Hidden Credential Retrieval from a Reusable Password

Xavier Boyen
Stanford Univ.
xb@boyen.org

December 21, 2008

Abstract

We revisit the venerable question of access credentials management, which concerns the techniques that we, humans with limited memory, must employ to safeguard our various access keys and tokens in a connected world. Although many existing solutions can be employed to protect a long secret using a short password, those solutions typically require certain assumptions on the distribution of the secret and/or the password, and are helpful against only a subset of the possible attackers.

After briefly reviewing a variety of approaches, we propose a user-centric comprehensive model to capture the possible threats posed by online and offline attackers, from the outside and the inside, against the security of both the plaintext and the password. We then propose a few very simple protocols, adapted from the Ford-Kaliski server-assisted password generator and the Boldyreva unique blind signature in particular, that provide the best protection against all kinds of threats, for all distributions of secrets. We also quantify the concrete security of our approach in terms of online and offline password guesses made by outsiders and insiders, in the random-oracle model.

The main contribution of this paper lies not in the technical novelty of the proposed solution, but in the identification of the problem and its model. Our results have an immediate and practical application for the real world: they show how to implement single-sign-on stateless roaming authentication for the internet, in a ad-hoc user-driven fashion that requires no change to protocols or infrastructure.

Keywords: stateless roaming credentials, reusable passwords, online authentication, partially trusted servers.

1 Introduction

Suppose that you need frequent access to a small piece of secret information, say, your credentials to various Internet services. However, the credentials are too complex to remember, and it would be unsafe simply to write them down and carry them in your wallet everywhere you go. What do you do?

Surprisingly, this practical question has not received much direct attention from the security community, despite the vast body of research on (sometimes very closely related) password-based protocols. To see why this problem is worth considering, let us deconstruct all the natural candidate strategies to expose their shortcomings.

The two most basic candidate strategies are the following:

(A) Encrypt the data under a short memorized password, and carry the ciphertext with you.

(B) Entrust the data to a remote server, and make it accessible using password authentication.

Strategy A is the only viable approach to a stand-alone user (*i.e.*, an isolated individual without access to any kind of remote storage, whether trusted or not). Because the ciphertext is not well defended, it is an easy target for capture, at which point the weak password that serves as its encryption key may quickly succumb to a brute-force offline dictionary attack. Since this strategy primarily relies on physical security of the bearer, this is not much better than merely carrying the plaintext and taking appropriate precautions to protect it. (As an aside, we note also that the assumption of a stand-alone user is specious if the credentials are intended for remote use, and especially for internet use.)

Strategy B is the diametrical opposite: it defers onto a remote server as much of the burden (and associated trust) of safekeeping one's credentials, as one possibly could. The main benefit of this second strategy is that it is immune to the physical and offline attacks of the previous strategy. In particular, since any password guessing must occur online (*i.e.*, by interacting with the storage server, or perhaps with the user), such attack can be quickly detected and interdicted. The drawback of the second approach, however, is that the storage server itself now has unfettered access to the user credentials in plaintext. This strategy thus requires a remote server that is ultimately trusted by the user, and is of course worthy of such trust.

In view of these shortcomings, the next natural idea is to try to combine the respective strengths of A and B: the former would provide security against insiders, by keeping the user credentials encrypted under the user's password; while the latter would provide security against outsiders, by making it difficult to access the ciphertext.

This leads us to the following hybrid strategy:

(C) Combine A plus B: store the password-encrypted data on a password-authenticating server.

Strategy C simultaneously removes the threat of physical theft against the user and provides good protection against a malicious storage server, but only if the encryption password and the access password are unrelated to each other. If the two passwords are the same, then the server can use its knowledge of the authentication password to decrypt the ciphertext it has been entrusted with. Alas, remembering two distinct passwords is not only an unnecessary burden, but it is also a serious blunder in terms of concrete security, since in principle it is exponentially easier to guess two sequential passwords of length n, than to guess a single password of length $2n$.

It turns out that the two main drawbacks of the latter approach (namely: the need for the server to know the authentication password, and the consequential need for the user to remember two distinct passwords) can be fixed easily. All we need to to is to use a "good" authentication protoocol.

Thus, we would like a version of Strategy C based on an authentication mechanism that does not require server knowledge of the user password, so that the same password can also be used to encrypt the user data. This means in particular that the complete retrieval protocol must be "oblivious"; *i.e.*, at the end of the protocol, the user client succeeds or fails to retrieve the data, depending on the password, but the storage server is none the wiser about the outcome.

This leads us to our fourth incremental strategy:

(D) Same as C, but with a user-private authentication password, equal to the encryption password.

Strategy D is a big step forward over the preceding ones, and is still easy to implement generically using “off-the-shelf” components. To implement it, we need an asymmetric password-based authentication and key exchange (APAKE) protocol. Such protocol involves two parties, a supplicant (the user) who knows a secret password, and a respondent (the server) who holds a secret token derived from the password, but from which the password is hard to reconstruct. At the end of the protocol, both parties get an authenticated random session key for a virtual secure channel.

The full Strategy D can thus be implemented as follows: in the storage phase, the user picks a password and registers with the server both a password-encrypted ciphertext and a password-derived secret token; in the retrieval phase, the two parties use APAKE to set up an authenticated private channel, over which the server then sends the ciphertext back to the user. The benefit is that the user can safely use the same password for authentication and decryption (provided that the two selected schemes can be safely be composed in this manner, which is always possible in the random-oracle model).

Nevertheless, a big problem remains: with its knowledge of the authentication token, the storage facility will have the ability to mount an offline dictionary attack against the password, simply by simulating both sides of the APAKE transactions. Fortunately, the server need not be as trusted as in Strategies B and C, where the server was given either the plaintext itself, or one half of the user passwords in the clear. Also, unlike Strategy A, not everyone but only the server has access to the information needed to mount an offline dictionary attack. Notwithstanding its clear benefits, Strategy D is still not ideal, because it contains an explicit authentication sub-protocol that irremediably grants the server the means to mount a successful (though potentially costly) offline password guessing attack.

Conceivably, if authentication could be kept implicit (to both parties), then the only way for the server to test a password guess would be on the basis of the resulting decrypted plaintext, and distinguishing a valid plaintext from an invalid one might not be easy.

Ideally, the only way for the storage facility to mount an offline guessing attack should be based exclusively on the *finality of the plaintext*: that is, not merely by inspecting a candidate decrypted plaintext, but by attempting to use it in its intended application. Therefore, if the finality of the plaintext happened to involve an online transaction (*e.g.*, authenticating to a third party), the server would not be able to test the validity of a password guess without going online, even if it had the ability to enumerate them all offline.

