Outsourced Bits

A research blog on cloud computing, cryptography, security, privacy, …

How to Search on Encrypted Data: Functional Encryption (Part 3)

This entry was posted on October 30, 2013, in Crypto Design, Encrypted Search, Privacy and tagged cloud storage, encrypted search, functional encryption, identity-based encryption, privacy, search on encrypted data, searchable encryption. Bookmark the permalink. 13 Comments

(https://cloudcrypto.files.wordpress.com/2013/10/sse.jpg)

This is the third part of a series on searching on encrypted data. See parts 1 (http://outsourcedbits.org/2013/10/06/how-to-search-on-encrypted-data-part-1/), 2 (http://outsourcedbits.org/2013/10/14/how-to-search-on-encrypted-data-part-2/) and 4 (http://outsourcedbits.org/2013/12/20/how-to-search-on-encrypted-data-part-4-oblivious-rams/).

Previously, we covered the simplest solution for encrypted search which consisted of using a deterministic encryption scheme (more generally, using a property-preserving encryption scheme) to encrypt keywords. This resulted in an encrypted search solution with sub-linear (in n) search time but that leaked quite a bit of information to the server.

We' ll now describe a different approach that provides the opposite properties: slow search but better security. At a high-level, one can view this approach as simply replacing the PPE scheme in the previous solution with a *functional encryption* (FE) scheme.

Functional and Identity-Based Encryption

The notion of FE was first described by Sahai and Waters in a talk [SW09 (http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt)] and later formalized by Boneh, Sahai and Waters [BSW10 (http://eprint.iacr.org/2010/543.pdf)] and by O'Neill [O'Neill10 (http://eprint.iacr.org/2010/556)]. Starting with the work of Boneh and Franklin on identity-based encryption (http://en.wikipedia.org/wiki/ID-based encryption) (IBE), there was a slew of new encryption schemes achieving various properties (e.g., attribute-based encryption, hidden vector encryption, predicate encryption). Many of these constructions felt loosely related so the idea behind FE was to capture all these schemes under a single framework. Though everything we'll cover can be done with FE, for concreteness, we'll consider the special case of IBE, which was first suggested by Shamir [Shamir84] (http://discovery.csc.ncsu.edu/Courses/csc774-S07/shamir84.pdf) and realized by Boneh and Franklin [BF01 (http://crypto.stanford.edu/~dabo/pubs/papers/bfibe.pdf)].

A public-key IBE scheme consists of four algorithms:

- A setup algorithm Setup used to generate a master secret and public key pair (msk, mpk).
- An encryption algorithm Enc that takes as input the master public-key mpk, an identity id and a message m as input and returns a ciphertext c.
- A key generation algorithm Keygen that takes as input the master secret key msk and an identity id and returns a secret key sk_{id}.
- And finally a decryption algorithm Dec that takes as input a secret key sk_{id} and a ciphertext c and returns a message m or a failure symbol $\perp$.

The motivation behind IBE is key distribution. In particular, using an IBE scheme should be easier than using a standard (public-key) encryption scheme where public keys have to be certified, revoked and verified.

Let's consider a concrete example. Suppose Alice wants to send an encrypted message to Bob who works at Microsoft. The idea is that Microsoft would first generate a pair of master keys (msk, mpk) and distribute mpk together with a certificate. To send her message m, Alice would retrieve Microsoft's master public key mpk, verify its certificate and then encrypt m under Bob's identity by computing:

$$c = \mathrm{Enc}(mpk, \text{``}\texttt{bob@microsoft.com}\text{''}, m).$$

To decrypt the ciphertext c, Bob needs to hold a secret key for his identity under Microsoft's master key:

$$sk = \mathrm{Keygen}(msk, \text{``}\texttt{bob@microsoft.com}\text{''}).$$

Given this key, he can then recover the message by computing $m = \mathrm{Dec}(sk, c)$.

Notice that Alice never needed to know what Bob's public key was or to verify any certificate for his key. The only certificate she had to verify was for Microsoft's master public key but once that key is authenticated she can send email to anyone at Microsoft without any additional work.

