Hidden Credential Retrieval Without Random Oracles

Atsuko Miyaji[1], Mohammad Shahriar Rahman[1], and Masakazu Soshi[2]

[1] School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, Japan 923-1292,
{miyaji,mohammad}@jaist.ac.jp
[2] School of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi Asa-Minami-Ku, Hiroshima, Japan 731-3194
soshi@hiroshima-cu.ac.jp

Abstract. To address the question of secure and efficient management of the access credentials so that a user can store and retrieve them using a 'short and easy-to-remember' password in a connected world, X. Boyen proposed a user-centric model in ASIACCS'09, named Hidden Credential Retrieval (HCR). The protocol was shown secure under random-oracle model. However, the construction does not explicitly prevent an HCR server from colluding with the third party service provider (i.e., an online bank), which can result into retrieving the hidden credential without the user's participation. In this paper[3], we show the HCR construction without the random-oracles with enhanced properties based on Okamoto's blind signature scheme proposed in TCC'06. For the "Insider attack" model, we provide the attacker (server) with more computational ability in trying to recover the plaintext message from the ciphertext that has been stored in the server by the user, being completely offline. Moreover, we include an explicit notion of identity ID that is useful in practice, so that the server knows whose encrypted credential is to be used in the protocol.

1 Introduction

Digital credentials prove something about their owner. It may contain personal information such as a person's name, birthplace, and birthdate, or biometric information such as a picture or a finger print, or some property, status, or right of its owner without revealing the owner's identity. Credentials are issued by organizations that ascertain the authenticity of the information and can be provided to verifying entities on demand. As the network world is growing in volume offering many services online, concerns also grow about the management of the required credentials to access those services. Various credentials have different formats- from simple to too complex, and it is difficult for the owners to remember or carry all the credentials with them. Here comes the question of

[3] This study is partly supported by Grant-in-Aid for Scientific Research (C), 20500075.

secure and efficient management of the credentials so that a user can store and retrieve them using a 'short and easy-to-remember' password.

Although many existing solutions can be employed to provide such a service, those solutions require certain assumptions on the distribution of the password secure against only a subset of the possible attackers. To solve this, a credential retrieval protocol that provides no explicit success/failure feedback to either party is required such that on the correct password, the user (alone) receives the correct decrypted plaintext, and the user retrieves a pseudo-random string that varies with the password on the incorrect password input. So, the trick is to keep the unauthenticated password-encrypted data on a server, and perform the retrieval and decryption in a single oblivious password-based protocol such that the failure is noticed to no party. When implemented correctly, this strategy offers the best protection against every (computationally-bounded) adversary, for every distribution of the password and the plaintext. It is optimal against both the outsiders who try one password at a time attempting to impersonate one party to the other, and the insiders who can simulate the protocol offline to create a list of plaintexts from the password dictionary. In order to avoid the successful password-recovery by the attacker, it is required that there is no information leaked on success/failure status- thus making the task as hard as exhaustive search. This situation is extremely desirable for security from the user's point of view, assuming low- or no-redundancy secret data. To achieve this, Hidden Credential Retrieval (HCR) has been proposed by X. Boyen in ASIACCS'09 [7] which requires a non-standard discrete-log-type hardness assumption in the random-oracle model. HCR refers to a client-server mechanism whereby the client (a user) can deposit a 'credential', or cryptographic key for some third-party service, on the server, remotely retrievable using a short password. Neither the server nor the communication network are trusted by the client: in particular, the server should not be able to learn either the password or the stored credential.

The whole premise of HCR of [7] is that the thing being hidden/encrypted is the credential that the user needs to use it when interacting with another party for online authentication (for example, the web server of an online bank, but not the server storing the hidden credential, the HCR server). Let us consider a real world scenario when this party colludes with the HCR server to recover the user's credential. This is particularly important for cloud infrastructure where users store their credentials in a server (HCR), and use some service from an online bank which also uses the cloud infrastructure. Now, after the user has stored his credential in an HCR server and has registered with the online bank, the HCR server and the bank may try to collude to retrieve the hidden credential without any participation of that user. Boyen's work have not addressed such a scenario for his proposed HCR scheme explicitly. Also, it is not clear from Boyen's scheme how the HCR identifies an user, and replies to the user's request. This is important for practical implementation where there can be thousands of users registered with an HCR server.

In this paper, we will often have brief visits to Boyen's work as our work follows the basic framework of the original HCR.

1.1 Related Work

In this section, we will have a look at some of the existing techniques that are closely related to the concept of HCR. A detailed discussion and comparison with many of them can be found in [7].

Multi-Party Computation: Two or more parties can compute a public function while keeping their respective inputs secret using general Multi-Party Computation (MPC) [9, 14, 23]. HCR may be viewed as a special kind of MPC. Where the characterization of HCR as mere MPC fails, is in the existing MPC protocols' difficulty to have the parties reuse their secret inputs, which is essential in HCR due to the need to keep the plaintext hidden from the storage server.

