Simple Blind Search on Public-Key Encrypted Data

May 09, 2007

D. Nali
School of Computer Science,
Carleton University, Canada
deholo@ccsl.carleton.ca

P.C. van Oorschot
School of Computer Science,
Carleton University, Canada
paulv@scs.carleton.ca

ABSTRACT

We consider the problem of blind search on encrypted data, wherein servers are to perform a search service on the data without being able to discover the associated underlying search criteria. We focus on the case in which search is performed on an index of blinded (i.e. confidentiality-protected) keywords appended to the encrypted data or message. In recent years, several attempts have been made to provide such functionality using public-key encrypted keyword search (PEKS) schemes. We show that all PEKS schemes we are aware of fall to a generic dictionary attack which compromises the privacy of search keywords. This suggests revisiting the existing PEKS model. Given this motivation, we distinguish blind search on *incoming* encrypted data (largely the focus of previous work) from blind search on *outgoing* encrypted data (which addresses novel applications), and provide both a new model for blind keyword search, and a specific proposal fitting the model. Our proposal is both (time- and space-) efficient, and prevents dictionary attacks by search servers. The simple idea is to build search tokens using message authentication codes (MACs) keyed by user-specific strings (in essence, confounders). The keys are shared between the user and the party creating the search tokens, the latter being distinct from the search server. We also distinguish between *systematic blind search* (search on each incoming or outgoing message), and *request-based blind search* (search prompted by sporadic requests, as best resembles the only scheme we are aware of that provides blind search on public-key encrypted data, and does not fall to the aforementioned attack). We show how our specific proposal can be used to address both systematic and request-based blind search efficiently.

1. INTRODUCTION

We use *blind search on encrypted data* (BSED) to refer to the service whereby parties called search servers perform search on encrypted data with the intention that they are unable to discover the associated search criteria. More narrowly, we actually use this term to mean "search on a list or *index* of blinded (i.e. confidentiality-protected, for example through encryption) keywords appended to an encrypted message". Here, a keyword is a relatively short string (e.g. single word appearing in a message). BSED can be viewed as the service whereby search tokens (i.e. system-defined keywords which have been blinded) are used to detect the occurrence of target blinded message keywords (i.e. blinded keywords appended by message originators to encrypted messages).

We distinguish blind search on *incoming* encrypted data from blind search on *outgoing* encrypted data. An example of the former case is that in which an incoming encrypted email arrives at a recipient's email gateway. An example of the latter case is that in which an outgoing encrypted email arrives at a corporate email gateway, and is checked before being sent to intended recipients. We also distinguish *systematic* blind search (i.e. blind search on each encrypted message received or sent by a party), from *request-based* blind search (i.e. blind search prompted by sporadic requests sent to search servers). Previous work on blind search on public-key encrypted data has focused either on incoming or request-based blind search [1, 2, 3, 8, 9, 17, 20]. We seek a solution suitable for both incoming and outgoing systematic blind search on public-key encrypted data (SBS-PED).

SBS-PED has numerous applications including privacy preserving search on encrypted email, web content, and transaction records. Such search can be performed for purposes such as routing, prioritizing, filtering, book-keeping, and flagging for further processing. SBS-PED appears to be particularly suitable for applications in which the data to be searched is encrypted by members of distributed communities (e.g. company employees), as opposed to applications in which users encrypt data, send it to a remote server, and later retrieve it via sporadic queries. Outgoing SBS-PED may be of interest to organizations that prefer (e.g. for improved or legislated user privacy) not to record all plaintext data processed by user devices under their jurisdiction, but want to inspect information leaving their virtual network boundaries. This includes medical organizations regulated by HIPAA [21], and financial institutions regulated by GLBA [26]. Incoming SBS-PED may be of interest to companies that wish to provide privacy-preserving email services. Users could request the following functionality from such service providers: systematic routing of incoming encrypted email to chosen addresses; systematic filtering of

incoming or outgoing encrypted email (e.g. using a white list of obfuscated senders' addresses); and systematic flagging of incoming or outgoing encrypted email for further processing. Similarly, incoming SBS-PED could be of interest to companies that want to provide services whereby encrypted transaction records are inspected and flagged if they contain blinded attributes specified by intended recipients.

We further explain our motivation as follows. At Crypto'01, Boneh et al. [6] proposed a scheme to perform keyword search on public-key encrypted data (i.e. *public-key encrypted keyword search*, or PEKS). Since then, many PEKS schemes have been proposed [1, 2, 3, 9, 17, 20]. In Section 2, we show that these schemes do not provide blind search; the servers can learn the search criteria. In a recent manuscript, Boneh et al. [8] proposed a computational private information retrieval (cPIR) scheme enabling blind search on public-key encrypted data. This scheme is designed for request-based retrieval of public-key encrypted data stored on a remote server; our focus is on systematic blind search. We also seek a simple SBS-PED scheme less likely to include unexpected limitations than known PEKS schemes, and less likely prone to implementation errors by programmers than more complex schemes (e.g. [8]).

To this end, our solution employs message authentication codes (MACs) keyed by user-specific strings or keys.[1] The simple idea is to blind message keywords by computing MACs on keywords, rather than public-key encrypting keywords. Each MAC'ed keyword is computed by a user software client, with a user-specific key obtained from a warden via a one-time authenticated user registration procedure. The warden generates search tokens by MAC'ing chosen keywords whose occurrence must be detected. These search tokens are given to a search server which compares blinded message keywords (appended to public-key encrypted messages) with search tokens given by the warden. If there is a match, the associated encrypted message is processed according to a predetermined policy. Since a keyword cannot be recovered from a MAC, and search servers have no access to user keys, the proposed method provides enables blinding and detection of chosen keywords.

We see our main contributions as threefold. (1) We present a generic attack against all PEKS schemes we are aware of [1, 2, 3, 9, 17, 20]. (2) We distinguish incoming vs. outgoing blind search, request-based vs. systematic blind search, and present a new model for systematic blind search on public-key encrypted data. (3) We propose a simple and efficient scheme, `KwdMAC`, which enables systematic blind search through use of MAC'ed index keywords, and analyze this solution. We also propose an extension of `KwdMAC`, `KwdMAC`*, enabling request-based (incoming and outgoing) blind search on public-key encrypted data. Table 1 compares the intended uses of `KwdMAC` and `KwdMAC`* with the intended use of the only alternative we are aware of that does not fall to our dictionary attack.

[1]Where we use MAC, we have in mind common functions such as HMAC. We also use the term MAC both as a noun and verb.

	Incoming	**Outgoing**
Request-Based	[8], `KwdMAC`*	`KwdMAC`*
Systematic	`KwdMAC`	`KwdMAC`

Table 1: Blind Search on Public-Key Encrypted Data – Comparison of Intended Use of Proposed and Known Schemes. `KwdMAC`* is the extension of `KwdMAC` described in Section 6. Known PEKS schemes analyzed in Section 2 do not provide blind search [1,2,3,6,9,17,20].