In other terms, we seek to deprive the storage server of any means to test an offline password guess, other than the inherent redundancy that the correct plaintext might contain (and which is out of the control of the retrieval protocol). If the correct plaintext is uniformly random over the domain of all candidate plaintexts, or at least if the plaintext has significantly fewer bits of redundancy than the entropy of the password, then even with unbounded computational powers it should be impossible for anyone, *even for the storage facility*, to guess the correct plaintext and/or the password in a purely offline attack. The best that the server will be able to do is to draw a list of candidates that must be validated online.

To achieve this, we need a credential retrieval protocol that merges handshake, transfer, and decryption in a single operation that provides no explicit success/failure feedback to either party. On the correct password, the user (alone) receives the correct decrypted plaintext. On an incorrect password, the user merely retrieves a reproducible pseudo-random string that varies with the password.

Our final password-based credential retrieval strategy is thus summarized as follows:

(E) Keep the *unauthenticated* password-encrypted data on a server, and perform the retrieval and decryption in a single *oblivious* password-based protocol with silent failure.

Strategy E can be viewed as an oblivious version of Strategy D, itself a blind version of Strategy C. *I.e.*, the user commits to a password, and accordingly retrieves either the decrypted plaintext or a garbled junktext, with no authentication/integrity check, and no indication of outcome to either party.

When implemented correctly, this strategy offers the best protection against every computational adversary, for every distribution of the password and the plaintext. Indeed:

- It is optimal against outsiders, *i.e.*, attackers other than the server custodian the ciphertext, because outsiders cannot do better than the unavoidable online guessing attack, which is to try one password at a time in an attempt to impersonate one party to the other.

- It is optimal against insiders, *i.e.*, the server entrusted with the ciphertext, because the most that the server can do is to simulate the protocol offline, in order to build from the password dictionary a list of candidate plaintexts. Although this offline attack cannot be avoided, it is only the first step in cracking the ciphertext. The second step is to recognize the correct plaintext among all the candidates; and how hard this can be will depend on the context.

  - On the one hand, if the correct plaintext has enough redundancy that it can be recognized by inspection of the candidate list, then the resulting attack by the server will be completely offline, and there is nothing one can do about it except pick an encryption password that is sufficiently hard to guess.
  - On the other hand, if the correct plaintext is not recognizable by mere inspection, then its appearance on a "short list" of possible candidates may not be very helpful, since the attacker still has to separate the correct plaintext from the other candidates.

    To ensure that this final step is difficult, the retrieval protocol *itself* should not be usable by either party to validate guesses (feedback to the server is clearly harmful, but so is feedback to the client, because in an insider attack the client is impersonated by the server in offline protocol simulations).

  It it for this reason that Strategy E requires that there be no success/failure status leak of any sort. Thus, depending on the application, *e.g.*, when the plaintext is an access key to a third-party online service, even an insider attacker will eventually have to fall back to an exhaustive online search over some candidate list, *i.e.*, by making authentication attempts to such third-party service. Such list of candidates may be very long, even as long as the password dictionary itself, depending on the (lack of) redundancy of the decrypted third-party credentials.

  Clearly, this situation is extremely desirable for security from the user's, *i.e.*, the secret holder's, point of view, assuming low- or no-redundancy secret data. In that case, we would then achieve the paradoxical feat of forcing the *insider* attacker back to an *online* attack scenario, just like any run-off-the-mill outsider.

In summary, Strategy E will always provide the best security among all password-based credential retrieval mechanisms, for all distributions of plaintexts.

We emphasize however that, depending on the circumstances, Strategy E is not necessarily the *only* best strategy among the ones we described, especially in the extremal cases of plaintext distributions:

- On the one end of the spectrum, if we know that the plaintext has zero redundancy, then Strategy A (with an unauthenticated encryption scheme) works equally well, since neither insiders nor thiefs have any advantage over outsiders: this is because without plaintext or ciphertext redundancy the validity of any decryption attempt can only be tested online (*e.g.*, against the third-party service they are intended to grant access to).
- At the opposite end of the spectrum, any time the plaintext (or the password/plaintext combination as a whole) has enough redundancy to be recognizable offline unambiguously when found, Strategy D becomes an equally viable alternative, since there is no longer any reason for the protocol to hide the success of the retrieval outcome to the parties.

In either case, though, Strategy E is always as good as the competing strategy; and furthermore it will be better than all of them in the intermediate cases where the plaintext redundancy either is known to be small but non-zero, or is unknown entirely.

What makes Strategy E superior is thus that it is a one-stop shop: it will be at least as good as the best competing strategy for the circumstances, and generally much more robust than all of them.

1.1 Related Work

In spite of its immediate and compelling applications, *e.g.*, for user-centric internet password security, the notion of Hidden Credential Retrieval appears not to have been formally defined before.

Nevertheless, there exists an extensive body of literature on closely related notions, some general such as MPC, OT, PIR, EKS, PAKE, and some specific such as the Ford-Kaliski server-assisted distributed password protocol [26] or Chaum's and Boldyreva's unique blind signature schemes [21, 9], that approximate, realize, or subsume the HCR functionality.

We shall attempt to give a brief survey of the most germane notions.

MPC: Multi-Party Computation

Like many multi-party cryptographic protocols, HCR may arguably be viewed as a special kind of Multi-Party Computation (MPC). General MPC is a very general tool which allows two or more parties to compute a public function while keeping their respective inputs secret. We mention that MPC finds its roots in Yao's "scrambled circuits" [46]; alternative constructions were later proposed, most notably from OT [30, 37].

Where the characterization of HCR as mere MPC fails, though, 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. For practical use, successful HCR constructions would also have to be orders of magnitude more efficient that general MPC, even without regard for the preceding restriction.

OT: Oblivious Transfer

More adequately perhaps, HCR may also be viewed as a special kind of Oblivious Transfer (OT). The notion of OT was first invented in the quantum world by Brassard and Crépeau (see [6]) and then realized in the classical model by Rabin [44] and further developed in [24, 16] among many others. More precisely, $\mathrm{OT}\,^{N}_{M}$ is a special case of MPC with a sender and a receiver, where the sender holds N secret messages (either bits or strings) and the receiver holds M secret indices between 1 and N. The protocol allows the 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. It is known from [30, 37] that MPC can be constructed from OT_1^4; it is thus a powerful primitive. Security definitions and more refined constructions of OT were subsequently proposed over the years, from [2] to [40, 41, 35, 32] using classic algebraic methods, and more recently [18, 31] using bilinear pairings (see [7, 27]).

Where HCR diverges from OT is that the natural realization of HCR from it would require an OT_1^N protocol with an impractically large value N equal to the size of the password space: one unique string for each possible password, set up by the client.