Oblivious Transfer and Private Information Retrieval: Oblivious Transfer (OT) allows a receiver to obtain the messages designated by its indices, "obliviously", i.e., without the sender learning anything about the indices, or the recipient about the remaining messages [3, 20]. Private Information Retrieval (PIR) is a type of OT where queries should remain secret from the sender only focusing on the privacy of the recipient [11, 13]. The idea of OT and PIR fails to provide a suitable HCR as because of the need to represent the password-to-plaintext map as an explicit database, of size linear in the admissible password space. On the other hand, in Encrypted Keyword Search (EKS), an encrypted data on the server is remotely searchable by the client against pre-programmed keywords using encrypted queries [6, 1]. However, it also does not provide a way to construct HCR as the client first needs to commit to a manageable sized password space, and then set up the server with one encrypted searchable string per password.

Password-Authenticated Key Exchange(PAKE): PAKE allows two parties to share a short password for establishing an authenticated secure channel across an adversarially controlled medium allowing the client to keep the password secret from the server [2, 16]. These protocols require explicit authentication since their main purpose is to provide mutual authentication in which case notifying the success or failure is necessary, where as HCR does not require this as this can result into an offline password test for the server.

Blind Signatures: The notion of blind signature protocols is such that they allow a user to obtain signatures from a signer on any document in such a manner that the signer learns nothing about the message that is being signed [12, 4, 21]. Since the concept of blind signatures was introduced by [12], it has been used in many applications including electronic voting and electronic cash. While [12] was based on RSA, and [4] proposed a blind signature based on the bilinear pairings, both of the constructions are showed secure in the random-oracle model from suitable complexity assumptions. On the other hand, [21] proposed a blind signature scheme based on bilinear pairings which is secure without random-oracles from 2SDH assumptions. The blind signature scheme of [21] is much more efficient than the other blind signature schemes in the standard model such as

the Camenisch-Koprowski-Warinsch [8] and Juels-Luby-Ostrovsky [15] schemes, and is also almost as efficient as the most efficient blind signature schemes whose security has been analyzed heuristically or in the random oracle model. HCR can be constructed from these blind signatures by setting the message as the password. However, blind signatures provide public verification feature, thus being more powerful than HCR which supports no such verification functions.

Boyen proposed HCR [7] based on Boldyreva's blind signature scheme [4]. The Boldyreva signature is very efficient using a bilinear pairing for its implementation, requiring a GDH assumption for its security reduction, in the random-oracle model. Boyen modified that scheme to build a concrete HCR protocol in prime-order abelian groups under the same assumption without the pairing requirement (since the signing function alone is sufficient to construct an HCR protocol). But it suffers from relying on the random-oracle model (using hash functions). However, the random oracle model cannot be realized in the standard (plain) model. Schemes constructed in random-oracle model do not rule out the possibility of breaking the scheme without breaking the underlying intractability assumption. Nor do they even rule out the possibility of breaking the scheme without finding some kind of weakness in the hash function, as shown by [10]. Moreover, Boyen's HCR does not address the problem when the HCR server and the third party service provider (i.e. online bank) try to collude to retrieve the crdential. Also, it does not clarify how an HCR server can identify a user requesting for her credentials, so that the server knows whose stored ciphertext is to be used in the protocol.

1.2 Our Contribution

In this paper, we show the HCR construction without random-oracles with enhanced properties based on Okamoto's blind signature scheme [22] under 2SDH assumption. For the "Insider attack" model, we provide the attacker (server) with more computational ability in trying to recover the plaintext message from the ciphertext that has been stored in the server by the user, being completely offline. This feature is particularly important to be addressed when the HCR server colludes with the third party service provider in order to retrieve the credential. We also enable the HCR server to identify a requesting user with its ID. Having an explicit notion of identity ID is useful in practice, so that the server knows whose stored ciphertext to use in the protocol. This ID is simple, public, and ideally chosen by the user.

Organization of the paper: The remainder of this paper is organized as follows: Section 2 presents the assumptions and an abstract view on the protocol model. Section 3 describes the security requirements and definitions. Section 4 includes protocol construction and a security analysis. We give some concluding remarks in Section 5.

2 Preliminary

In this section, we will give an outline of our HCR, assumptions, adversarial threat models, and definitions.

2.1 Outlining HCR

The Hidden Credential Retrieval model involves three entities: a preparer $\mathcal{P}$, a querier $\mathcal{Q}$, and a server $\mathcal{S}$. $\mathcal{P}$ and $\mathcal{Q}$ represent a user during the setup (preparing) and the query (retrieval) phases of the protocol. $\mathcal{S}$ is the server having unlimited amount of storage capability, and where a user stores his/her retrievable credentials. In general, HCR consists of the following two protocols:

Store: $\langle \mathcal{P}([P,\, M],\, ID),\, \mathcal{S}[\perp] \rangle \rightarrow \langle (C, ID),\, (C, ID) \rangle$

This protocol is the initial setup phase, and assumed to be done once over a secure channel. In this phase, a user acts as the preparer $\mathcal{P}$ and $\mathcal{S}$ is the selected storage server. *Store*'s purpose is to set up the long-term secrets, especially that of server's. In a practical setting, a user must have selected a server with which to get services, and must be able to communicate securely with it for the initial setup: this is done in the usual way in HCR where we require an authentic private channel for the setup phase. The reason is to provide $\mathcal{P}$ the ability to limit the disclosure of ciphertext to $\mathcal{S}$ it trusts to act as an insider. By definition, as we will see, the knowledge of ciphertext separates an "insider" from an "outsider". The user $\mathcal{P}$ also registers its id ID. This ID can be public, and ideally chosen by the user. It can be transmitted in the clear later in retrieval phase in the messages, and it plays no role in the security.