The remainder of this paper is organized as follows. Section 2 describes our generic attack against PEKS schemes. Section 3 presents our model of the SBS-PED problem, including parties involved, and functional, efficiency and security requirements. Section 4 describes details of our proposed scheme, which Section 5 analyzes. Section 6 presents an extension of `KwdMac` enabling request-based (incoming and outgoing) blind search on public-key encrypted data. Section 7 reviews related work. Section 8 concludes. The Appendix details specific dictionary attacks on pairing-based PEKS schemes, showing that search servers can verify guesses of keywords used to generate search tokens. We expect these attacks may be of independent interest.

2. LIMITATION OF PEKS

In this section, we review the definition of public-key encrypted keyword search (PEKS), and explain why it does not (under its current formulation) ensure blind search on encrypted data, by presenting a generic dictionary attack on PEKS schemes which defeats the blinding.

2.1 Review of PEKS

A public-key encrypted keyword search scheme consists of the following polynomial-time algorithms [6]:

`Setup` (k): Given a security parameter k, this algorithm generates a tuple π of system-wide public parameters.

`KeyGen` (π): Given a tuple π of public parameters, this algorithm produces a user-specific public/private key pair (A_{pub}, A_{priv}).

`PEKS` (A_{pub}, w, π): Given a public-key A_{pub}, a keyword w, and a tuple π of public parameters, this algorithm produces a ciphertext C corresponding to w.

`Token` (A_{priv}, w, π):[2] Given a private-key A_{priv}, a keyword w, and a tuple π of public-key parameters, this algorithm produces a search token t_w enabling search on ciphertexts produced by `PEKS`.

`Test` (A_{pub}, C, t_w, π): Given a public-key A_{pub}, a ciphertext $C = \texttt{PEKS}(A_{pub}, w', \pi)$, a search token t_w, and a tuple π of public system parameters, this algorithm returns "yes" if $w = w'$, and "no" otherwise.

This definition and slight variants thereof have been adopted by several PEKS scheme designers [1, 2, 3, 6, 9, 17, 20].

[2]*Search tokens* have also been referred to as *trapdoors* [6]. Hence, the `Token` algorithm has also been called `Trapdoor`.

2.2 Generic Dictionary Attack on PEKS

We present a generic dictionary attack on PEKS schemes, such that in the defined framework of public-key encrypted keyword search, search servers can confirm guesses of keywords associated with given search tokens. Search servers are therefore able to compromise the confidentiality of search tokens given to them, including information about the content of the associated encrypted message.

THEOREM 1. *PEKS schemes as defined in Section 2.1 are susceptible to a generic dictionary attack allowing verification of guesses of keywords associated with any given search token.*

Proof: Let S be a PEKS scheme (per Section 2.1), and $\mathbb{W} = \{w_1, w_2, \cdots, w_n\}$ any dictionary of keywords. Let also $t_w = \texttt{Token}(A_{priv}, w, \pi)$ be the search token associated with an arbitrary keyword w. Given t_w, test whether $w = w_i$ for any $w_i \in \mathbb{W}$, by checking whether

$$\texttt{Test}(A_{pub}, C_i, t_w, \pi) = \text{“yes”} \text{ with } C_i = \texttt{PEKS}(A_{pub}, w_i, \pi).$$

Hence any party with access to t_w (such as a search server) can launch a dictionary attack against any given t_w by performing the aforementioned test with each element of a target dictionary $\mathbb{W}$. $\mathbb{W}$ can be built based on target applications. For example, in the case of a health care email application, $\mathbb{W}$ may include disease names, physicians' names, and patients' identifiers. In the case of an email application used by an organization managing intellectual property, $\mathbb{W}$ may include technical terms describing inventions. In the case of an email or instant message application used by stock brokers, $\mathbb{W}$ may include chosen company names.

Example: As an example of our generic attack, we consider the scheme of Boneh et al. [6].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
3. Generate two cryptographic hash functions $\mathbb{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathbb{H}_2 \colon \mathbb{G}_2 \to \{0,1\}^k$.
4. Output $\pi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, \mathbb{H}_1, \mathbb{H}_2)$.

`KeyGen` (π):

1. Pick $s \in_R \mathbb{Z}_p^*$,[3] and a generator P of $\mathbb{G}_1$.
2. Compute $S = sP$, and set $(A_{pub}, A_{priv}) = ((P, S), s)$.
3. Output (A_{pub}, A_{priv}).

`PEKS` $(A_{pub} = (P, S), w, \pi)$:

1. Pick $r \in_R \mathbb{Z}_p^*$.
2. Compute $U_1 = \mathbb{H}_2(\hat{e}(\mathbb{H}_1(w), S))$ and $U_2 = rP$.
3. Output $C = (U_1, U_2)$.

`Token` (A_{priv}, w, π): Output $t_w = A_{priv}\mathbb{H}_1(w) \in \mathbb{G}_1$.

`Test` $(A_{pub}, C = (U_1, U_2), t_w, \pi)$:

Output “yes” if $\mathbb{H}_2(\hat{e}(U_2, t_w)) = U_1$, and “no” otherwise.

[3]For any finite set X, $x \in_R X$ denotes that x is chosen uniformly at random from X.

To test whether a given search token T_w was generated for a candidate keyword w', an attacker can check whether

$$\mathbb{H}_2(\hat{e}(rP, t_w)) = \mathbb{H}_2(\hat{e}(\mathbb{H}_1(w'), S))$$

for any $r \in_R \mathbb{Z}_p^*$, i.e. whether

$$\texttt{Test}(A_{pub}, C', t_w, \pi) = \text{“yes”}, \text{ with } C' = \texttt{PEKS}(A_{pub}, w', \pi).$$

This dictionary attack applies to a number of prominent PEKS schemes [1, 2, 3, 6, 9, 17, 20]. The fact that it does not seem to have previously appeared in the literature suggests that it is either not previously known, or at best not widely known (particularly in its generic form). In [1, 2, 3, 6, 17], the authors are silent with respect to the support for blind search by their schemes. Kim et al. [20] claim that their scheme supports blind search. (This claim is therefore inaccurate.) Boneh and Waters [9] mention that their PEKS scheme does not support blind search, but do not further discuss the implications or relevance of the statement. In a recent manuscript, Boneh et al. [8] assert that two schemes [1, 6] do not provide blind search, without describing an attack.

Theorem 1 provides a sufficient criterion for mounting a dictionary attack against PEKS schemes, namely the very definition of PEKS schemes introduced by Boneh et al. [6], and adopted by many [1, 2, 3, 6, 9, 17, 20]. This implies that the attack is not only generic, but also applies to all (known and future) PEKS schemes following this definition.

The main issue here is the assumed threat model. Security proofs provided for the vulnerable PEKS schemes are concerned with semantic security with respect to parties having no access to search tokens. Theorem 1 demonstrates why such a threat model is incomplete, in light of the fact that the original motivation for PEKS is the ability to perform blind search using search tokens. This motivates our investigation, in the next section, of a more realistic threat model for systematic blind search on encrypted data.

The attacks presented in the Appendix are specific to individual PEKS schemes, and can be viewed as special cases (vs. direct instantiations) of the generic attack presented in the proof of Theorem 1.