PIR: Private Information Retrieval

A third notion that is related to HCR is that of Private Information Retrieval (PIR). In short, PIR is a relaxation of OT that only cares about the privacy of the recipient [22], whose queries should remain secret from the sender. An extensive body of research has focused on making PIR efficient for large databases [17, 39], and to add functionalities such as keyword search [38] and private write operations [11].

Notwithstanding the similarities, the notion of PIR fails to provide a suitable HCR for the same reason as OT_1^N above, namely, arising from the need to represent the password-to-plaintext map as an explicit database on the server side, of size linear in the admissible password space.

EKS: Encrypted Keyword Search

Of particular relevance is the notion of Encrypted Keyword Search (EKS), which is a specialized modification of PIR. An EKS scheme is a client-server private retrieval protocol that involves an encrypted corpus of data on the server side, remotely searchable by the client against pre-programmed keywords using encrypted queries. Public-key EKS can be constructed from, and is known to imply, identity-based encryption with certain anonymity properties [10, 1, 14].

Though HCR is even closer in spirit to EKS than it is to PIR or OT, this EKS notion still fails to provide an efficient HCR realization for the same reasons as before: the need for the client first to commit to a manageably sized password space, and then set up the server with one encrypted searchable string per password.

PAKE: Password-Authenticated Key Exchange

In a different way, HCR also draws comparisons to Password-Authenticated Key Exchange (PAKE). First proposed by Bellovin and Merritt [4] under the name Encrypted Key Exchange (EKE), PAKE allows two parties sharing a short password to establish an authenticated secure channel across an adversarially controlled medium. "Augmented" (APAKE) extensions of the same [5] further allow the client to keep the password secret from the server, which is only given a one-way function thereof. This is in order to reduce the risk of password exposure in case of server compromise, in anticipation of people's tendencies to reuse (variations of) the same password in more than one context. Many further extensions have been proposed over the years, concerning, *e.g.*, definitions [3, 15], analysis [8, 20], and PAKE [36, 28, 19] and APAKE [33, 29] constructions.

Asymmetric APAKE protocols in particular have much in common with the HCR notion we propose, though they differ in at least one important point: explicit authentication. Since the main purpose of (A)PAKE protocols is to provide mutual authentication, clearly they must indicate to the parties whether the authentication succeeded or not. Alas, as we already discussed earlier in connection with Strategy D, any such feature can be subverted into an explicit offline password test for the server, and is thus undesirable per the HCR threat model.

Server-Assisted Password Generation

In 2000, Ford and Kaliski proposed a server-assisted password generation protocol, whereby a user wishing to authenticate with a group of servers from a password, would first use the password to retrieve from each server a share of some authentication key, obliviously, and then combine all the shares to obtain the full key.

It is not the only similarity to HCR: the single-server Ford-Kaliski protocol directly realizes HCR with an appropriate redefinition of the players. Indeed, one of the stated goal of the Ford-Kaliski protocol is to deprive any incomplete coalition of servers the ability to perform an offline dictionary attack against the user password, much as HCR does with respect to the storage server.

The main conceptual difference is that Ford-Kaliski specifies the password-based reconstitution of a zero-redundancy random key for authentication within the same pool of servers, whereas HCR seeks the password-based retrieval of an arbitrary low-redundancy plaintext from a single server for unspecified purposed with an unspecified third party.

Unique Blind Signatures

In 1982, Chaum proposed the notion of blind signature protocols [21], whereby signatures can be made blindly on committed messages, and later unblinded by the original requestor then subsequently verified by anyone, without the help of the signer. The original construction from [21] was based on RSA, and was in fact a unique blind signature, meaning that the signature is entirely determined by the signing key and the unblinded message. In 2003, Boldyreva proposed a very efficient construction of a unique blind signature [9], based on the bilinear pairings. Both constructions can be proven secure in the random-oracle model from suitable complexity assumptions.

The connection with our concern is that unique blind signature protocols can easily be transformed into HCR protocols, in the random-oracle model, by letting the message be the password, and the hash of the signature be the encryption key for the plaintext of the credentials to be stored. Unique blind signatures are in fact strictly more powerful, since they come with a public verification function that has no counterpart in HCR.

Coming full circle, one of the Ford-Kaliski protocol instantiations was described in [26] as a direct application of Chaum's blind signature, whereas the other instantiation is a close cousin of Boldyreva's signature (by comparison, the Ford-Kaliski construction seems to take a shortcut, which may affect its security reducibility to the same assumption).

"Entropic" Encryption

Earlier we saw that *low-redundancy* messages such as keys are the main type of applications targeted by HCR. For completeness, we mention the closely related notion of encryption for *high-entropy* messages.

For the same reason that it may be difficult to recognize the correct plaintext in a guessing attack against HCR, it can be shown that encrypting high-entropy messages is generally easier than arbitrary messages if the encryption key is short, because it is the total entropy of the key and message that a potential attacker will have to overcome. See for example [45], [23], and the references therein. Of course, those notions hearken all the way back to the ideas of Shannon, whose one-time pad represents a bright spot all the way on one side of the message-*vs.*-key-entropy spectrum.

Although HCR operates on a similar principle, the main difference is that in HCR we care about protecting the password as much (or perhaps even more) than the plaintext itself (since the same password may protect more than one plaintext). To take an extreme example, a good HCR scheme should maintain its composure even if the plaintext is the empty string (surely a zero-entropy message, but also a zero-redundancy one at the same time), whereas most notions of encryptions trivially break down in this situation.

Halting Passwords

For completeness, we mention some recent progress in stand-alone password-based encryption, provided by the Halting Key Derivation Functions (HKDF) from [13]. There, the author shows that security gains can be achieved for short passwords by using a very expensive password-to-key derivation function, and whose expense parameter is furthermore cryptographically hidden from view. Such parameter is chosen by the encryptor on a case-by-case basis, and need not be remembered.

One might think that HKDFs would be useful in our context to make redundant plaintexts much more difficult to recognize than they would normally be, *e.g.*, by applying the HKDF to the plaintext before setting up the HCR using the same password. However, their feature of halting only on the correct password is technically a liability in our context, as it could be exploited by an insider adversary to provide offline password validation (albeit a very and unpredictably expensive one), which is something we try to avoid with HCR.

Practical Security Systems

We conclude this tour by mentioning a number of system architectures that seek to leverage a weak password into strong and/or reusable credentials, principally for single sign-on applications on the internet.

Some of them [43, 25] rely on external servers with varying trust requirements, perhaps requiring the user to carry a list of one-time tokens that are translated into usable passwords by a proxy web-site [42].

Other approaches [34] seek to confuse the attacker by giving the appearance of many valid decryption passwords, by making the true plaintext hard to recognize.