- $\mathcal{P}$ takes two private inputs: a memorable password P and a plaintext credential M. $\mathcal{P}$ also picks its ID on its own choice.

- $\mathcal{S}$ takes no private input, denoted by the null symbol.

At the end of the protocol, $\mathcal{S}$ will have acquired a private credential ciphertext C and the ID of user $\mathcal{P}$. Although C is intended for $\mathcal{S}$ alone, $\mathcal{P}$ can learn it too; but nobody else should. ID is simple, plain, and can be known publicly.

Retrieve: $\langle \mathcal{Q}([P'],\, ID),\, \mathcal{S}([C'],\, ID) \rangle \rightarrow \langle M',\, \perp \rangle$

This protocol can be repeated for any number of times over adversarial channels between the user (named as $\mathcal{Q}$) and the server $\mathcal{S}$.

- The querier $\mathcal{Q}$ takes one private input: a password P'. It also takes its public value ID.

- The server $\mathcal{S}$ takes one private input: a ciphertext C'. It also takes the public identifier ID of the requesting querier $\mathcal{Q}$.

At the end of this protocol, $\mathcal{S}$ learns $\perp$, or nothing at all; whereas $\mathcal{Q}$ retrieves a plaintext M' which is a deterministic function of both parties' inputs. M' must satisfy the following condition with respect to the inputs used by $\mathcal{P}$ and $\mathcal{S}$ in the *Store* protocol:

$(P' = P) \wedge (C' = C) \Rightarrow (M' = M)$

It is important that, neither of $\mathcal{S}$ and $\mathcal{Q}$ can learn the fact whether $\mathcal{Q}$ could retrieve the correct M from this protocol. We suppose that the password P is always drawn uniformly from a public dictionary D, and that, in the view of the adversary, the prior distribution of the plaintext M is uniform over the whole domain $\{0,1\}^k$ (and which in the concrete protocol is further represented as an element of $\mathbb{G}$). This is because, as we will see in the security definitions, M is drawn from a subset MS about which the adversary has no prior information other than $MS \subseteq \{0,1\}^k$.

2.2 Bilinear Groups

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups as follows:

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of prime order p, where possibly $\mathbb{G}_1 = \mathbb{G}_1$,
2. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$,
3. ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$,
4. e is a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, i.e.,
   (a) Bilinear: for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$,
   (b) Non-degenerate: $e(g_1, g_2) \neq 1$ (i.e., $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$),
5. e, ψ and the group action in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ can be computed efficiently.

2.3 Assumptions

Here we use the assumption from [22], the 2-variable strong Diffie-Hellman (2SDH) assumption on which the security of the proposed signature scheme (i.e. the HCR) is based.

Variant of q 2-Variable Strong Diffie-Hellman (q-2SDH$_S$) Problem. The q-2SDH$_S$ problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: given a $(3q+4)$-tuple $(g_1, g_2, w_2 \leftarrow g_2^x, u_2 \leftarrow g_2^y, g_2^{\frac{y+b_1}{x+a_1}}, \ldots, g_2^{\frac{y+b_q}{x+a_q}}, a_1, \ldots, a_q, b_1, \ldots, b_q)$ as input, output $(\sigma \leftarrow g_1^{\frac{y+d}{\theta x+\rho}}, \alpha \leftarrow g_2^{\theta x+\rho}, d)$ as well as $Test(\alpha) \leftarrow (U, V)$, where $a_1, \ldots, a_q, b_1, \ldots, b_q, d, \theta, \rho \in \mathbb{Z}_p^*$; $w_1 \leftarrow \psi(w_2), \sigma, U \in \mathbb{G}_1$; $\alpha, V \in \mathbb{G}_2$; and

$$e(\sigma, \alpha) = e(g_1, u_2 g_2^d), \quad e(U, \alpha) = e(w_1, w_2) \cdot e(g_1, V), \quad d \notin \{b_1, \ldots, b_q\} \tag{1}$$

Algorithm $\mathcal{A}$ has advantage, $Adv_{2SDH_S}(q)$, in solving q-2SDH$_S$ in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$\begin{aligned} Adv_{2SDH_S}(q) \leftarrow Pr[\mathcal{A}(g_1, g_2, w_2, u_2, g_2^{\frac{y+b_1}{x+a_1}}, \ldots, g_2^{\frac{y+b_q}{x+a_q}}, a_1, \ldots, a_q, b_1, \ldots, b_q) \\ = (\sigma, \alpha, d, Test(\alpha))] \end{aligned} \tag{2}$$

where Eq.(1) holds. The probability is taken over the random choices of $g_2 \in \mathbb{G}_2, x, y, a_1, b_1, \ldots, a_q, b_q \in \mathbb{Z}_p^*$, and the coin tosses of $\mathcal{A}$.