3. MODELING SBS-PED

In this section, we present a model for systematic incoming and outgoing blind search on public-key encrypted data (SBS-PED). (Recall that systematic blind search refers to blind search on each of the encrypted messages received by a search server.) In contrast, previous models for blind search on encrypted data in the public-key setting have focused on applications involving either incoming encrypted data [1, 2, 3, 8, 9, 17, 20] or incoming request-based search [8]. Our model includes functional, efficiency, and security requirements. Section 3.1 identifies parties involved in SBS-PED. Section 3.2 and Section 3.3 specify functional and efficiency requirements desirable for SBS-PED. Section 3.4 presents the threat model we consider for SBS-PED. Section 3.5 identifies the security requirements driving our proposal allowing systematic blind search on public-key encrypted data.

3.1 Principals

Our model for SBS-PED involves four main parties – $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{S}$ – with the following roles and objectives.

- $\mathcal{A}$ wants to send a confidential message m to B.
- $\mathcal{B}$ wants messages intended for her to be unintelligible to all parties except these messages' senders.
- $\mathcal{C}$ (the *warden*) desires automatic blind search be performed. $\mathcal{C}$ is either $\mathcal{B}$, or a party which bears liability for $\mathcal{A}$'s actions and/or to whom $\mathcal{A}$ is accountable (e.g. $\mathcal{A}$'s employer). In the latter case, $\mathcal{A}$ wants blind search to be performed on outgoing data sent by $\mathcal{A}$.
- $\mathcal{S}$ provides a systematic blind search service to $\mathcal{C}$ as follows. $\mathcal{S}$ wants to perform search on a class of (incoming or outgoing) encrypted messages it receives (e.g. those sent by $\mathcal{A}$ or addressed to specified recipients), using search tokens provided by $\mathcal{C}$, and handles encrypted messages according to a predefined policy (e.g. notify $\mathcal{C}$ each time a received message sent by $\mathcal{A}$ meets a search criterion encoded in a search token given by $\mathcal{C}$).

We model $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{S}$ as probabilistic polynomial-time finite automata, to express our assumption that these parties have access to polynomially-bounded computational and storage resources. We also use the following terminology. A *message keyword* is a word (keyword) contained in or associated with a message generated or selected by $\mathcal{A}$; message keywords are assumed to be automatically associated with their corresponding messages (e.g. by client software). A *blinded message keyword* is a message keyword made unintelligible to all parties except $\mathcal{A}$ and $\mathcal{C}$. A *search keyword* is a system-defined target word (keyword) whose occurrence in messages is to be detected by $\mathcal{S}$. A *search token* is a search keyword made unintelligible to all except $\mathcal{C}$ and $\mathcal{A}$. The idea is that search tokens are compared with blinded message keywords in some manner to allow detection.

3.2 Functional Requirements

An SBS-PED scheme consists of the following polynomial-time algorithms:

`Setup` (π_s, π_f):
Given a tuple π_s of security parameters, and a tuple π_f of functional parameters, this algorithm outputs a tuple π of system public parameters. `Setup` should be run by $\mathcal{C}$. (See Section 4 for a concrete example.)

`User-Reg` $(\mathcal{A}, P_{auth}, \pi)$:
Given a party $\mathcal{A}$ claiming an identity $ID_{\mathcal{A}}$, and an authentication policy P_{auth}, the algorithm verifies the authenticity of $ID_{\mathcal{A}}$ with respect to $\mathcal{A}$. If $\mathcal{A}$ is not authenticated, `User-Reg` outputs a short constant string $\perp$ indicating user authentication failure. Otherwise, the algorithm outputs a tuple $r_{\mathcal{A}}$ including π and $k_{\mathcal{A}}$, where π is a tuple of system public parameters specified as an input of `User-Reg`, and $k_{\mathcal{A}}$ is a secret string (key) specific to $\mathcal{A}$. `User-Reg` should be run by $\mathcal{C}$.

`Search-Kwd-Gen` $(\mathcal{A})$:
Given the identity of a party $\mathcal{A}$, this algorithm outputs a dictionary $\mathbb{W}_{\mathcal{A}}$ of search keywords. `Search-Kwd-Gen` should be run by $\mathcal{C}$.

`Search-Token-Gen` $(\mathbb{W}_{\mathcal{A}}, \pi, k_{\mathcal{A}})$:
Given a dictionary $\mathbb{W}_{\mathcal{A}}$ of (potentially user-specific) search keywords, tuple π of system public parameters, and optional user key $k_{\mathcal{A}}$, this algorithm outputs a data structure $T_{\mathbb{W}_{\mathcal{A}}}$ encoding the set of all search tokens associated with the elements of $\mathbb{W}_{\mathcal{A}}$. `Search-Token-Gen` should be run by $\mathcal{C}$, and the SBS-PED scheme must enable membership tests of $\mathbb{W}_{\mathcal{A}}$ while protecting the confidentiality of $\mathbb{W}_{\mathcal{A}}$'s elements.

`Encrypt` $(m, Pub_{\mathcal{B}}, \pi, k_{\mathcal{A}})$:
Given a message m, public key $Pub_{\mathcal{B}}$ of an intended recipient $\mathcal{B}$, tuple π of system public parameters, and optional user key $k_{\mathcal{A}}$, this algorithm outputs a tuple c such that: (1) the private key associated with $Pub_{\mathcal{B}}$ is necesary and sufficient to recover m from c; (2) authorized search servers are able to determine whether m contains specified search keywords; and (3) such servers can forward c to $\mathcal{B}$ (i.e. they must be provided with an identifier of $\mathcal{B}$ enabling them to do so). `Encrypt` should be run by $\mathcal{A}$ (i.e., $\mathcal{A}$'s client software).

`Blind-Search` $(c, T_{\mathbb{W}_{\mathcal{A}}}, P_{sch}, \pi)$:
Given a tuple c output by `Encrypt`, data structure $T_{\mathbb{W}_{\mathcal{A}}}$ output by `Search-Token-Gen`, search policy P_{sch}, and system public parameters π, this algorithm identifies the keywords associated with c that belong to $\mathbb{W}_{\mathcal{A}}$, and, for each of these keywords, follows a procedure specified by P_{sch}. (For an example of P_{sch}, see Section 4.) `Blind-Search` should be run by $\mathcal{S}$.

3.3 Efficiency Requirements

We set the following efficiency requirements for the SBS-PED model.

E1. *Off-line Blinded Message Keyword Generation.* Each blinded message keyword can be independently generated (i.e. without interaction with another party) by any authorized message sender.

E2. *Efficient Blinded Message Keyword Generation.* (1) The computational and space costs due to the generation of each blinded message keyword is constant and short (e.g. similar to the cost of computing a cryptographic hash with a relatively short input string). (2) For each message, the computational and space costs due to the generation of all blinded message keywords is at most linear in the message length.

E3. *Efficient Keyword Search.* The computational and space costs due to each incoming SBS-PED message inspection is at most linear in the number of search tokens. For outgoing SBS-PED, the computation and space costs due to the inspection of each message is at most linear in the number of message keywords.

3.4 Threat Model

The SBS-PED model addresses the following threats.

T1. *Attack on Message Confidentiality.* $\mathcal{S}$ might attempt to recover the plaintext content corresponding to encrypted data it processes.

T2. *Attack on Search Keyword Confidentiality.* $\mathcal{S}$ might attempt to discover which keywords correspond to the search tokens given to $\mathcal{S}$ by $\mathcal{C}$. (For example, $\mathcal{S}$ might attempt a dictionary attack against these search tokens, using a dictionary of potential search keywords.)

