Implementing Public Key Cryptography with Key Word Search

Prashanth Ramakrishna * Hao Shu † Dov Salomon ‡ Jonathan Alter §

May 5, 2020

1 Introduction

For Project 2, we decided to choose Option 6. Option 6 requires summarizing a paper, implementing it's constructions, then testing the performance of our implementations as relevant parameters grow. The paper we chose to implement is entitled, "Public Key Cryptography with Key Word Search". This paper attempts to solve the following question:

Alice is an email user who would like to access her email on multiple devices. However, she would like emails to be sorted, via routing, to different devices based on certain keywords they contain. For example, suppose Bob sends Alice an email containing the keyword "urgent". This email should be routed to Alice's pager rather than her desktop. Such a scheme would be trivial to implement, of course, if Bob's email were in plaintext. The gateway, given a set of keywords, could simply search the plaintext and act accordingly. But, how can such a scheme be accomplished if Bob's email is encrypted and, along with it, the keywords contained therein?

In this case, as should be apparent, routing decisions cannot be made by the gateway because keywords are hidden. This paper develops a scheme whereby Alice enables the gateway to test whether a specified set of keywords is contained in encrypted emails being received without learning anything else. Formally, Bob sends Alice an encrypted email of the following form:

$$E_{A_{pub}}(M) \;||\; \texttt{PEKS}(A_{pub}, W_1) \;||\; \ldots \;||\; \texttt{PEKS}(A_{pub}, W_m) \tag{1}$$

where A_{pub} is Alice's public key, $E_{A_{pub}}$ refers to some standard public key encryption using Alice's public key, PEKS is a *Public-Key Encryption with Keyword Search* of each keyword $W_1, \ldots, W_m$ sent by Bob in message M. The advantage of this form is that Alice can provide a trapdoor T_W which will allow the gateway to test whether any of the keywords associated with Bob's message match Alice's chosen keyword W. That is, given $\texttt{PEKS}(A_{pub}, W')$ and T_W, the gateway has the ability to determine if $W = W'$. Note that if $W \neq W'$, then the gateway learns nothing further about the email other than that $W \notin M$.

Definition 1.1. Non-Interactive PEKS [FILL IN DESCRIPTION OF NONINTERACTION PEKS]

1. $KeyGen(s)$: $KeyGen$ takes the key length security parameter s as input and return a public-private key pair, denoted $\{A_{pub}, A_{priv}\}$
2. $\texttt{PEKS}(A_{pub}, W)$: Given public key A_{pub} and word W, PEKS returns the searchable encryption of W.
3. $T(A_{priv}, W)$: Alice uses her private key A_{priv} to create a trapdoor T_W for chosen word W.
4. $Test(A_{pub}, S, T_W)$: Given Alice's public key, searchable encryption of W', $S = \texttt{PEKS}(A_{pub}, W')$, and trapdoor $T_W = T(A_{priv}, W)$, $Test$ returns boolean $W = W'$.

*pmr347@nyu.edu
†hs3812@nyu.edu
‡dms833@nyu.edu
§ja3943@nyu.edu

First, Alice uses $KeyGen$ to produce her public/private key pair. Then, she uses T to produce a trapdoor T_{W_i} for each $W_i \in W^*$, where W^* is the set of chosen keywords $\{W_1, \ldots, W_m\}$. This set of trapdoors is denoted T_{W^*} and is given by Alice to her mail server. Alice When Alice's mail server receives a message from Bob

$$E_{A_{pub}}(M) \;||\; S_1 \;||\; ... \;||\; S_m$$

it uses $Test$ to determine whether message contains any member of W^*.