Definition 1. *Adversary $\mathcal{A}$ (t, ϵ)-breaks the q-2SDH$_S$ problem if $\mathcal{A}$ runs in time at most t and $Adv_{2SDH_S}(q)$ is at least ϵ. The (q, t, ϵ)-2SDH$_S$ assumption holds if no adversary $\mathcal{A}$ (t, ϵ)-breaks the q-2SDH$_S$ problem.*

We occasionally drop q, t, ϵ, S and refer to the 2SDH assumption rather than the (q, t, ϵ)-2SDH$_S$ assumption denoting a polynomial number of q, a polynomial-time of t and negligible probability of ϵ in security parameter n. A detailed discussion on the assumption can be found in [22].

3 Security Requirements

In this section, we first have a brief look at the threat model informally. Then we will provide game-based definitions to formalize the security requirements capturing all the required properties. To state informally:

- Passive eavesdroppers should gain no computational advantage in recovering M or P by observing arbitrarily many protocol execution transcripts between the two honest players $\mathcal{Q}$ and $\mathcal{S}$.

- An active adversary impersonates $\mathcal{Q}$ or $\mathcal{S}$, or modifies messages between $\mathcal{Q}$ and $\mathcal{S}$. Allowed with maximum one guess test per protocol execution, it should gain no advantage in learning anything other than whether a particular password guess P' is correct or not.

- Even though the server $\mathcal{S}$ is entrusted with the ciphertext C, recovering the corresponding plaintext M efficiently should not be possible more than by running a brute-force offline dictionary attack against the encryption password P. Even arbitrarily many protocol executions with the user $\mathcal{Q}$ should not enable $\mathcal{S}$ to recover M.

- The retrieval protocol itself should be blind, i.e., keep the password invisible to the server.

- The retrieval protocol should also be oblivious, i.e., not disclose its success to either party.

- The encrytion of message M into ciphertext C has to be redundancy-free.

If the plaintext M is a random access key for a separate third-party service, then under the above conditions, it will be impossible for the server to recover the password (or the plaintext) in an offline dictionary attack. We assume that the server $\mathcal{S}$ is partially trusted, and we explicitly allow it to misbehave. On the other hand, the user is ultimately trusted, since all the data belong to him.

3.1 Oracles for Validity Tests

In this model, the attacker will not know a priori the set $MS \subseteq \{0,1\}^k$ of admissible plaintexts from which the correct message M is to be drawn uniformly. We have the following oracles:

- $\mathcal{TO}_1$ captures the offline recognition of a potentially valid plaintext on the basis of its intrinsic redundancy: $\mathcal{TO}_1[M'] = 1$ means that M' is well-formed, i.e., it is in the set MS, though it is not necessarily correct.

- $\mathcal{TO}_2$ requires an online component and the cooperation of a third party to run an expensive but perfectly accurate validity check: $\mathcal{TO}_2[M'] = 1$ indicates

that M' is usable in stead of the correct M with the third party, and thus typically that M' is the correct M.

- $\mathcal{TO}_3$ captures the feature that being offline and given a valid C, runs an expensive validity check: $\mathcal{TO}_3[M''] = 1$ indicates that any M'' is usable in stead of the correct M. $\mathcal{TO}_3$ makes sense when a third party colludes with the HCR server to recover the user's credential. This oracle is only available to the "Insider".

3.2 Outsider Security

We define the privacy model against outsider attacks. It is based on the following game, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$. The challenger simulates all the parties in the HCR protocol. We consider $\mathcal{Q}$ and $\mathcal{S}$, and exclude $\mathcal{P}$ from consideration as it communicates with $\mathcal{S}$ over a secure channel. The outsider adversary acts passively when it makes requests for transcripts of *Retrieve* protocol between $\mathcal{Q}$ and $\mathcal{S}$.

Besides passive eavesdropping, the adversary can also actively impersonate $\mathcal{Q}$ to $\mathcal{S}$, or $\mathcal{S}$ to $\mathcal{Q}$, by interfering the concurrent but independent protocol executions. (It cannot corrupt or read the internal state of any of the actual players.) The following is the attack game analogous to the attack game in [7]:

Game 1:

- **Initialization:** $\mathcal{B}$ privately simulates an execution of the *Store* protocol between $\mathcal{P}$ and $\mathcal{S}$, for a random password $P \in \{0,1\}^n$ and a random message $M \in \{0,1\}^n$.

The distribution of M is assumed to be uniform over some subset $MS \subseteq \{0,1\}^k$, such that $\forall m \in \{0,1\}^k$: $m \in MS \Leftrightarrow \mathcal{TO}_1[m] = 1$. MS is thus the set of well-formed plaintexts, and is a parameter of the game but is not given to $\mathcal{A}$. This is to force $\mathcal{A}$ to make accountable calls to the $\mathcal{TO}_1$-oracle if it wants to test candidate messages for membership to MS.

- **Eavesdropping queries:** $\mathcal{A}$ can adaptively request to see the transcript of a random execution between $\mathcal{Q}$ and $\mathcal{S}$, in which $\mathcal{Q}$ uses the correct password $P' = P$.

- **Impersonation queries:** $\mathcal{A}$ can adaptively send messages to $\mathcal{S}$ or to $\mathcal{S}$; it immediately obtains the corresponding reply if any reply is due.

- **Offline validity tests:** $\mathcal{A}$ can make adaptive calls to the offline oracle $\mathcal{TO}_1$ on any string of its choice. The response indicates whether the string belongs in MS.