We do not attempt to address the following two threats.

T3. *Search Pattern Confidentiality* [8]. $\mathcal{S}$ might attempt to infer the blind search patterns associated with $\mathcal{C}$'s search tokens, i.e. the time of occurrence, search token, blinded message keyword, and blind search result associated with each blind search performed by $\mathcal{S}$ with any search token given to $\mathcal{S}$ by $\mathcal{C}$.

T4. *Blinded Message Keyword Validity.* $\mathcal{A}$ might attempt to either delete, modify, or insert message keywords used to generate blinded message keywords.

Addressing T3 might not be necessary in a number of practical application scenarios (e.g. search on encrypted email). T4 might be mitigated through the use of trusted software generating keywords from messages.

3.5 Security Requirements

Let m be a message $\mathcal{A}$ wants to send to $\mathcal{B}$, w a keyword contained in m, $\hat{w}$ a blinded message keyword associated with w, and t_w the search token generated by $\mathcal{C}$ for the same keyword w. We set the following security requirements for the SBS-PED model.

S1. *Completeness.* Given m and w, $\mathcal{A}$ can generate $\hat{w}$.

S2. *Soundness.* Given $\hat{w}$ and t_w, $\mathcal{S}$ can always determine whether $\hat{w}$ corresponds to a keyword whose occurrence $\mathcal{S}$ is supposed to detect.

S3. *Message Confidentiality and Integrity.* The confidentiality and integrity of m is preserved between $\mathcal{A}$ and $\mathcal{B}$.

S4. *Blinded Message Keyword Confidentiality and Integrity.* The confidentiality and integrity of $\hat{w}$ is preserved between $\mathcal{A}$ and $\mathcal{S}$.

S5. *Search Token Blindness.* The probability that $\mathcal{S}$ discovers w, given only $\hat{w}$, t_w, and publicly available information, is negligible (with respect to π_s).

4. PROPOSED SCHEME: KWDMAC

We now describe `KwdMAC`, our specific proposal for blind search wherein (only) the MACs of index keywords are appended to public-key encrypted data. We use the notation and model of Section 3, and Bloom filters.

Bloom filters [5] provide a space- and time-efficient method to test set membership. Each Bloom filter B_Y associated with a t-element set Y is parameterized with two positive integers t_1 and t_2, where $h_1, \ldots, h_{t_1}$ are independent hash functions such that $h_i : \{0,1\}^* \rightarrow \{0,1\}^{t_2}$ for each $i = 1, \cdots, t_1$. Without loss of generality, assume that each element of Y is bit string. B_Y can be implemented with a t_2-bit array initially "zeroed out". For each $s \in Y$ and $i \in \{0, \cdots, t_1\}$, $h_i(s)$ is set to 1. To test whether any given bit string s' belongs to Y, one checks whether $h_i(s') = 1$ for each $i = 1, \cdots, t_1$. If so, s' is declared to be in S. Otherwise, s' is declared not to be in Y. The probability that s' is declared to be in Y when this is inaccurate is approximately $.6185^{t_2/t}$; testing set membership requires at most t_1 hash computations. We now present `KwdMAC` (see Fig. 1).

`Setup` $(\pi_s, \pi_f = (ID_{\mathcal{S}}, Pub_{\mathcal{S}}))$:
Let π_s be a security parameter, $ID_{\mathcal{S}}$ be a permanent global identifier of $\mathcal{S}$, and $Pub_{\mathcal{S}}$ an encryption public-key of $\mathcal{S}$. $\mathcal{C}$ obtains $ID_{\mathcal{S}}$ and an authentic copy of $Pub_{\mathcal{S}}$. Then, $\mathcal{C}$ chooses n_1 and n_2 to be integer values polynomial in π_s and sufficiently large to prevent correct guessing of elements chosen uniformly at random from $\{0,1\}^{n_1}$ and $\{0,1\}^{n_2}$ respectively (e.g. set $n_1 = n_2 = 80$). $\mathcal{C}$ chooses another integer n_3 such that $n_3 \geq \max(n_1, n_2)$. $\mathcal{C}$ selects a MAC function $H\colon \{0,1\}^* \times \{0,1\}^{n_2} \rightarrow \{0,1\}^{n_3}$ (e.g. HMAC-SHA1, in which case $n_3 = 160$). For any $(w,k) \in \{0,1\}^* \times \{0,1\}^{n_2}$, we denote $H(w,k)$ by $H_k(w)$. $\mathcal{C}$ chooses an integer n_4 to denote the bit length of the maximal number of system users (e.g. $n_4 = 30$ for 10^9 users). $\mathcal{C}$ also generates a public-private key pair $(Pub_{\mathcal{C}}, Priv_{\mathcal{C}})$. Let d_H be a description of H. $\mathcal{C}$ outputs system parameters $\pi = (n_1, n_4, d_H, ID_{\mathcal{S}}, Pub_{\mathcal{S}}, Pub_{\mathcal{C}})$.

`User-Reg` $(\mathcal{A}, P_{auth}, \pi)$:
$\mathcal{C}$ authenticates $\mathcal{A}$ using a predefined policy P_{auth} (e.g., specifying out-of-band means). If $\mathcal{A} = \mathcal{S}$, $\mathcal{C}$ rejects $\mathcal{S}$'s attempt to become a registered user.[4] If $\mathcal{A} \neq \mathcal{S}$, $\mathcal{C}$ selects a user secret key $k_{\mathcal{A}}$ uniformly at random from $\{0,1\}^{n_1}$, securely shares this secret with $\mathcal{A}$, and publicly or secretly communicates π to $\mathcal{A}$.

`Search-Kwd-Gen` $(\mathcal{A})$:
Given the identity of $\mathcal{A}$, $\mathcal{C}$ generates a dictionary $\mathbb{W}_{\mathcal{A}}$ of search keywords, possibly specific to $\mathcal{A}$. For example, $\mathbb{W}_{\mathcal{A}}$ might include stock symbols which $\mathcal{A}$ is forbidden to communicate to external parties, or the names of drugs for which critical clinical trial results should not be communicated.

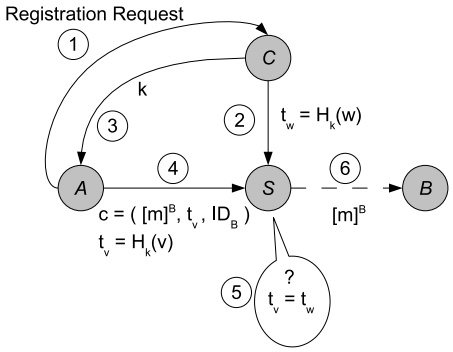
`Search-Token-Gen` $(\mathbb{W}_{\mathcal{A}}, \pi, k_{\mathcal{A}})$:
$\mathcal{C}$ computes $t_w = t_{(k_{\mathcal{A}}, w)} = H_{k_{\mathcal{A}}}(w)$ for each $w \in \mathbb{W}_{\mathcal{A}}$. Then $\mathcal{C}$ forms a data structure $T_{\mathbb{W}_{\mathcal{A}}}$ encoding the set of all search keywords $t_{(k_{\mathcal{A}}, w)}$, where $w \in \mathbb{W}_{\mathcal{A}}$. To this end, we propose to use a Bloom filter to encode $T_{\mathbb{W}_{\mathcal{A}}}$. $\mathcal{C}$ gives $T_{\mathbb{W}_{\mathcal{A}}}$ to $\mathcal{S}$; this is done secretly, if $\mathcal{C}$ wants $T_{\mathbb{W}_{\mathcal{A}}}$ to be a secret shared between $\mathcal{C}$ and $\mathcal{S}$. `Search-Token-Gen` is not executed on a per-message basis, but rather at registration time, and perhaps periodically thereafter.