2 Definitions

We now give a precise definition of the HCR primitive, and formalize the security requirements.

2.1 Abstract Primitive

The Hidden Credential Retrieval model involves three entities: a preparer $\mathcal{P}$, a querier $\mathcal{Q}$, and a server $\mathcal{S}$.

The first two entities ($\mathcal{P}$ and $\mathcal{Q}$) embody the user during the preparation and the retrieval phases of the protocol, *i.e.*, they embody the same person but at different times along the life of the protocol, to capture the possibility of a "memory loss" incurred by the user.

The third entity ($\mathcal{S}$) embodies the remote storage facility, which has unrestricted amounts of persistent memory, and is of course distinct from the user.

HCR then consists of the following two protocols:

Store : $\langle \mathcal{P}[\mathsf{Pwd}, \mathsf{Msg}], \mathcal{S}[\not\phi] \rangle \mapsto \langle \mathsf{Ctx}, \mathsf{Ctx} \rangle$
done once initially, over a secure channel

The *Store* protocol involves the user acting as preparer $\mathcal{P}$ and the selected storage server $\mathcal{S}$; its purpose is to set up the parties' long-term secrets, especially the server's.

To bootstrap our storage and retrieval protocol, the client must have selected a server with which to do business, and must be able to communicate securely with it for the initial setup: we model this the usual way by requiring an authentic private channel for this storage phase only.

- The preparer $\mathcal{P}$ takes as two private inputs: a memorable password Pwd and a plaintext credential Msg.
- The server $\mathcal{S}$ takes no private input, denoted here by the null symbol $\not\phi$.

At the end of the exchange, $\mathcal{S}$ (and possibly $\mathcal{P}$) will have acquired a randomized private credential ciphertext Ctx.

Note: Ctx is intended for $\mathcal{S}$ alone. $\mathcal{P}$ can learn it too, but nobody else should. The point of requiring authenticated private communications in the storage phase is so that $\mathcal{P}$ can limit the disclosure of Ctx to the server $\mathcal{S}$ it trusts to act as an insider. By definition, the knowledge of Ctx is what will separate an "insider" from an "outsider".

Retrieve : $\langle \mathcal{Q}[\mathsf{Pwd}'], \mathcal{S}[\mathsf{Ctx}'] \rangle \mapsto \langle \mathsf{Msg}', \not\phi \rangle$
can be repeated, over adversarial channels

The *Retrieve* protocol is an exchange between the user acting as querier $\mathcal{Q}$ and the server $\mathcal{S}$. It may be conducted any number of times, and make no presumption about the security of communication, or the parties' identities.

- The querier $\mathcal{Q}$ takes one private input: a password Pwd'.
- The server $\mathcal{S}$ takes one private input: a ciphertext Ctx'.

Upon completion of the exchange, $\mathcal{S}$ has learned $\not\phi$, or nothing at all; whereas $\mathcal{Q}$ has obtained a plaintext Msg' that is a deterministic function of both parties' inputs. Msg' must satisfy the following condition w.r.t. the inputs used by $\mathcal{P}$ and $\mathcal{S}$ in the previous *Store* protocol:

$$(\mathsf{Pwd}' = \mathsf{Pwd}) \wedge (\mathsf{Ctx}' = \mathsf{Ctx}) \Rightarrow (\mathsf{Msg}' = \mathsf{Msg}) \ .$$

We stress that neither $\mathcal{S}$ nor $\mathcal{Q}$ learns from this protocol whether $\mathcal{Q}$ retrieved the correct Msg.

2.2 Informal Threat Model

Before we formally define the security requirements of Hidden Credential Retrieval, it is useful to enumerate informally all the possible threats one by one, to better motivate the model. The security goals we seek to defeat the various threats are as follows.

1. *Total security against passive eavesdroppers:* Passive observers that do not participate in the protocol should gain no computational advantage in recovering Ctx much less Msg or Pwd from observing arbitrarily many protocol execution transcripts between the two honest players $\mathcal{Q}$ and $\mathcal{S}$.

2. *Online security against active outsiders:* An external attacker that takes an active part in the protocol, *e.g.*, by impersonating $\mathcal{Q}$ or $\mathcal{S}$, or by modifying messages between $\mathcal{Q}$ and $\mathcal{S}$, should not be able to learn anything other than whether a particular password guess Pwd' is correct or not (with at most one guess tested per protocol execution).

3. *Offline security against insiders:* The server $\mathcal{S}$, though entrusted with the ciphertext Ctx, should not be able to recover the corresponding plaintext Msg more efficiently than by conducting a brute-force offline dictionary attack against the encryption password Pwd. This must hold even though $\mathcal{S}$ may be involved in arbitrarily many protocol executions with the user $\mathcal{Q}$ who knows the correct password.

4. *Absence of any spurious induced validity test:* The retrieval protocol itself must not be subvertible into providing an offline password validity test that is not already implicitly present in the message Msg (whether that is the case depends on the plaintext distribution and is thus application-specific). In other words:

   - the retrieval protocol should be *blind*, *i.e.*, keep the password invisible to the server;
   - the retrieval protocol should be *oblivious*, *i.e.*, not disclose its success to either party;
   - the Pwd-based encrytion of Msg into Ctx has to be *redundancy-free* (and thus not carry any integrity check), *i.e.*, the only information-theoretic redundancy in Ctx should come from Msg itself.

   Under those conditions, if the plaintext Msg is intrinsically unrecognizable in itself, *e.g.*, being a random access key for a separate third-party service, then it will be impossible even for the insider server to recover the password (or the plaintext) in an offline dictionary attack.

We stress that the server $\mathcal{S}$ is only *partially trusted*, since we merely ask it to maintain a private copy of the blinded ciphertext Ctx (a "secrecy" requirement), and participate in the retrieval protocol with anyone who asks posing as the client $\mathcal{Q}$ (an "availability" requirement). We make no strong security assumption on it, and explicitly allow it to misbehave.

We also stress that, by contrast, the user is ultimately trusted, since all the data is his. Hence, it is pointless to contemplate corruptions of either $\mathcal{P}$ or $\mathcal{Q}$ (even though impersonations should of course be considered).

2.3 Formal Security Requirements

We formalize the preceding security requirements using a game-based definition that captures all the properties we need.

We actually define two games, since there are two possible (and mutually exclusive) adversaries: the first is an "outsider game", pitting the user and server against and external attacker; the second is an "insider game", pitting the user against a dishonest server. As we mentioned, it is pointless to consider a dishonest user, since all the valuable data ultimately belongs to the user. Our two security games thus capture the notion that the message Msg (and the password Pwd) must remain private to the user, under passive and active attacks by an outsider and/or by a dishonest server.