- **Online validity tests:** $\mathcal{A}$ can make adaptive calls to the online oracle $\mathcal{TO}_2$ on any string of its choice. The response indicates whether the string is the correct message M.

- **Message guess:** $\mathcal{A}$ eventually outputs one guess $\tilde{M}$ for the value of M.

- **Adjudication:** The adversary wins if $\tilde{M} = M$.

Definition 2. *The advantage of an adversary $\mathcal{A}$ in a (w, q, t_1, t_2)-outsider attack is defined as the probability that $\mathcal{A}$ wins the Game 1, when $\mathcal{A}$ makes a total*

of w passive eavesdropping queries, q active impersonation queries, and t_1 and t_2 calls to $\mathcal{TO}_1$ and $\mathcal{TO}_2$, respectively.

3.3 Insider Security

The user ($\mathcal{P}$ or $\mathcal{Q}$) is trusted, and the only possible insider attacker is $\mathcal{S}$ (or any entity that has managed to acquire C from $\mathcal{S}$, and which is thus equivalent to $\mathcal{S}$). This game is played between a malicious server $\mathcal{A}_S$ and a challenger $\mathcal{B}$. $\mathcal{B}$ simulates the trusted user $\mathcal{P}$ and $\mathcal{Q}$. As in the full protocol, the attack may have two phases. The first one contains a single execution of the *Store* protocol between $\mathcal{P}$ and $\mathcal{A}_S$; the second may be run by the adversary as many number of independent executions of *Retrieve* between $\mathcal{Q}$ and $\mathcal{A}_S$ as it wants, and in which $\mathcal{Q}$ will use the correct password $P' = P$. $\mathcal{A}_S$ wants to recover M. In our definition, the notion of insider security adds the fact to the original definition that the adversary tries to guess the valid (M, P) pair right after the *Store* phase by interacting with $\mathcal{TO}_3$.

The insider attack game proceeds as follows:

Game 2:

- Storage interaction: $\mathcal{B}$, acting on behalf of $\mathcal{P}$, picks a random password $P \in \{0,1\}^n$ and a random message $M \in MG \subseteq \{0,1\}^k$, and engages in the *Store* protocol with the adversary $\mathcal{A}_S$.

- Offline recovery tests: $\mathcal{A}_S$ can make adaptive calls to the offline oracle $\mathcal{TO}_3$ on stored ciphertext C and any string M'' of its choice. The response indicates whether the string is usable instead of the correct M.

- Retrieval interactions: $\mathcal{B}$, acting on behalf of $\mathcal{Q}$, initiates the *Retrieve* protocol multiple times with the adversary $\mathcal{A}_S$, using the correct access password $P' = P$.

- Offline validity tests: $\mathcal{A}_S$ can make adaptive calls to the offline oracle $\mathcal{TO}_1$ on any string of its choice. The response indicates whether the string belongs in MS.

- Online validity tests: $\mathcal{A}_S$ can make adaptive calls to the online oracle $\mathcal{TO}_2$ on any string of its choice. The response indicates whether the string is the correct message M.

- Message guess: $\mathcal{A}_S$ eventually outputs one guess $\tilde{M}$ for the value of M.

- Adjudication: The adversary wins if $\tilde{M} = M$.

Definition 3. *The advantage of an adversary $\mathcal{A}_S$ in a (z, t_1, t_2, t_3)-insider attack is defined as the probability that $\mathcal{A}_S$ wins the preceding game, after a total of z initiated instances of the Retrieve protocol, and a total of t_1, t_2, and t_3 oracle calls to $\mathcal{TO}_1$, $\mathcal{TO}_2$, and $\mathcal{TO}_3$, respectively.*

Analogous definitions on the password recovery can also be stated in a similar game-based approach.

Definition 4. *An HCR is fully secure if the advantages of $\mathcal{A}$ winning (w, q, t_1, t_2)-outsider attack game, and $\mathcal{A}_S$ winning the (z, s, t_1, t_2)-insider attack game, respectively, are negligible.*

4 The Proposed HCR Scheme

We have already referred to a generic transformation from blind signature protocols into HCR protocols without random oracles. We refer to two secure blind signature schemes that have been presented without random oracle model [8, 15] other than [21, 22]. However, the construction of [15] is based on a general two-party protocol and is thus extremely inefficient. The solution of [8] is much more efficient than that of [15], but it is still much less efficient than the secure blind signature schemes in the random oracle model. To construct our HCR schme, we use the signature scheme of [21, 22] since this is almost as efficient as the most efficient blind signature schemes whose security has been analyzed heuristically or in the random oracle model. The blind signature scheme is also secure for polynomially many synchronized (or constant-depth concurrent) attacks, but not for general concurrent attacks. The [21, 22] blind signatures actually require a bilinear pairing for their implementation, but that is because the pairing is needed for signature verification. The blind signature strategy is mainly applied in the retrieval phase. The signing function alone is sufficient to construct an HCR protocol, therefore our construction will not need a pairing (however, the scheme remains compatible with pairing-friendly groups).

Moreover, the HCR server requires to store the ID of the user along with the user's other values in the *Store* phase. Having an explicit notion of user ID is very useful in practical implementation, so that the server knows whose C to use in the *Retrieve* protocol. There are a number of efficient database search algorithms in the literature [17, 19]. Many other searching algorithms can be found in [18]. We leave it to the designers to pick up the most efficient and convenient searching algorithm.