`Encrypt` $(m, Pub_{\mathcal{B}}, \pi, k_{\mathcal{A}})$:
After selecting or generating a message m, $\mathcal{A}$ (i.e., $\mathcal{A}$'s client software) automatically generates a set $\mathbb{V}$ of message keywords contained in m. This may be done according to some policy set by $\mathcal{C}$, or in the extreme by taking all words of m excluding a black-list of the most common everyday words. Let $[m]^{\mathcal{B}}$ denote the asymmetric encryption of m under $\mathcal{B}$'s encryption public key, using any public-key cipher semantically secure against adaptive chosen-ciphertext attacks (e.g. RSA-OAEP).[5] Let $ID_{\mathcal{B}}$ be a global identifier of $\mathcal{B}$. $\mathcal{A}$ computes $t_{(k_{\mathcal{A}}, v)} = H_{k_{\mathcal{A}}}(v)$ for each $v \in \mathbb{V}$, and sends $c = ([m]^B, (t_{(k_{\mathcal{A}}, v)})_{v \in \mathbb{V}}, ID_{\mathcal{B}})$ to $\mathcal{S}$, via an integrity- and confidentiality- protected communication channel.

`Blind-Search` $(c, T_{\mathbb{W}_{\mathcal{A}}}, P_{sch}, \pi)$:
Upon receiving c, $\mathcal{S}$ checks whether $t_{(k_{\mathcal{A}}, v)}$ is in $T_{\mathbb{W}_{\mathcal{A}}}$ for each $v \in \mathbb{V}$, and if so, $\mathcal{S}$ notifies $\mathcal{C}$ via a communication channel providing mutual authentication, integrity and confidentiality protection. This same channel is used by $\mathcal{C}$ to reply to

[4] If $\mathcal{S}$ becomes a registered user, $\mathcal{S}$ can confirm, via a dictionary attack, guesses of search keywords used by $\mathcal{C}$ for messages originating from $\mathcal{S}$. These keywords might be the same as those used for other users.

[5] For simplicity, it is assumed that message m is small enough for direct asymmetric encryption. The case in which it is too long can be handled by standard hybrid asymmetric-symmetric enveloping techniques.



k is a high-entropy secret user key. H_k is MAC keyed with k. w is a keyword generated by $\mathcal{C}$. t_w is a search token. m is a message generated by $\mathcal{A}$. v is a keyword contained in message m. $ID_{\mathcal{B}}$ is a global identifier of $\mathcal{B}$. $[m]^{\mathcal{B}}$ is a public-key encryption of m using $\mathcal{B}$'s encryption public key.

Figure 1: Overview of Proposed Scheme.

$\mathcal{S}$ (if necessary). $\mathcal{S}$ then uses a policy P_{sch} previously (or newly) given by $\mathcal{C}$ to handle c (e.g. blocking, dropping, or delaying its delivery). If policy specifies that $\mathcal{S}$ send $[m]^{\mathcal{B}}$ to $\mathcal{B}$, then $\mathcal{S}$ uses $ID_{\mathcal{B}}$ from c.

`KwdMAC` can be used for blind search on both incoming and outgoing data. For the incoming case, $\mathcal{B} = \mathcal{C}$. Using a MAC with user-specific keys prevents the situation of access to one user's key compromising the confidentiality of other users' blinded keywords. Being deterministic, MACs allow the warden to generate, in advance, search keywords matching targeted blinded message keywords.

5. ANALYSIS

In this section, we analyze the efficiency and security of `KwdMAC`, using the requirements from Section 3. Requirements E1–E3 and S1–S5 are met by `KwdMAC`.

5.1 Efficiency Analysis

E1. *Off-line Blinded Message Keyword Generation.* This requirement is met, since `KwdMAC` is an off-line scheme by design. $\mathcal{A}$ must interact with $\mathcal{C}$ only once, at registration time.

E2. *Efficient Blinded Message Keyword Generation.* The generation of each valid blinded message keyword requires the computation of a single MAC. The computational and space costs due to the generation of all blinded message keywords associated with a given message is linear in the message length (enumerating the words of a message requires space and work resources linear in the message length, and the cost of blinding each message word requires the computation of a single MAC).

E3. *Efficient Keyword Search.* By design, the computational and space costs due to the inspection of each message keyword is constant and short, since $T_{\mathbb{W}_{\mathcal{A}}}$ is encoded as a Bloom filter (at most t_1 hash computation are required when the Bloom filter has t_1 input functions). Hence, the cost of inspection of each message is linear in the number of message keywords (for outgoing SBS-PED). For incoming SBS-PED, the cost of search for an arbitrary query is linear in the number of search keywords.

5.2 Security Analysis

S1. *Completeness.* Given m and w, $\mathcal{A}$ can generate $\hat{w}$ with H and $k_{\mathcal{A}}$ (using the `Encrypt` algorithm). Completeness is therefore achieved by design.

S2. *Soundness.* Since $\hat{w} = t_w$ (by design), and $\mathcal{C}$ provides $\mathcal{S}$ with a data structure $T_{\mathbb{W}_{\mathcal{A}}}$ encoding the set of all search tokens whose occurrence $\mathcal{S}$ is supposed to detect (including, e.g., t_w), $\mathcal{S}$ can always determine whether $\hat{w}$ is associated with a keyword whose occurrence it is supposed to detect.

S3. *Message Confidentiality and Integrity.* By design, m is encrypted by $\mathcal{A}$ for $\mathcal{B}$ with a public-key encryption scheme semantically secure against adaptive chosen ciphertext attacks. Consequently, the confidentiality and integrity of m is preserved between $\mathcal{A}$ and $\mathcal{B}$ (assuming, of course, that $\mathcal{B}$'s decryption key is kept secret by $\mathcal{B}$, and, if $\mathcal{C} \neq \mathcal{B}$, that $\mathcal{C}$ does not have access to m, e.g. for privacy policy compliance).

S4. *Blinded Message Keyword Confidentiality and Integrity.* By design, the `Encrypt` algorithm sends $c = ([m]^{\mathcal{B}}, (t_{(k_{\mathcal{A}},v)})_{v\in\mathbb{V}}, ID_{\mathcal{B}})$ via an integrity- and confidentiality-protected communication channel.

S5. *Search Token Blindness.* We need to provide evidence that the probability that $\mathcal{S}$ discovers w, given $\hat{w}$, t_w, and public available information, is negligible (with respect to any security parameter with polynomially-long bit-size). To do this, we assume that H is a pseudorandom function, and use the following argument.