Security for a `PEKS` scheme is defined against active attacker $\mathcal{A}$ who, given parameter s, is able to query an oracle for the trapdoor T_W of any chosen word $W \in \{0,1\}^*$. Despite this ability, $\mathcal{A}$ should be unable to distinguish $\texttt{PEKS}(A_{pub}, W_1)$ and $\texttt{PEKS}(A_{pub}, W_2$ if he has T_{W_0} but not T_{W_1}. The challenge for the attacker is, when given two new keywords W_0, W_1 for which he doesn't have the trapdoor, and searchable encryption $\texttt{PEKS}(A_{pub}, W_b$, where $b \in \{0,1\}$, to guess b. $\mathcal{A}$ wins the security game if his advantage,

$$Adv_{\mathcal{A}}(s) = |P(b = b') - \frac{1}{2}| \tag{2}$$

where b' is $\mathcal{A}$'s guess, is non-negligible.

Remark 1.1. The general construction given in Def 1.1 is not secure against a chosen cipher text attack, since $\mathcal{A}$ can simply decrypt after reorder the searchable keyword encryptions appended to M. However, Def 1.1 does ensure semantic security for the message form given in Eq 1 so long a $E_{A_{pub}}$ is itself semantically secure. With modifications, chosen cipher-text security can be achieved. Indeed, in the paper, a chosen ciphertext secure IBE system, IND-ID-CCA, is detailed.

The paper gives two different constructions of the general `PEKS` scheme. The first construction is based on a bilinear function and the second, simpler albeit less efficient one, is based only on general trapdoor permutations. They are detailed in Section 2 and Section 3. In this project, both constructions are implemented

2 Bilinear Construction

This construction makes depends on the hardness of a cousin to the Computational Diffie-Hellman problem known as the **Bilinear Diffie Hellman Problem**.

Definition 2.1. *Bilinear Diffie Hellman Problem (BDH)* Let G_1, G_2 be two groups of fixed prime order p with generator sets g_1 and g_2, respectively. Further, let $e : G_1 \times G_1 \to G_2$ be a bilinear function. That is, e satisfies the following properties:

1. Computability: $\forall x_1, x_2 \in G_1$, $\exists$ polytime algorithm A to compute $e(x_1, x_2) \in G_2$
2. Bilinearity: $\forall x_1, x_2 \in [1, p]$ and $\forall g \in G_1$, $e(g^{x_1}, g^{x_2}) = e(g, g)^{x_1 x_2}$.
3. Non-degeneracy: $\forall g \in g_1$, $e(g, g) \in g_2$.

Now, fix $g \in g_1$. Given $g, g^a, g^b, g^c \in G_1$, compute $e(g, g)^{abc} \in G_2$. If all polytime algorithms have a negligable advantage in solving BDH, then we say BDH is intractable.

The `PEKS` bilinear map construction, in accordance with Def requires G_1, G_2, and e. In addition, however, it requires two hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_2 \to \{0,1\}^{log(p)}$. The scheme, then consists of the four functions defined in Def 1.1.

1. $KeyGen$: Security parameter p is the prime order of G_1 and G_2. $KeyGen$ randomly chooses $\alpha \in \mathbb{Z}_p^*$ and $g \in g_1$. $A_{pub} = [g, h = g^\alpha]$ and $A_{priv} = \alpha$. Finally, $\{A_{pub}, A_{priv}\}$ is returned.
2. $\texttt{PEKS}(A_{pub}, W$: First, $r \in \mathbb{Z}_p^*$ is randomly chosen. Then, $t = e(H_1(W), h^r) \in G_2$ is computed. Finally, $[g^r, H_2(t)]$ is returned.

3. $T(A_{priv}, W)$: given chosen keyword W, T returned trapdoor $T_W = H_1(W)^\alpha \in G_1$.

4. $Test(A_{pub}, S, T_W)$: Let $S = [A, B]$. $Test$ returns the boolean $H_2(e(T_W, A)) = B$.