4.1 Protocol Construction

Let ($\mathbb{G}_1$,$\mathbb{G}_2$) be bilinear groups as shown in Section 2.2. Here, we also assume that the password P to be blindly signed is an element in $\mathbb{Z}_p^*$ (analogous to message m in [21]), but the domain can be extended to all of $\{0,1\}^*$ by using a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, as mentioned in [5]. We follow the constructions of [22] - the extended version of [21].

Key Generation: Randomly select generators $g_2, u_2, v_2 \in \mathbb{G}_2$ and set $g_1 \leftarrow \psi(g_2)$, $u_1 \leftarrow \psi(u_2)$, and $v_1 \leftarrow \psi(v_2)$. Randomly select $x \in \mathbb{Z}_p^*$ and compute $w_2 \leftarrow g_2^x \in \mathbb{G}_2$. The public and secret keys are:

Public key: g_1, g_2, w_2, u_2, v_2
Secret key: x

Store: $\langle \mathcal{P}[P, M], \mathcal{S}[\perp] \rangle \rightarrow \langle \perp, C \rangle$
where $P \in \mathbb{Z}_p^*$ and M is in $\mathbb{G}_1$

1. $\mathcal{P}$ picks a generator g_1

2. $\mathcal{P}$ randomly picks $t \in \mathbb{Z}_p^*$, and computes
$$\zeta \leftarrow (g_1^{Pt})^{1/x}$$
3. $\mathcal{P}$ computes
$$\gamma \leftarrow \zeta^{-1} M$$
4. $\mathcal{P}$ chooses its id ID
5. $\mathcal{P}$ sends x, γ, its id ID, and other public values to $\mathcal{S}$
6. $\mathcal{S}$ stores x, γ, ID, and other public values of $\mathcal{P}$ in its database so that it knows whose γ and public values to use in the retrieve protocol.

Retrieve: $\langle \mathcal{Q}[P'], \mathcal{S}[C'] \rangle \rightarrow \langle M', \perp \rangle$
where $P' \in \mathbb{Z}_p^*$

1. $\mathcal{Q}$ sends the retrieval request along with its ID to $\mathcal{S}$.
2. $\mathcal{S}$ searches in its database the requesting ID. If it finds an entry for the ID, then it sends $\mathcal{Q}$ the public values along with γ.
3. $\mathcal{Q}$ randomly selects $s \in \mathbb{Z}_p^*$, and computes
$$X \leftarrow g_1^{P't} u_1^t v_1^{st}$$
and sends the blinded request to $\mathcal{S}$. In addition, $\mathcal{Q}$ proves to $\mathcal{S}$ that $\mathcal{Q}$ knows ($P't \bmod p$, t, $st \bmod p$) for X using the witness indistinguishable proof as follows:
(a) $\mathcal{Q}$ randomly selects a_1, a_2, a_3 from $\mathbb{Z}_p^*$, computes
$$W \leftarrow g_1^{a_1} u_1^{a_2} v_1^{a_3},$$
and sends W to $\mathcal{S}$.
(b) $\mathcal{S}$ randomly selects $\eta \in \mathbb{Z}_p^*$ and sends η to $\mathcal{Q}$
(c) $\mathcal{Q}$ computes
$$b_1 \leftarrow a_1 + \eta P't \bmod p,\ b_2 \leftarrow a_2 + \eta t \bmod p,\ b_3 \leftarrow a_3 + \eta st \bmod p,$$
and sends (b_1, b_2, b_3) to $\mathcal{S}$.
(d) $\mathcal{S}$ checks whether the following equation holds or not:
$$g_1^{b_1} u_1^{b_2} v_1^{b_3} = W X^{\eta}$$
After checking the equation, $\mathcal{S}$ moves on to next steps.
4. $\mathcal{S}$ randomly selects $r \in \mathbb{Z}_p^*$. In the unlikely event that $x + r = 0 \pmod p$, $\mathcal{S}$ tries again with a different random r. $\mathcal{S}$ also randomly selects $l \in \mathbb{Z}_p^*$, computes
$$Y \leftarrow (X v_1^l)^{1/(x+r)}$$
and sends (Y, r, l) to $\mathcal{Q}$.
Here $Y = (X v_1^l)^{1/(x+r)} = (g_1^{P't} u_1^t v_1^{st+l})^{1/(x+r)} = (g_1^{P'} u_1 v_1^{s+\frac{l}{t}})^{t/(x+r)}$
5. $\mathcal{Q}$ randomly selects $f, \lambda \in \mathbb{Z}_p^*$, and computes

$$\tau = (ft)^{-1} \bmod p,\ \sigma \leftarrow Y^{\tau},\ \alpha \leftarrow w_2^f g_2^{rf},\ \beta \leftarrow s + \frac{l}{t} \bmod p$$

Compute $Test(\alpha) \leftarrow (U, V)$ as follows:

$$U \leftarrow w_1^{1/f} g_1^{\lambda},\ V \leftarrow w_2^{f\lambda + r} g_2^{fr\lambda}$$

Here, $\sigma = (g_1^{P'} u_1 v_1^{s+\frac{l}{t}})^{1/(fx+fr)} = (g_1^{P'} u_1 v_1^{\beta})^{1/(fx+fr)}$, and $\alpha = w_2^f g_2^{fr} = g_2^{fx+fr}$.