Proof Sketch: Given publicly available information and a search token $t_w = \hat{w}$ generated with a user key $k_{\mathcal{A}}$, $\mathcal{S}$ has only two options to discover w. First, $\mathcal{S}$ may use a polynomial-time algorithm able to determine a polynomial-size set that includes a pre-image of the t_w. Second, $\mathcal{S}$ may make a correct random guess of both k_A and w. The first option cannot be achieved with non-negligible success probability, since H is assumed to be a pseudorandom function (otherwise one could build a polynomial-time algorithm that distinguishes H from an element chosen uniformly at random from the set of all random functions from $\{0,1\}^{n_2}$ to $\{0,1\}^{n_3}$). The second option succeeds if an only if the attacker makes the right guess, and the probability that this happens is at least 2^{-n_2}. Consequently, $\mathcal{S}$ discovers w with probability at least $p = \mu(\pi_s) + 2^{-n_2}$, where μ is a negligible function such that $\mu(\pi_s)$ represents the probability that the first option succeeds. Since n_2 is a polynomial function of π_s, we conclude that p is negligible in π_s. Hence, p is negligible, and the probability that $\mathcal{S}$ discovers w, given only t_w and public available information, is negligible with respect to π_s.

As a summary, we conclude that `KwdMAC` is an efficient scheme enabling blind keyword search on public-key encrypted data.

6. EXTENSION FOR REQUEST-BASED BLIND SEARCH

In this section, we describe KwdMAC*, an extended version of KwdMac intended to be used for request-based blind search on public-key encrypted data (RBS-PED). We use the notation of Section 4 (employed to describe KwdMac). The main difference between SBS-PED and RBS-PED is that, in the latter case, blind search is prompted by sporadic requests sent to $\mathcal{S}$ by $\mathcal{C}$.[6] This implies the following fundamental functional difference between SBS-PED and RBS-PED: in SBS-PED, $\mathcal{C}$ must provide $\mathcal{S}$ with a data structure allowing set membership tests on an encoded list of search tokens; in RBS-PED, $\mathcal{C}$ must provide a list of search tokens used by $\mathcal{S}$ to retrieve associated ciphertexts which $\mathcal{S}$ must have stored. Given this functional difference, we provide, in this section, a procedure to convert the SBS-PED scheme presented in Section 4 into an RBS-PED scheme called KwdMAC*. We observe that the functional difference between SBS-PED and RBS-PED entails the following computational challenge: testing set membership (as done in SBS-PED for the encoded list of search tokens) can be made efficient through the use of Bloom filters, whereas performing keyword-based search (as in done RBS-PED in a set of ciphertexts appended with keywords) is more challenging. Consequently, the main purpose of this section is to describe a method whereby $\mathcal{S}$ can efficiently store and retrieve ciphertexts based on search tokens sporadically sent by $\mathcal{C}$.

To this end, $\mathcal{S}$ can use two hash tables $T_{\mathcal{S}}$ and $T'_{\mathcal{S}}$ linking hashed blinded message keywords with sets of associated ciphertexts. More specifically, each entry of $T'_{\mathcal{S}}$ has the form $(b, [m_b]^{\mathcal{B}})$ where $b = h'([m_b]^{\mathcal{B}})$, h' is a collision-resistant hash function with appropriate domain and codomain, and $[m_b]^{\mathcal{B}}$ is a public-key encrypted message received by $\mathcal{S}$. Each time a new ciphertext $[m_b]^{\mathcal{B}}$ is received by $\mathcal{S}$, it is added to $T_{\mathcal{S}}$ at the appropriate index, as explained below. Each entry of $T_{\mathcal{S}}$ has the form $(a, \{id_{(a,i)}\}_{i\in I})$, where $a = h(t_w)$, h is a collision-resistant hash function with appropriate domain and codomain, t_w is a blinded message keyword, $id_{(a,i)} = h'([m_{(a,i)}]^{\mathcal{B}})$, and $\{[m_{(a,i)}]^{\mathcal{B}_i}\}_{i\in I}$ is the set of public-key encrypted messages $m_{(a,i)}$ received by $\mathcal{S}$ and whose indexes include t_w. If $\mathcal{S}$ receives a ciphertext tuple $c = ([m]^B, (t_{(k_{\mathcal{A}},v)})_{v\in\mathbb{V}}, ID_{\mathcal{B}})$ and $T_{\mathcal{S}}$ does not have an entry whose first component is $h(t_{(k_{\mathcal{A}},v)})$, then $\mathcal{S}$ adds $(h([m_b]^{\mathcal{B}}), [m_b]^{\mathcal{B}})$ to $T'_{\mathcal{S}}$, and $(a = h(t_{(k_{\mathcal{A}},v)}), \{h'([m]^{\mathcal{B}})\})$ to $T_{\mathcal{S}}$. If, on the other hand, $T_{\mathcal{S}}$ already has an entry of the form $(a = h(t_{(k_{\mathcal{A}},v)}), \{id_{(a,i)}\}_{i\in I})$, then $\mathcal{S}$ adds $(h([m]^{\mathcal{B}}), [m^{\mathcal{B}}])$ to $T'_{\mathcal{S}}$ and $h'(m^{\mathcal{B}})$ to the set $\{id_{(a,i)}\}_{i\in I}$. Given a search token t_w, $\mathcal{S}$ proceeds as follows to retrieve the set of associated ciphertexts. If $T_{\mathcal{S}}$ has an entry of the form $(a = h(t_w), \{id_i\}_{i\in I})$, then $\mathcal{S}$ outputs the set $\{[m_i]^{\mathcal{B}_i}\}_{i\in I}$ where $(h([m_i]^{\mathcal{B}_i}), [m_i]^{\mathcal{B}_i})$ is an entry of $T'_{\mathcal{S}}$ for each $i \in I$. Otherwise, $\mathcal{S}$ outputs the empty set. This procedure can be trivially generalized to retrieve ciphertexts associated with conjunctive and disjunctive combinations of search tokens.

Assuming that encrypted messages received by $\mathcal{S}$ have a maximum constant number of appended blinded keywords, we conclude that storing $T_{\mathcal{S}}$ requires space linear in the number of (unique) blinded message keywords received by $\mathcal{S}$. The time required to determine which messages are associated with a given search token is constant and short (a single hash computation is required). The cost of generating the union or intersection of sets of encrypted messages is asymptotically linear in the number of messages stored in the database (since each entry of $T_{\mathcal{S}}$ may contain all encrypted messages). In practice, however the space requirements of $T_{\mathcal{S}}$ might be smaller. For example, one's emails typically do not all have the same words (and hence potentially the same keywords). We emphasize that $T_{\mathcal{S}}$ requires a small space overhead (namely space required to store pointers $id_{(a,i)}$ and field delimiters), since received ciphertexts and associated blinded message keywords must be stored in order to provide a request-based blind search service.

We now provide a detailed description of KwdMAC*.

Setup*(π_s, π_f):
This algorithm is identical to KwdMac's Setup. It outputs public system parameters π.

User-Reg*$(\mathcal{A}, P_{auth}, \pi)$:
This algorithm is identical to KwdMac's User-Reg. If $\mathcal{A}$ is authenticated, $\mathcal{C}$ secretly gives a high-entropy key $k_{\mathcal{A}}$ to $\mathcal{A}$, and secretly or public gives π to $\mathcal{A}$.