2.3.1 Quantifying the Adversary's Work

This model presents a technical conundrum, however. Supposing that the plaintext Msg contains enough redundancy for an offline attack to be possible in principle, how can we argue that such attack must be as expensive as the task of enumerating the entire password dictionary (or half of it on expectation), when all the work for that enumeration is spent internally by the adversary, out of view of the outside world? Surely, it would be nice to show from first principles that the complexity of computing Msg and Pwd from Ctx alone is exponential in the password bit-length $|\mathsf{Pwd}|$ (assuming that Msg brings enough intrinsic redundancy for the pair $\langle \mathsf{Msg}, \mathsf{Pwd} \rangle$ to be uniquely determined by Ctx), but this would imply a separation $\mathbf{P} \neq \mathbf{NP}$.

As our ambitions in this paper are more modest, we shall instead consider such lower bound relative to a *validity check oracle*. In this model, the attacker will not know *a priori* the set $\mathbf{M} \subseteq \{0,1\}^k$ of admissible plaintexts from which the correct message Msg is to be drawn uniformly. The adversary will have to make a call to a test oracle f_1 to determine whether a given string belongs to $\mathbf{M}$.

We cannot emphasize enough, though, that f_1 is a gimmick that we use in our games for accounting purposes. In reality, the adversary may very well know what $\mathbf{M}$ is, and thus be able to make the same determination as f_1 in a purely offline manner.

2.3.2 Modeling Online and Offline Validity Tests

In addition to f_1 which lets us quantify an important aspect of the adversary's offline work, we also introduce a second predicate f_2 to model the process by which the adversary can truly validate a candidate plaintext by performing an expensive online check.

More precisely, the oracle $f_2[\mathsf{Msg}']$ models the real-world action of trying to use the decrypted candidate credential Msg' in lieu of the correct credential Msg for its intended purpose, *i.e.*, it models the attempt to impersonate the user by presenting the credential Msg' to the appropriate third party. Generally, this will only succeed if $\mathsf{Msg}' = \mathsf{Msg}$, and thus $f_2[\mathsf{Msg}']$ models a very expensive but accurate test for $\mathsf{Msg}' \overset{?}{=} \mathsf{Msg}$.

The two "message validity test" oracles f_1 and f_2 thus have the following semantics:

- The first oracle, f_1, captures the offline recognition of a potentially valid plaintext on the basis of its intrinsic redundancy: $f_1[\mathsf{Msg}'] = 1$ means that Msg' is well-formed, *i.e.*, it is in the set $\mathbf{M}$, though it is not necessarily correct.

- The second oracle, f_2, models an expensive but perfectly accurate validity check, often requiring an online component and the cooperation of a third party: $f_2[\mathsf{Msg}'] = 1$ indicates that Msg' is usable in lieu of the correct Msg with the third party, and thus typically that Msg' is the correct Msg.

These oracles are always defined relative to a specific reference ciphertext Ctx, the target of the attack, which will be clear from context. More summarily:

$$f_1 : \{0,1\}^k \to \{0,1\} : \mathsf{Msg} \mapsto \text{“Is } \mathsf{Msg} \text{ well-formed?”} \ ,$$
$$f_2 : \{0,1\}^k \to \{0,1\} : \mathsf{Msg} \mapsto \text{“Is } \mathsf{Msg} \text{ the answer?”} \ .$$

This model strikes a good balance between flexibility and simplicity in our quest to characterize a plaintext's *intrinsic redundancy* and *extrinsic validity* that an attacker may be able to exploit.

2.3.3 Outsider Security

We first 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, *i.e.*, $\mathcal{Q}$ and $\mathcal{S}$ (recall that $\mathcal{P}$ only interacts with $\mathcal{S}$ over a secure channel, so we do not consider $\mathcal{P}$). The adversary is an outsider; it makes passive requests for transcripts of legitimate executions of *Retrieve* between $\mathcal{Q}$ and $\mathcal{S}$; it can also actively impersonate $\mathcal{Q}$ to $\mathcal{S}$, or $\mathcal{S}$ to $\mathcal{Q}$, by interfering with the flows in concurrent but independent protocol executions. (What the outsider adversary cannot do is corrupt or read the internal state of any of the actual players.)

The outsider attack game thus proceeds as follows:

Initialization. $\mathcal{B}$ privately simulates an execution of the *Store* protocol between $\mathcal{P}$ and $\mathcal{S}$, for a random password $\mathsf{Pwd} \in \{0,1\}^n$ and a random message $\mathsf{Msg} \in \{0,1\}^k$.
We assume that the distribution of Msg is uniform over some subset $\mathbf{M} \subseteq \{0,1\}^k$, such that $\forall m \in \{0,1\}^k : m \in \mathbf{M} \iff f_1[m] = 1$. $\mathbf{M}$ 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 f_1-oracle if it wants to test candidate messages for membership to $\mathbf{M}$. (In the real world, $\mathbf{M}$ will likely be public, allowing the adversary to run f_1 privately.)

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 $\mathsf{Pwd}' = \mathsf{Pwd}$.

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

Offline validity tests. $\mathcal{A}$ can make adaptive calls to the offline predicate f_1 on any string of its choice. The response indicates whether the string belongs in $\mathbf{M}$.

Online validity tests. $\mathcal{A}$ can make adaptive calls to the online predicate f_2 on any string of its choice. The response indicates whether the string is the correct message Msg.

Message guess. $\mathcal{A}$ eventually outputs one guess $\widehat{\mathsf{Msg}}$ for the value of Msg.

Adjudication. The adversary wins if $\widehat{\mathsf{Msg}} = \mathsf{Msg}$.

Definition 1. *The advantage of an adversary $\mathcal{A}$ in a (p, q, t_1, t_2)-outsider attack is defined as the probability that $\mathcal{A}$ wins the preceding game, when $\mathcal{A}$ makes a total of p passive eavesdropping queries, q active impersonation queries, and t_1 and t_2 calls to f_1 and f_2 respectively.*

2.3.4 Insider Security

We now define the privacy model against insider attacks. Since the user embodied by $\mathcal{P}$ and $\mathcal{Q}$ is ultimately trusted, the only potentially malicious insider is $\mathcal{S}$ (or any entity that has managed to acquire Ctx from $\mathcal{S}$, and which is thus equivalent to $\mathcal{S}$).