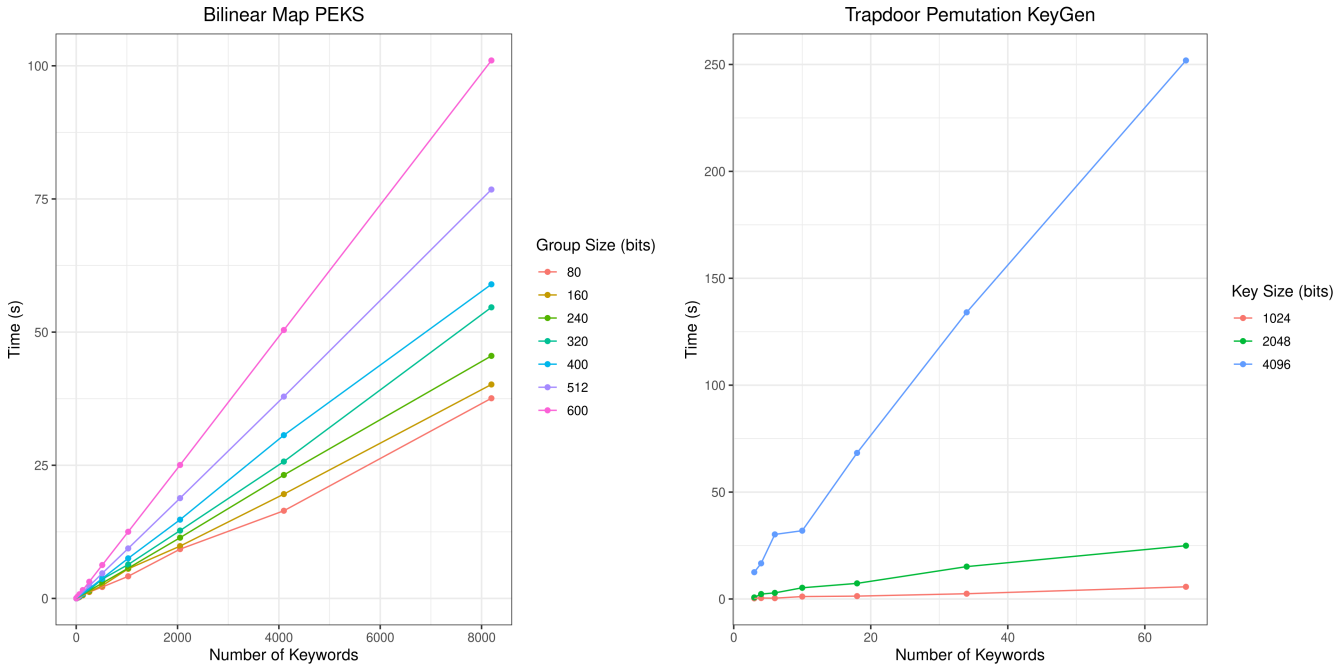
The mechanics of the encryption scheme – how and when these functions are used – are inherited from the generic construction given in Def 1.1. About this scheme, we have our first theorem.

Theorem 1. The non-interactive searchable encryption scheme (`PEKS`) with the bilinear map construction above is semantically secure against a chosen keyword attack in the random oracle model assuming BDH is intractable.

The proof of Thm 1 is too extensive to include here. The proof's high level approach, however is not. We have attack algorithm $\mathcal{A}$ with advantage ϵ against `PEKS`. $\mathcal{A}$ makes $\leq q_{H_2}$ queries to H_2 and $\leq q_T$ trapdoor queries. Further, we have algorithm $\mathcal{B}$ that solves BDH with probability $\epsilon' = \frac{\epsilon}{eq_T q_{H_2}}$, where e is the base of the natural log. Because, by construction, $runtime(\mathcal{A}) \approx runtime(\mathcal{B})$, the BDH assumption holding for G_1, ϵ' will be negligible, and thus, so will ϵ. The proof, therefore, requires the construction of $\mathcal{B}$ such that ϵ' is negligible.

3 Trapdoor Construction

4 Testing



5 Conclusion