Search-Kwd-Gen*:
This algorithm differs from that of KwdMac. $\mathcal{C}$ generates and outputs a set $\mathbb{W}_{\mathcal{A}}$ of search keywords. These keywords correspond to the message keywords of messages that $\mathcal{C}$ wants $\mathcal{S}$ to retrieve.

Search-Token-Gen*$(\mathbb{W}_{\mathcal{A}}, \pi, k_{\mathcal{A}})$:
This algorithm is similar to that of KwdMac. It outputs a data structure $T_{\mathbb{W}_{\mathcal{A}}}$ encoding the set of all search keywords $t_{(k_{\mathcal{A}},w)} = H_{k_{\mathcal{A}}}(w)$, where $w \in \mathbb{W}_{\mathcal{A}}$. $T_{\mathbb{W}_{\mathcal{A}}}$ is the list of search tokens derived from $\mathbb{W}_{\mathcal{A}}$, rather than a Bloom filter.

Encrypt*$(m, Pub_{\mathcal{B}}, \pi, k_{\mathcal{A}})$:
This algorithm is identical to KwdMac's Encrypt. It outputs a ciphertext tuple $c = ([m]^B, (t_{(k_{\mathcal{A}},v)})_{v\in\mathbb{V}}, ID_{\mathcal{B}})$.

Blind-Search*$(T_{\mathbb{W}_{\mathcal{A}}}, z)$:
This algorithm differs from that of KwdMac. z is a bit specifying whether the given blind search request is conjunctive ($z = 0$) or disjunctive ($z = 1$). $\mathcal{C}$ sends $T_{\mathbb{W}_{\mathcal{A}}}$ to $\mathcal{S}$ via a communication channel providing mutual authentication and integrity and confidentiality protection (e.g. using SSL in mutual authentication mode). $\mathcal{S}$ uses the hash tables $T_{\mathcal{S}}$ and $T'_{\mathcal{S}}$ to retrieve the public-key encrypted messages whose appended blinded message keywords include each (if $z = 0$) or any (if $z = 1$) of the search tokens listed in $T_{\mathbb{W}_{\mathcal{A}}}$. The retrieved ciphertexts are sent to $\mathcal{C}$ via the communication channel used send $T_{\mathbb{W}_{\mathcal{A}}}$ to $\mathcal{S}$.

Comparing KwdMAC* to cPIR. Boneh et al.'s cPIR scheme [8] is the only request-based blind search scheme we are aware of that does not fall to the attack of Section 2.2. As such, we compare KwdMAC* to it. In KwdMAC*, the confidentiality of search keywords is protected for the same reasons as those presented in Section 5.2 for KwdMac. Note, however, that, unlike Boneh et al.'s cPIR-based PEKS scheme

[6] Recall that $\mathcal{C} = \mathcal{B}$ in the case of incoming (systematic and request-based) blind search.

Scheme	Operation	Communication Overhead (for each Keyword)
cPIR-based Scheme [8]	Sending encrypted message	$O(polylog(n)\sqrt{n \log n})$
	Retrieving encrypted message	$O(polylog(n))$
KwdMAC*	Sending encrypted message	$O(\|V\|)$
	Retrieving encrypted message	$O(\|T_{\mathrm{W}_{\mathcal{A}}}\|)$

Table 2: Comparison of Communication Efficiency: KwdMAC* vs. cPIR-based Scheme. n is the number of encrypted messages stored by a search server $\mathcal{S}$. $|V|$ is the number of blinded message keywords appended to a ciphertext. $|T_{\mathrm{W}_{\mathcal{A}}}|$ is the number of search tokens sent in a query.

[8], KwdMAC* does not aim to (and, in fact, does not) protect the confidentiality of $\mathcal{C}$'s search pattern. Let n denote the number of encrypted messages stored by a search server $\mathcal{S}$, $|V|$ the number of blinded message keywords appended to a ciphertext, and $|T_{\mathrm{W}_{\mathcal{A}}}|$ the number of search tokens sent in a query. Table 2 compares the communication costs of KwdMAC* with those of its cPIR-based alternative. With respect to this metric, KwdMAC* is more efficient if $|V| < polylog(n)\sqrt{n \log n}$ and $|T_{\mathrm{W}_{\mathcal{A}}}| < polylog(n)$. This is (asymptotically) true if $|V|$ and $|T_{\mathrm{W}_{\mathcal{A}}}|$ are bounded above by a constant, e.g. $|V| = 100$ and $|T_{\mathrm{W}_{\mathcal{A}}}| = 20$. We also observe that the cPIR scheme involves substantially more computationally intensive techniques, including homomorphic encryption and zero-knowledge proofs. To clarify the computational efficiency of KwdMAC*, let us assume that a disjunctive query sent by $\mathcal{C}$ to $\mathcal{S}$ includes 20 search tokens corresponding to keywords shorter than 16 bytes. Computing these search tokens with HMAC-SHA1 would take at most 10^{-2}ms on a P4 2.4GHz machine with 512Kb L2 cache, 256MB DDR266 RAM, with OpenSSL running on FreeBSD. Identifying the sets of ciphertexts containing at least one of these search tokens would take a bit less time if we assume the use of SHA-1 for both h and h'. This yields about 10^{-2}ms to form the query, and, since the query is disjunctive, at most 10^{-2}ms for the server to retrieve the relevant ciphertexts. While Boneh et al. [8] do not provide a computational analysis of their cPIR scheme, our intuition (based on preliminary informal analysis) is that KwdMAC* is more efficient by several orders of magnitude (assuming, for example, that $\mathcal{S}$ stores a database of 10K ciphertexts with 20 keywords each).

7. RELATED WORK

Oblivious RAM. Goldreich and Ostrovsky [15] propose an oblivious RAM scheme allowing search on encrypted data stored on remote servers while hiding data access patterns. The scheme requires a number of communication rounds (between clients and servers) which is polylogarithmic in the size of the database stored by the server.

PIR. Introduced by Chor et al. [12], private information retrieval (PIR) allows a user to retrieve a record stored in a remote (typically *unencrypted*) database server without revealing which record she is retrieving. Sion and Carbunar [23] recently pointed out the current (and potentially future) impracticality of some of the most efficient known PIR techniques, some of which are used by Boneh et al.'s cPIR-based public-key blind search scheme [8].

Search on Symmetrically Encrypted Data. Song et al. [24] propose an efficient scheme enabling parties to perform controlled and hidden search on data symmetrically encrypted by these same parties (as opposed to other parties belonging to distributed communities, as in the public-key setting). Ucal [25] designs and implements a scheme allowing search on symmetrically encrypted data *à la* Song et al. [24]. Curtmola et al. [13] propose a scheme allowing users to perform controlled and hidden search on data symmetrically encrypted by these users.

Search on Public-Key Encrypted Data. Boneh et al. [6] propose a scheme allowing authorized servers to perform search on data publicly-encrypted by arbitrary parties for users associated with the authorized servers. Baek et al. [3] propose an extension of Boneh et al.'s scheme, which provides the following benefits: users can provide search tokens to servers without establishing a secure (i.e. encrypted and authenticated) communication channel with these servers; users can generate search tokens associated with conjunctions of keywords (vs. tokens each associated with a single keyword).