The model is based on a game played between a malicious server $\mathcal{A}_\mathcal{S}$ and a challenger $\mathcal{B}$ (where the subscripted $\mathcal{A}_\mathcal{S}$ is a reminder that the adversary is the server itself). The challenger simulates the trusted user in the storage and retrieval phases of the HCR protocol, *i.e.*, $\mathcal{P}$ and $\mathcal{Q}$. The attack proceeds in two phases: the first phase involves a single execution of the *Store* protocol between $\mathcal{P}$ and $\mathcal{A}_\mathcal{S}$; the second phase involves as many independent executions of the *Retrieve* protocol between $\mathcal{Q}$ and $\mathcal{A}_\mathcal{S}$ as the adversary desires, and in which $\mathcal{Q}$ will use the correct password $\mathsf{Pwd}' = \mathsf{Pwd}$. (Note that if the server wishes to execute the protocol with someone using a different password, it can simulate the protocol execution all by itself, so we need not consider those.) As before the task of $\mathcal{A}_\mathcal{S}$ is to recover the value of Msg.

The insider attack game proceeds as follows:

Storage interaction. $\mathcal{B}$, acting on behalf of $\mathcal{P}$, picks a random password $\mathsf{Pwd} \in \{0,1\}^n$ and a random message $\mathsf{Msg} \in \mathbf{M} \subseteq \{0,1\}^k$, and engages in the *Store* protocol with the adversary $\mathcal{A}_\mathcal{S}$.

The distribution of Msg is uniform over the subset $\mathbf{M} \subseteq \{0,1\}^k$, which as before is the cover set of the predicate f_1; it is a parameter of the game but is not given to $\mathcal{A}$. This is to force $\mathcal{A}$ to make accountable calls to an f_1-oracle in order to determine whether a message is well-formed. (In the real world, the adversary would know $\mathbf{M}$ and be able to run f_1 offline locally.)

Retrieval interactions. $\mathcal{B}$, acting on behalf of $\mathcal{Q}$, initiates the *Retrieve* protocol with the adversary $\mathcal{A}_\mathcal{S}$, using the correct access password $\mathsf{Pwd}' = \mathsf{Pwd}$. This may happen multiple times, adaptively, at the request of $\mathcal{A}_\mathcal{S}$.

Offline validity tests. $\mathcal{A}$ makes adaptive calls to the offline predicate f_1 on any chosen string. The response indicates whether the string is in the set $\mathbf{M}$.

Online validity tests. $\mathcal{A}$ makes adaptive calls to the online predicate f_2 on any chosen string. The response indicates whether the string is the correct Msg.

Message guess. $\mathcal{A}$ eventually outputs its guess $\widehat{\mathsf{Msg}}$ for the value of Msg.

Adjudication. The adversary wins if $\widehat{\mathsf{Msg}} = \mathsf{Msg}$.

Definition 2. *The advantage of an adversary $\mathcal{A}$ in a (r, t_1, t_2)-insider attack is defined as the probability that $\mathcal{A}$ wins the preceding game, after a total of r initiated instances of the* Retrieve *protocol, and a total of t_1 and t_2 oracle calls to f_1 and f_2 respectively.*

2.3.5 Recovery *vs.* Distinguishing Challenges

A peculiarity of the two preceding games is that we ask the adversary to recover the full plaintext Msg and not merely recognize it in a left-*vs.*-right distinguishing challenge, which would correspond to the usual notion of semantic security for encryption schemes. The reason why a distinguishing challenge is inadequate is that, given Msg_1 and Msg_2 such that one of them is the true Msg, a single call to the online oracle f_2 will immediately solve the challenge, and two calls to the offline oracle f_1 are likely to solve it too (if f_1 returns 0 on either plaintext, we know that the correct answer is the other one.)

The root of the discrepancy is that in the traditional context of encryption schemes, we seek to protect the plaintext from semantic leaks caused entirely by the encryption scheme itself. In the envisioned applications of HCR, the plaintext has no intrinsic semantic value; it is an arbitrary string to be used as an authenticator to a third-party remote service. Thus, in our case, the ultimate goal of the adversary is not to compute the actual plaintext Msg, or decide which one of Msg_1 or Msg_2 might have been committed to storage, but to gain access to whatever service the owner of the plaintext is entitled to. If the adversary knows that either Msg_1 or Msg_2 will grant such access, trying them both will not present much of a challenge, whereas recovering a randomly chosen string $\mathsf{Msg} \in \mathbf{M}$ still would.

For all these reasons, we contend that the preceding definitions provide the simplest model that captures the essential security requirements of an HCR protocol for its intended application.

Remark also that applications that require semantic security in the traditional sense can easily be accommodated, using standard amplification techniques from one-way to semantic security (*e.g.*, by hashing Msg with an appropriate function, and invoking either the left-over hash lemma or the random-oracle model).

3 Realization

We observed when discussing related works that HCR and protocols that subsume it may already be found in the literature, though sometimes addressing a completely different problem.

We already alluded to a generic transformation from unique blind signature protocols into HCR protocols in the random-oracle model, and indeed the three most practical examples of constructions to date that are suitable for HCR are related to unique blind signatures: first is the Chaum blind signature [21], second is the Ford-Kaliski server-assisted password generation protocol [26], and third is the Boldyreva blind signature [9].

For the same of illustration, we focus on the Boldyreva signature, since it is very efficient and comes with a security reduction (albeit to a non-standard assumption in the random-oracle model). The resulting HCR scheme is directly usable in practice.

3.1 From Unique Blind Signatures

First, to fix ideas, we briefly explain how (suitably secure) unique blind signatures already realize, and in fact subsume, the HCR functionality in the random-oracle model. To do so, we sketch how to construct the HCR protocols *Store* and *Retrieve* using only a hash function and the signature's blind signing and unblinding functions (while noting that the verification function is not needed).

Store. The user does the following, interacting with the server over a secure channel:

1. Generate a random signing private key κ for the blind signature scheme (a public verification key is not needed).
2. Hash the password Pwd into a string $\mu = H_1[\mathsf{Pwd}]$ and compute its (deterministic) signature $\sigma = \mathit{Sign}[\kappa : \mu]$.
3. Hash the signature into a one-time pad $\pi = H_2[\sigma]$ and use it as a mask for Msg in $\gamma = \mathsf{Msg} \oplus \pi$.
4. Give the key κ and the string γ to the server. The pair $\langle \kappa, \gamma \rangle$ acts as the HCR ciphertext Ctx, though only κ ought to remain secret.

Retrieve. The user does the following, interacting with the server over any channel:

1. Hash the password Pwd' into a string $\mu' = H_1[\mathsf{Pwd}']$

2. Perform the blind signature protocol with the server and unblind the result to obtain a signature σ' on μ'.
3. Also request the string γ back from the server. Let γ' be the received copy.
4. Hash σ' to obtain a one-time pad $\pi' = H_2[\sigma']$ and use it to unmask γ' to get the result $\mathsf{Msg}' = \gamma' \oplus \pi'$.

3.2 A Concrete Construction

Based on the foregoing, we construct a simple HCR scheme using the Boldyreva blind signature as a starting point.