Public-Key Encrypted Search

We are now ready to see how (anonymous) IBE can be used to search over encrypted data. This idea was first proposed by Boneh, Di Crescenzo, Ostrovsky and Persiano [BCOP04 (http://crypto.stanford.edu/~dabo/pubs/papers/encsearch.pdf)] and is best explained by considering the following email scenario where Alice wants to send an encrypted email to Bob.

Bob first generates a master secret and public key pair for the IBE scheme (msk, mpk) and a secret and public key pair for a standard public-key encryption scheme (sk, pk) . He then makes the public keys (mpk, pk) public and keeps the secret keys (msk, sk) private. Alice encrypts her message under pk using the standard public-key encryption scheme, resulting in a ciphertext c . She then attaches IBE encryptions of "1" under Bob's master public key mpk with the keywords as the identity. This results in a set of IBE encryptions $(e_1, \ldots, e_m)$ where each e_j (for $1 \leq j \leq m$) is defined as

$$e_j = \mathrm{Enc}(mpk, w_j, 1),$$

where $(w_1, \ldots, w_m)$ are the keywords.

Let's suppose Bob's email server has received n emails of this form, so that it now holds a set of encrypted emails $(c_1, \ldots, c_n)$ and an encrypted database

$$\mathrm{EDB} = \Big(\big(e_{1,1}, \ldots, e_{1,m}, \mathrm{ptr}(c_1)\big), \ldots, \big(e_{n,1}, \ldots, e_{n,m}, \mathrm{ptr}(c_n)\big) \Big).$$

Now, if Bob wants to retrieve the emails with keyword w, he just needs to generate a secret IBE key as $sk_w = \mathrm{Keygen}(msk, w)$ and send it as the token to the server. The server then tries to decrypt each IBE ciphertext in EDB and if successful follows the associated pointer to return the appropriate encrypted email.

An important observation is that a standard IBE scheme here will not be enough. The problem is that the notion of IBE does not necessarily guarantee that a ciphertext hides information about the identity used to create it. This means that if we were to use a standard IBE scheme, EDB could leak the keywords to the server. To address this, Boneh et al. observe that what you actually need is an *anonymous* IBE scheme which essentially means that the ciphertexts do not reveal information about the identities. Fortunately, we know how to construct such schemes efficiently so this is not a major concern from a practical point of view (e.g., the Boneh-Franklin IBE scheme is anonymous).

Search time for the server is $O(nm)$ since it has to try to decrypt each ciphertext in the EDB. Assuming $m \ll n$, this is $O(n)$ which is a *a lot* slower than the solution based on deterministic encryption described in the previous post (http://outsourcedbits.org/2013/10/14/how-to-search-on-encrypted-data-part-2/) which required time $o(n)$ (i.e., sub-linear in n).

Is this Secure?

While this approach is slower than the PPE-based approach, it has better security properties. First, the encrypted database by itself does not reveal much useful information to the server since—unlike the deterministic approach—keywords are encrypted using a *randomized* (identity-based) encryption scheme. So even if two documents have keywords in common, the encrypted keywords in EDB will be different. This means that we don't have to make unnatural assumptions about the data (e.g., that it has high entropy) to use it safely.

There is an issue, however, with this approach: *it does not protect the search terms*. In particular, the server could mount the following attack to figure out which keyword the client is searching for.

Suppose the server has some dictionary W of d words. For each keyword $w \in W$ it encrypts "1" with key mpk and identity w. This results in a set of d (identity-based) encryptions $(e'_1, \ldots, e'_d)$. Now, given some token sk_w, the server can learn w by simply trying to decrypt each of the ciphertext e'_i with sk_w. If the decryption works for some e'_i, then the server knows that sk_w is for the identity used to generate e'_i.

Notice that the attack does not result from a deficiency of any particular IBE scheme but that it applies to *any* public-key encrypted search solution. The fundamental problem is that the server has both the ability to create EDBs (since it has the public-key) and to search over them. So what this tells us is that, as defined, the notion of search on publicly-encrypted data cannot protect search terms.