6. $(\sigma, \alpha, \beta, Test(\alpha))$ is the blind signature of P'.

Now, since $\mathcal{Q}$ has got all the secrets and public values, it unblinds and decrypts to retrieve the plaintext. Recall that $\mathcal{Q}$ does not need to verify the validity of the signature, it needs to decrypt and recover the exact message. In order to do so, $\mathcal{Q}$ does the following computations:

$$\begin{aligned}
\sigma^{\frac{1}{\tau}} &= g_1^{P't} u_1^t v^{st+l} = (X v_1)^{\frac{1}{x+r}} \\
\Rightarrow \sigma^{\frac{(x+r)}{\tau}} &= X v_1^l \\
\Rightarrow \sigma^{\frac{(x+r)}{\tau}} v_1^{-l} &= g_1^{P't} u_1^t v_1^{st} \\
\Rightarrow \sigma^{\frac{(x+r)}{\tau}} v_1^{-l} u_1^{-t} v_1^{-st} &= g_1^{P't} \\
\Rightarrow \sigma^{\frac{(x+r)}{\tau}} v_1^{-\frac{(l+st)}{x}} u_1^{-\frac{t}{x}} &= (g_1^{P't})^{\frac{1}{x}} = \zeta' \\
M' &\leftarrow \zeta' \gamma
\end{aligned}$$

4.2 Security of the Proposed HCR

The blinding in the *Retrieve* phase is perfectly blind and unforgeable because of the two following theorems from [22] provided that the 2SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$:

Theorem 1. *The blind signature is perfectly blind.*

Theorem 2. *If the (q_S, t', ϵ')-2SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, the blind signature is (q_S, t, ϵ)-unforgeable against an L-interval synchronized run of adversaries, provided that*

$$\epsilon' \leq \frac{1 - 1/(L+1)}{16} \cdot \epsilon \text{ and } t' \geq O(\frac{24 L ln(L+1)}{\epsilon} \cdot t) + \Theta(q_S T)$$

The definition of $L-$*interval synchronized run*, and the proofs of the theorems can be found in [22].

Before going to provide the adversaries'(Insider and Outsider) success probability, we provide some notations. We have already stated that t_1, t_2 and t_3 are the number of unique valid queries to $\mathcal{TO}_1$, $\mathcal{TO}_2$, and $\mathcal{TO}_3$ respectively. We define n_1, n_2, n_3 as the number of negative responses to those queries, so that $n_1 \leq t_1$, $n_2 \leq t_2$ (and $n_2 \geq t_2 - 1$), and $n_3 \leq t_3$. It is also assumed that each query to $\mathcal{TO}_1$ is always followed by an identical query to $\mathcal{TO}_2$ such that

if the query to $\mathcal{TO}_1$ returns negative answer, the query to $\mathcal{TO}_2$ is not made. Furthermore queries $\mathcal{TO}_1$ then $\mathcal{TO}_2$ are systematically made on the final guess M' output by $\mathcal{A}$. As we have said earlier, $\mathcal{TO}_3$ is only available to the "Insider" attacker($\mathcal{A}_S$) only right after the *Store* phase.

Proposition 1. *(Outsider Security:) In this setting, suppose that the 2SDH complexity assumption holds in $(\mathcal{G}_1, \mathcal{G}_2)$, for some chosen security parameter n for the class of all PPT algorithms running in time t. Then, no (w, q, t_1, t_2)-outsider adversary $\mathcal{A}$ running in time t can recover the stored message M with a probability that exceeds the following bound:*

$$Pr[\mathcal{A}^{\mathcal{TO}_1, \mathcal{TO}_2} wins] \leq \frac{min\{q, t_2\}}{|D| - min\{q, n_1\}} + \frac{t_2}{2^k - n_1} + negl[n]$$

Proposition 2. *(Insider Security) In this setting, without random oracles and without any computational hardness assumption, every (z, t_1, t_2, t_3)-insider adversary $\mathcal{A}_S$ that recovers the stored message $M \in MS$, succeeds with probability at most:*

$$Pr[{\mathcal{A}_S}^{\mathcal{TO}_1, \mathcal{TO}_2, \mathcal{TO}_3} wins] \leq \frac{t_2}{|D| - n_1} + \frac{t_2}{2^k - n_1} + \frac{t_3}{2^{k-1} - n_3}$$

The probability bound of recovering the user password P can be derived similarly without random oracle model.

Note that the parameters w and r are not included in the adversaries' success probability bounds. In the "Outsider" attack, w is the number of sessions being passively eavesdropped upon, which looks random to a computationally bounded adversary. Similarly, r is the number of sessions the "Insider" attacker conducts with the user, but we know that the server receives no feedback from the user in the protocol.

Also note that the "Insider" security includes the success probability of the attacker when it is completely offline, targeting to extract the valid message M by using its own M'' from the stored ciphertext C. The attacker in this case is even stronger than that in [7], since it is allowed to make attempts being in purely offline state. The protocol provides unconditional security for the user password against insider attackers, even against dictionary attacks if furthermore the plaintext lacks redundancy, and this is arguably the most important consideration for password reusability.