The Boldyreva blind signature actually requires a bilinear pairing for its implementation, but that is because the pairing is needed for signature verification. The signing function alone is sufficient to construct an HCR protocol, therefore our construction will not need a pairing (though it nonetheless remains compatible with pairing-friendly groups). The Boldyreva signature is derived from the Boneh-Lynn-Shacham short signature scheme of [12], it is very efficient, but it requires a non-standard discrete-log-type hardness assumption for its security reduction, in the random-oracle model.

We modify that scheme to build a concrete HCR protocol in prime-order abelian groups under the same assumption (though without the pairing requirement). The concrete protocol is very simple, very efficient, and can be implemented in a broad variety of discrete-log-hard prime-order abelian groups, including those constructed on elliptic curves (pairing-friendly or not, such as those standardized by NIST for elliptic-curve ECDSA), as well as multiplicative subgroups of finite fields as in the original DSA standard.

3.2.1 Protocol Description

Let $\mathbb{G}$ be a (multiplicatively written) cyclic abelian group of order p, for some cryptographically large public prime p. Let $\mathbb{G}^\times = \mathbb{G} \setminus \{1_\mathbb{G}\}$ be the set of all non-neutral elements in $\mathbb{G}$. Let then $H : \{0,1\}^n \to \mathbb{G}^\times$ be a cryptographic hash function, which is to be viewed as a random oracle.

A concrete HCR scheme can therefore be constructed as follows:

Store : $\langle \mathcal{P}[\mathsf{Pwd}, \mathsf{Msg}], \mathcal{S} \rangle \mapsto \langle \not\phi, \mathsf{Ctx} \rangle$
where $\mathsf{Pwd} \in \{0,1\}^n$ and $\mathsf{Msg} \in \mathbb{G}$

1. $\mathcal{P}$ picks an arbitrary (*e.g.*, random or fixed) generator $g \in \mathbb{G}$.
2. $\mathcal{P}$ selects a random integer $s \in \{1, \ldots, p-1\}$, and lets $h = g^s$.
3. $\mathcal{P}$ computes the signature $\sigma = H[\mathsf{Pwd}]^s$ and sets $\gamma = \sigma^{-1}\,\mathsf{Msg}$.
4. $\mathcal{P}$ hands to $\mathcal{S}$ the public values g, h, γ and the secret key s.

Retrieve : $\langle \mathcal{Q}[\mathsf{Pwd}'], \mathcal{S}[\mathsf{Ctx}'] \rangle \mapsto \langle \mathsf{Msg}', \not\phi \rangle$
where $\mathsf{Pwd}' \in \{0,1\}^n$

1. $\mathcal{Q}$ retrieves from $\mathcal{S}$ the public values g, h, γ.
2. $\mathcal{Q}$ picks a random integer $r \in \{0, \ldots, p-1\}$ and sends the blind request $\mu = g^r\, H[\mathsf{Pwd}']$.
3. $\mathcal{S}$ responds with the blind signature $\beta = \mu^s$.
4. $\mathcal{Q}$ unblinds the signature $\sigma' = h^{-r}\,\beta$, and decrypts the message $\mathsf{Msg}' = \sigma'\,\gamma$.

3.2.2 Security Properties

In order to quantify the adversaries' success probality, we need some additional notation.

Recall that t_1 and t_2 are the number of unique valid queries to f_1 and f_2 respectively. We define n_1 and n_2 as the number of *negative* responses to those queries, so that $n_1 \leq t_1$ and $n_2 \leq t_2$ (and $n_2 \geq t_2 - 1$).

For simplicity of the arithmetic, we shall also assume that each query to f_2 is always preceded by an identical query to f_1, and that the query to f_2 will not be issued (or counted) if the answer from f_1 was already negative. Furthermore we assume that queries f_1 then f_2 are systematically made on the final guess Msg' output by $\mathcal{A}$.

We also suppose that the password Pwd is always drawn uniformly from a public dictionary D, and that, in the view of the adversary, the prior distribution of the plaintext Msg 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, in the security definitions, Msg is drawn from a subset $\mathbf{M}$ about which the adversary has no prior information other than $\mathbf{M} \subseteq \{0,1\}^k$.

Proposition 3 (Outsider Security). *In this setting, suppose that the complexity assumption from [9] holds in $\mathbb{G}$ for the chosen security parameter λ, for the class of all PPT algorithms running in time T. Then, no (p, q, t_1, t_2)-outsider adversary $\mathcal{A}$ running in time T can recover the stored message $\mathsf{Msg} \in \mathbf{M} \subseteq \{0,1\}^k$ with a probability that exceeds the following bound:*

$$\Pr[\mathcal{A}^{f_1,f_2}\, wins] \leq \frac{\min\{q, t_2\}}{|D| - \min\{q, n_1\}} + \frac{t_2}{2^k - n_1} + \text{negl}[\lambda] \ .$$

Proposition 4 (Insider Security). *In the above setting, in the random-oracle model without any computational hardness assumption, every (r, t_1, t_2)-insider adversary $\mathcal{A}_{\mathcal{S}}$ that recovers the stored message $\mathsf{Msg} \in \mathbf{M} \subseteq \{0,1\}^k$ while making no more than o random-oracle queries, succeeds with probability at most:*

$$\Pr[\mathcal{A}_{\mathcal{S}}^{f_1,f_2}\, wins] \leq \frac{\min\{o, t_2\}}{|D| - \min\{o, n_1\}} + \frac{t_2}{2^k - n_1} \ .$$

As one might expect, the unconditional bound stated in Proposition 4 for insider attacks applies to outsider attacks as well, since the insider can always simulate an outsider attack against the pair formed by itself and any user. Nevertheless, the purpose of Proposition 3 is to show that the outsider bound is even stronger when the group $\mathbb{G}$ upholds the complexity assumption from [9] (therein called the Chosen-Target Computational Diffie-Hellman assumption [9]).

The outsider bound in stronger because in reality the number q corresponds to expensive man-in-the-middle attacks and is likely to be small compared to any number o of random-oracle hash functions calculations (though t_2 may be even smaller, depending on the application).

Notice also that the parameters p and r do not intervene into the stated success probability bounds. This makes sense, since p is the number of sessions being passively eavesdropped upon in an outsider attack, which look random to a computationally bounded adversary. The parameter 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 protoocol.

4 Conclusion

We have proposed the notion of Hidden Credential Retrieval, a class of protocols that lets the rest of us — people with limited memory — defer to remote entities the task of storing our access keys to other online services, whilst placing the minimum possible amount of trust on the storage facility and no trust whatsoever on the communication channels.