So what can we do about this? Recently, Boneh, Raghunathan and Segev [BRS13 (http://eprint.iacr.org/2013/283.pdf)] and Arriaga and Tang [AT13 (http://eprint.iacr.org/2013/330.pdf)] set out to design public-key encrypted search solutions that achieved the best possible level of confidentiality for search terms. Roughly speaking, what this means is that if the search terms are hard enough to guess, then the schemes proposed will protect them.

But what do we do if (as in most cases) our search terms are not hard to guess? Well, we don't really have a good answer except that this problem does not occur in the symmetric setting since only the client can generate EDBs so, depending on the application, a symmetric solution might be preferable.

Conclusions

So far, we' ve seen two approaches to searching on encrypted data. The first, the PPE-based approach (http://outsourcedbits.org/2013/10/14/how-to-search-on-encrypted-data-part-2/), resulted in schemes with fast search (sub-linear in n) but with relatively weak security guarantees. The second, the FE-based approach, resulted in schemes with slow search (linear in n) but with better security guarantees.

In the next post, we'll go over solutions that are even slower, but that achieve the strongest possible levels of security!

13 thoughts on "How to Search on Encrypted Data: Functional Encryption (Part 3)"

1. Pingback: How to Search on Encrypted Data: Intro (Part 1) | Outsourced Bits
2. **Megha says:**
   November 29, 2013 at 10:09 pm
   Waiting for next part…

   Reply
3. Pingback: How to Search on Encrypted Data (Part 4): Oblivious RAMs | Outsourced Bits
4. Pingback: How to Search on Encrypted Data: Deterministic Encryption (Part 2) | Outsourced Bits
5. **Wi says:**
   May 5, 2014 at 1:52 pm
   In the email scenario under section "Public-Key Encrypted Search". when Bob wants to retrive an email from the server he sends sk_w to the serve. If the server successfully decrypt an IBE ciphertext, e, in EDB then he knows the keyword: w = Dec(sk_w, e). Would you please explain this point in more details.

Thanks

Reply

senykam says:
May 5, 2014 at 6:25 pm
Oh, it looks like I forgot the 1. Thank you very much for catching this! w_j is the *identity* and the *message* is 1, so decryption would not leak any information.

It's fixed in the text.

Reply

Wi says:
May 5, 2014 at 8:14 pm
Aha … Thank you so much!

6. Pingback: How to Search on Encrypted Data: Searchable Symmetric Encryption (Part 5) | Outsourced Bits

7. **Prasanna says:**
December 10, 2014 at 10:11 am
I want to know that, can we able to encrypt the documents using RSA. and aslo encrypt the index structure based on B Tree using RSA. How we can perform search operation on this RSA encrypted index B Tree. Please answer

Reply

senykam says:
December 10, 2014 at 2:00 pm
I don't know of a searchable encryption scheme that is based on RSA. I'm not sure why you want to build one based on RSA though. Also note that encrypting the documents with RSA woule be very inefficient.

Reply

8. **Ghada Dessouky says:**
October 9, 2015 at 2:21 am
"keywords are encrypted using a randomized (identity-based) encryption scheme. So even if two documents have keywords in common, the encrypted keywords in {\mathrm{EDB}} will be different."

How is that the case? The identity is IBE encyrptions of "1" under Bob's master public key along with the keywords….so if the keyword is the same for different documents, the encrypted keywords should be the same, isn't it so? How can they be different?

Reply

senykam says:
October 9, 2015 at 9:02 am
I'm not sure I follow what you mean by "the identity is IBE encryptions of "1" under …". The "searchable encryption" of the keyword w is an IBE encryption of "1" under the identity "w" using the recipient's (master) public-key. So if the IBE scheme is randomized then the ciphertext will be randomized even if you encrypt the same message (i.e., "1") under the same identity (i.e., "w").

Reply

9. **Rachel Nallathamby says:**
March 15, 2016 at 7:32 pm

how can i use IBE for encryption and decryption and verify the datagenerated as well.For IBE we use a public and private key. For verification (over cloud) a lookup key is used.How is this lookup key generated. Can both concepts be combined.

Reply