Theorem 3. *The proposed HCR is fully secure without random oracles as the advantages of $\mathcal{A}$ winning the (w, q, t_1, t_2)-outsider attack game, and $\mathcal{A}_S$ winning the (z, t_1, t_2, t_3)-insider attack games, respectively, are negligible.*

Reducing the Required Computation: Step 5. in the *Retrieve* phase can be avoided to reduce the required computation, since we need to decrypt the received ciphertext instead of computing pairing. As for security, it is still based on discrete-log-type hardness assumption without random oracles.

5 Conclusion

In this paper, for the first time, we show the HCR construction without random oracles with enhanced properties. Our construction is based on Okamoto's blind signature scheme [22] with some modifications. Our construction does not use the signature verification steps of [22]. The security of credential and/or password against "Outsider attacker" is achieved based on 2SDH assumption. Although our 2SDH assumption is stronger than that of Boyen's HCR (GDH), it is still a reasonable assumption. For the "Insider attack" model, we provide the attacker (server) with more computational ability in trying to recover the message from the ciphertext that has been stored in the server by the user, being completely offline. This feature is particularly important to be addressed when the HCR server colludes with the third party service provider in order to retrieve the credential. We also enable the HCR server to identify a requesting user with its ID. Having an explicit notion of identity ID is useful in practice, so that the server knows whose stored ciphertext is to be used in the protocol. Still our protocol provides unconditional security for the user password against "Insider attackers". An HCR with refined security model under relaxed assumptions, and/or with different properties are interesting open problems.

References

1. Baek J., Naini R.S. and Susilo W.: Public Key Encryption with Keyword Search Revisited. In The International Conference on Computational Science and Its Applications- ICCSA'08. pp. 1249-1259, 2008.
2. Bellovin S.M. and Merritt M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In IEEE Symposium on Security and Privacy-SP'92. pp. 72-84, 1992.
3. Bellare M. and Micali S.: Non-interactive oblivious transfer and applications. In Advances in Cryptology- CRYPTO'89. pp. 547-557, 1989.
4. Boldyreva A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Public Key Cryptography-PKC'03. pp. 31-46, 2003.
5. Boneh D. and Boyen X.: Short Signatures Without Random Oracles. In Advances in Cryptology- EUROCRYPT'04. pp. 56-73, 2004.
6. Boneh D., Crescenzo G.D., Ostrovsky R. and Persiano G.: Public key encryption with keyword search. In Advances in Cryptology- EUROCRYPT'04. pp. 506-522, 2004.
7. Boyen X.: Hidden Credential Retrieval from a Reusable Password. In the Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security- ASIACCS'09. pp. 228-238, 2009.
8. Camenisch J., Koprowski M. and Warinschi B.: Efficient Blind Signatures without Random Oracles. In the Fourth Conference on Security in Communication Networks- SCN'04. pp. 134-148, 2004.
9. Canetti R., Feige U., Goldreich O. and Naor M.: Adaptively Secure Multi-Party Computation. In the Proceedings of the 28th Annual ACM Symposium on Theory of Computing, STOC'96. pp. 639-648, 1996.

10. Canetti R., Goldreich O. and Halevi S.: The Random Oracle Methodology, Revisited. In the Proceedings of the 30th ACM Symposium on Theory of Computing, STOC'98. pp. 209-218, 1998.
11. Cachin C., Micali S. and Stadler M.: Computationally private information retrieval with polylogarithmic communication. In Advances in Cryptology- EUROCRYPT'99. pp. 402-414, 1999.
12. Chaum D. Blind signatures for untraceable payments. In Advances in Cryptology-CRYPTO'82. pp. 199-203, 1982.
13. Chor B., Goldreich O., Kushilevitz E. and Sudan M.: Private information retrieval. In IEEE Symposium on Foundations of Computer Science- FOCS'95. pp. 41-51, 1995.
14. Goldreich, O., Micali, S. and Wigderson A.: How to Play any Mental Game . The 19th Annual ACM Symposium on the Theory of Computing, STOC'87. pp. 218-229, 1987.
15. Juels A., Luby M. and Ostrovsky R.: Security of blind digital signatures. In Advances in Cryptology- CRYPTO'97. pp. 150-164, 1997.
16. Katz J., Ostrovsky R. and Yung M.: Efficient password-authenticated key exchange using human-memorable passwords. In Advances in Cryptology- CRYPTO'01. pp. 475-494, 2001.
17. Kimelfeld B. and Sagiv Y.: Efficient engines for keyword proximity search. In WebDB'05. 2005.
18. Knuth D.: The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997.
19. Liu F., Yu C., Meng W. and Chowdhury A.: Effective keyword search in relational databases. In Proceedings of the 2006 ACM SIGMOD'06. pp. 563-574, 2006.
20. Naor M. and Pinkas B.: Oblivious transfer with adaptive queries. In Advances in Cryptology- CRYPTO'99. pp. 573-590, 1999.
21. Okamoto T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In Theory of Cryptography- TCC'06. pp. 80-99, 2006.
22. Okamoto T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In Cryptology ePrint Archive: Report 2006/102. http://eprint.iacr.org/2006/102
23. Yao, A.: How to Generate and Exchange Secrets. The 27th Annual IEEE Symposium on the Foundations of Computer Science, FOCS'86. pp. 162-167, 1986.