Although the solution appears quite simple in retrospect, there were a few interesting issues regarding the handling of the threat model and the security definitions, for payload plaintexts that may benefit from having little or no intrinsic redundancy. In particular, HCR protocols have to handle the combined threat of outsider and insider attackers, both of which must be repelled in the best possible way, regardless of the offline testability of the access keys that is being placed in remote storage.

Our main contribution was to identify the problem formally, and propose two security models for the two types of adversaries we must consider. The models are unusual for a cryptographic protocol, because in our application the plaintext itself may be difficult to recognize due to its possible lack of redundancy. To capture this feature, we equipped the model with a couple of oracle test predicates, which the adversary must call in order to determine the plausible or exact validity of any candidate decryption. These oracles let us quantify the work of an attacker that otherwise could be entirely offline.

We then proposed a very simple though very efficient construction of HCR from a discrete-log-based hardness assumption in plain prime-order algebraic groups. The construction itself is not new, unlike its analysis. The generic construction is based on the very old notion of unique blind signatures, and its concrete implementation is a straightforward modification of Boldyreva's efficient unique blind signature, and closely related to the Ford-Kaliski password generation scheme. On the downside, it suffers from relying on a strong computational assumption in the random-oracle model. On the upside, our 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. Of course, other trade-offs are possible, and we leave it as an open problem to motivate and devise efficient HCR schemes with different properties, and/or from weaker assumptions.

The notion of HCR is important because it has a very clear and immediate application for the internet: a "remote key locker" that can be operated with a short password, and yet does not require the remote server to be trusted. In one sentence, what HCR does for all of us, users of the net, is to provide a sound answer to the simple question,

How can I store all of my net passwords and retrieve them safely on the net itself?

We leave it as a theoretical open problem to refine the HCR security model (perhaps by modeling the redundancy of the user-selected plaintext in a better way), as well as to construct an efficient HCR scheme from reasonable assumptions in the standard model.

Acknowledgments

The author would like to thank the anonymous referees of ASIACCS 2009 for valuable comments.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology—CRYPTO 2005*, pages 205–22, 2005.

[2] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In *Advances in Cryptology—CRYPTO 1989*, pages 547–57, 1989.

[3] Mihir Bellare, David Pointcheval, and Philip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology—EUROCRYPT 2000*, pages 139–55, 2000.

[4] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symposium on Security and Privacy—SP 1992*, pages 72–84, 1992.

[5] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange. In *ACM Conference on Computer and Communications Security—CCS 1993*, pages 224–50, 1993.

[6] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer. In *Advances in Cryptology—CRYPTO 1991*, pages 351–66, 1991.

[7] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, editors. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2005.

[8] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *Proceedings of the Sixth IMA International Conference on Cryptography and Coding*, pages 30–45, 1997.

[9] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *Public Key Cryptography—PKC 2003*, pages 31–46, 2003.

[10] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology—EUROCRYPT 2004*, pages 506–22, 2004.

[11] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and W. Skeith. Public key encryption that allows PIR queries. In *Advances in Cryptology—CRYPTO 2007*, pages 50–67, 2007.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology—ASIACRYPT 2001*, pages 514–32, 2001.

[13] Xavier Boyen. Halting password puzzles – hard-to-break encryption from human-memorable keys. In *16th USENIX Security Symposium—SECURITY 2007*, pages 119–134. The USENIX Association, 2007.

[14] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology—CRYPTO 2006*, pages 290–307, 2006.

[15] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology—EUROCRYPT 2000*, 2000.

[16] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology—CRYPTO 1986*, pages 234–38, 1986.

[17] Christian Cachin, Silvio Micali, and Michael Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology—EUROCRYPT 1999*, pages 402–14, 1999.

[18] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In *Advances in Cryptology—EUROCRYPT 2007*, pages 573–90, 2007.

[19] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology—EUROCRYPT 2005*, pages 404–21, 2005.

[20] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology—EUROCRYPT 2001*, pages 453–74, 2001.

[21] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology—CRYPTO 1982*, pages 199–203, 1982.

[22] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science—FOCS 1995*, pages 41–51, 1995.

[23] Yevgeniy Dodis and Adam Smith. Entropic security and the encryption of high-entropy messages. In *Theory of Cryptography Conference—TCC 2005*, 2005.

[24] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology—CRYPTO 1982*, pages 205–10, 1982.

[25] D. Florencio and Cormac Herley. Klassp: Entering passwords on a spyware infected machine using a shared-secret proxy. In *Proc. ACSAC 2006*, 2006.

[26] Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *Proc. IEEE 9th Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 176–80. IEEE Press, 2000.

[27] Steven Galbraith, Kenneth Paterson, and Nigel Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006.

[28] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. Password authenticated key exchange using hidden smooth subgroups. In *ACM Conference on Computer and Communications Security—CCS 2005*, pages 299–309. ACM Press, 2005.

[29] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In *Advances in Cryptology—CRYPTO 2006*, LNCS, pages 142–59. Springer-Verlag, 2006.

[30] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on Theory of Computing—STOC 1987*, pages 218–29, 1987.

[31] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *Advances in Cryptology—ASIACRYPT 2007*, 2007.

[32] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. Cryptology ePrint Archive, Report 2007/118, 2007.

[33] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. In *ACM Conference on Computer and Communications Security—CCS 1998*, pages 122–31. ACM Press, 1998.

[34] D. N. Hoover and B. N. Kausik. Software smart cards via cryptographic camouflage. In *IEEE Symposium on Security and Privacy—SP 1999*, 1999.

[35] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology—CRYPTO 2004*, pages 335–54, 2004.

[36] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology—CRYPTO 2001*, 2001.

[37] Joe Kilian. Founding cryptography on oblivious transfer. In *ACM Symposium on Theory of Computing—STOC 1988*, pages 20–31, 1988.

[38] Kaoru Kurosawa and W. Ogata. Oblivious keyword search. *Journal of Complexity*, 20(2-3):356–71, 2004.

[39] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. Cryptology ePrint Archive, Report 2004/063, 2004.

[40] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *Advances in Cryptology—CRYPTO 1999*, pages 573–90, 1999.

[41] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Symposium on Discrete Algorithms—SODA 2001*, pages 448–57, 2001.

[42] Andreas Pashalidis. `http://www.kyps.net`, 2007.

[43] Andreas Pashalidis and Chris J. Mitchell. Impostor: a single sign-on system for use from untrusted devices. In *Proc. IEEE Globecom 2004*, 2004.

[44] Michael Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, Aiken Computation Laboratory, 1981.

[45] A. Russell and Y. Wang. How to fool an unbounded adversary with a short key. In *Advances in Cryptology—EUROCRYPT 2002*, 2002.

[46] Andrew Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science—FOCS 1986*, pages 162–67, 1986.