Diffie-Hellman problem, neither entity can enforce the key agreement on a pre-selected value.

The I-PAKE protocol is secure against the key control attacks.

### 4.3 I-PAKE - Dictionary Attack

The I-PAKE protocol is a PAKE protocol that incorporates IBE. Thus, it fulfills the requirements of PAKE security against dictionary attacks.

#### 4.3.1 On-line Dictionary Attack

We say PAKE is secure against on-line dictionary attacks if an active adversary in the client side is only able to test a single guess from a password dictionary per on-line attempt while a server is able to count the number of failed attempts consistently, and also in the server side cannot test any guess from the password dictionary per on-line attempt. Note that the second requirement is very important in practical settings.

In order to test a guessed password on-line in the client side, the adversary should send the honest server the encryption of guessed password and its confirmation C1 after exchanging X and Y. The adversary can verify the guess according to the response of the server, while the server can also verify it and count its failure due to the decryption result. If the failure count gets to the limit, the server can lock the corresponding account.

In order to test a guessed password on-line in the server side, the adversary should decrypt out the password which was encrypted by the honest client after exchanging X and Y. The adversary cannot verify the guess because the corresponding decryption key K is not derivable.

The I-PAKE protocol is secure against the on-line dictionary attacks.

### 4.3.2 Off-line Dictionary Attack

We say PAKE is secure against off-line dictionary attacks if an active adversary is only able to remove at most a single guess from a password dictionary per session, and a passive adversary cannot remove any guess from the dictionary.

As described in the on-line dictionary attack, the active adversary can only remove a single guess from the dictionary in the client side, and no guess in the server side per session.

As described in the passive attack, the passive adversary cannot derive a decryption key K, and thus cannot remove any guess from the dictionary per session.

The I-PAKE protocol is secure against the off-line dictionary attacks.

## 4.4 I-PAKE - Server Compromise

The I-PAKE protocol is an augmented PAKE protocol that fulfills the requirements of PAKE security against server compromise, not only in the previous (soft-) augmented model but also in the new hard-augmented model.

### 4.4.1 Soft-Augmented Model

We say PAKE is secure against the server compromise in the soft-augmented model if an adversary who obtained a password verification directory stored by the server cannot impersonate the client of the target user directly, i.e., without launching the off-line dictionary attack. Note that the off-line dictionary attack is possible for the server compromise in the soft-augmented model.

Since the server stores only ID, salt, and verifier in the password verification directory, the adversary who obtained the directory cannot construct the encryption of password.

The I-PAKE protocol is secure against the server compromise in the soft-augmented model.

#### 4.4.2 Hard-Augmented Model

We say PAKE is secure against the server compromise in the hard-augmented model if an adversary who obtained a password verification directory stored by the server cannot impersonate the client of the target user and launch the off-line dictionary attack. The basic assumption is that the HSM module is never compromised.

Since the server stores only ID, salt, and verifier in the password verification directory, the adversary who obtained the directory cannot construct the encryption of password.

Since the verifier is the MAC of password information under the MAC key which is only stored in the HSM module, the adversary who obtained the directory cannot launch the off-line dictionary attack.

The I-PAKE protocol is secure against the server compromise in the hard-augmented model.

## 5 IANA Considerations

This document includes no request to IANA.

## 6 References

### 6.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", RFC 2945, September 2000.

### 6.2 Informative References

[RFC5408] Appenzeller, G., Martin, L., and M. Schertler, "Identity-Based Encryption Architecture and Supporting Data Structures", RFC 5408, January 2009.

[RFC5409] Martin, L. and M. Schertler, "Using the Boneh-Franklin and

Boneh-Boyen Identity-Based Encryption Algorithms with the Cryptographic Message Syntax (CMS)", RFC 5409, January 2009.

[RFC5091] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.

[IEEE P1363.2]
IEEE P1363.2, "Password-Based Public-Key Cryptography", Submissions to IEEE P1363.2, <http://grouper.ieee.org/groups/1363/passwdPK/submissions.html>.

[ISO/IEC 11770-4]
ISO/IEC JTC 1/SC 27 11770-4, "Information technology - Security techniques - Key management - Part 4: Mechanisms based on weak secrets", May 2006, <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39723>.

## Authors' Addresses

Taekyoung Kwon
Sejong University
98 Kunja-dong, Kwangjin-gu
Seoul 143-747, Korea

EMail: tkwon@sejong.edu

Hyojin Yoon
Samsung SDS
5th Fl., Medison Bldg.,
Daechi-dong, Gangnam-gu,
Seoul 135-280, Korea

EMail: hj1230.yoon@samsung.com

Sangyoub Kim
Samsung SDS
4th Fl., Medison Bldg.,
Daechi-dong, Gangnam-gu,
Seoul 135-280, Korea

EMail: sy9.kim@samsung.com