Waters et al. [27] propose a scheme to perform search on an encrypted audit log. Audit logs are encrypted by audit-log servers (vs. arbitrary parties), and auditors obtain search tokens from a trusted audit escrow agent. Golle et al. [16] propose a scheme allowing any party to perform search on data which a trusted third party has publicly-encrypted and remotely stored on a data warehousing server. This differs from the case we consider wherein public-key encrypted ciphertexts are retrieved by parties who did not (necessarily and typically) produce these ciphertexts.

Kim et al. [20] propose a scheme to perform search on authenticated and encrypted (inter-domain) network routing information, without revealing search criteria to unauthorized parties. Both searchable routing information and search queries must be sent by parties belonging to the same security domain. Gu et al. [17] propose an efficient scheme enabling search servers to perform disjunctive keyword search on encrypted data publicly encrypted by arbitrary parties for parties associated with the search servers.

Abdalla et al. [1] discuss the issue of perfect, statistical, and computational consistency of keyword search on publicly-encrypted data, where consistency can be informally defined as the extent to which false positives are produced by PEKS schemes (i.e. PEKS schemes incorrectly indicate that given ciphertexts contain keywords associated with given search tokens). Abdalla et al. also propose a technique to build

PEKS schemes from anonymous hierarchical identity-based encryption schemes, and discuss both anonymous hierarchical IBE and IBE with keyword search. Baek et al. [2] propose a scheme combining general public-key encryption with EKS. Boneh and Waters [9] propose a scheme enabling general (i.e. conjunctive, subset, and range) search queries on data publicly encrypted by arbitrary parties.

Secure Index. Goh [14] proposes a scheme allowing an arbitrary party equipped with a search token for a keyword w to test in $O(1)$ time if a data structure (called a *secure index*) "contains" w. Secure indexes can be used to search on encrypted data. Chang and Mitzenmacher [11] propose two secure index schemes providing stronger security guarantees than Goh's scheme. Bellovin and Cheswick [4] propose a scheme enabling multiple cooperating parties that do not fully trust each other to share selected pieces of data, using a semi-trusted third party that mediates search queries between the cooperating parties. Ohtaki [22] proposes and implements a variant of Goh's secure index scheme [14]. Joseph et al. [19] discuss benefits and disadvantages of combining the schemes of Song et al. [24] and Goh [14].

8. CONCLUDING REMARKS

Using symmetric MAC keys and keywords associated with public-key encrypted data, `KwdMAC` is a hybrid scheme. A limitation of `KwdMAC` is that senders must receive an initial string from a warden, via a one-time registration procedure, before sending ciphertexts. We see this as a reasonable assumption when senders are part of trusted organizations and the target application is search on outgoing public-key encrypted data. For search on incoming public-key encrypted data, senders need to contact individual recipients once to obtain a secret sender key, before sending public-key encrypted messages. This is currently the price to pay for the functionality of systematic blind search on public-key encrypted data. What we would ideally like is an SBS-PED scheme whose efficiency is similar to that `KwdMAC`, but which does not require initial user registration and pre-sharing of a secret string.

9. REFERENCES

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. M. Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In *CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer-Verlag, 2005.

[2] J. Baek, R. Safavi-Naini, and W. Susilo. On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search. In *Information Security Conference (ISC'06)*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer-Verlag, 2006.

[3] J. Baek, R. Safavi-Naini, and W. Susilo. Public Key Encryption with Keyword Search Revisited. In *Australian Conference on Information Security and Privacy (ACISP'06)*, volume 4058 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[4] S. Bellovin and W. Cheswick. Privacy-Enhanced Searches Using Encrypted Bloom Filters. February 1, 2004, IACR ePrint archive, 2004. `http://eprint.iacr.org/2004/022.pdf`. Site accessed in January 2007.

[5] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer-Verlag, 2004.

[7] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[8] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W.E. Skeith III. Public Key Encryption that Allows PIR Queries. Feb. 23, 2007, IACR ePrint archive, 2007. `http://eprint.iacr.org/2007/073`. Site accessed in February 2007.

[9] D. Boneh and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Theory of Cryptography Conference (TCC'07)*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer-Verlag, 2007.

[10] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer-Verlag, 2006.

[11] Y.C. Chang and M. Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Applied Cryptography and Network Security (ACNS'05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer-Verlag, 2005.

[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–981, 1998. .

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *ACM Conference on Computer and Communications Security (CCS'06)*, pages 79–88. ACM Press, 2006.

[14] E.-J. Goh. Secure Indexes. March 16, 2004, IACR ePrint archive, 2004. `http://eprint.iacr.org/2003/216.pdf`. Site accessed in January 2007.

[15] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[16] P. Golle, J. Staddon, and B. Waters. Secure Conjunctive Keyword Search over Encrypted Data. In *Applied Cryptography and Network Security (ACNS'04)*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 2004.

[17] C. Gu, Y. Zhu, and Y. Zhang. Efficient public key encryption with keyword search schemes from pairings, 2006. Manuscript. `http://citeseer.ist.psu.edu/gu06efficient.html`. Site accessed in January 2007.

[18] K. Harrison, D. Page, and N.P. Smart. Software Implementation of Finite Fields of Characteristic

three, for use in pairing based cryptosystems. *London Mathematical Society Journal of Computing Mathemetics*, 5:181–193, 2002.

[19] L.T.A. Joseph, A. Samsudin, and B. Belaton. Efficient Search on Encrypted Data. In *IEEE International Conference on Networks*, volume 1, page 6. IEEE Computer Society, 2005.

[20] E-Y. Kim, K. Nahrstedt, L. Xiao, and K. Park. Identity-Based Registry for Secure Interdomain Routing. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pages 321–331. ACM Press, 2006.

[21] Department of Health and Human Services (USA). HIPAA (Health Insurance Portability and Accountability Act) Administrative Simplification: Enforcement, 2007. `http://www.hhs.gov/ocr/hipaa/FinalEnforcementRule06.pdf`. Site accessed in April 2007.

[22] Y. Ohtaki. Constructing a Searchable Encrypted Log Using Encrypted Inverted Indexes. In *International Conference on Cyberworlds (CW'05)*, pages 130–138. IEEE Computer Society, 2005.

[23] R. Sion and B. Carbunar. On the Computational Practicality of Private Information Retrieval. In *Network and Distributed System Security (NDSS'07)*. The Internet Society, 2007.

[24] D. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[25] M. Ucal. Searching on Encrypted Data, 2005. Manuscript. `http://citeseer.ist.psu.edu/gu06efficient.html`. Site accessed in January 2007.

[26] Federal Trade Commission (USA). The Gramm-Leach Bliley Act, 2007. `http://www.ftc.gov/privacy/privacyinitiatives/glbact.html`. Site accessed in April 2007.

[27] B.R. Waters, D. Balfanz, G. Durfee, and D.K. Smetters. Building an Encrypted and Searchable Audit Log. In *Network and Distributed System Security (NDSS'04)*, pages 205–214. The Internet Society, 2004.

Appendix: Attacks on Specific PEKS Schemes

We now present blind-search-related non-generic attacks on a number of pairing-based PEKS. We consider these attacks of independent interest from the generic attack described in Section 2. The attacks we present here exploit the following properties of analyzed PEKS schemes: search tokens generated by the analyzed schemes are linear combinations of guessable values, and the secret coefficients of these linear combinations are used as coefficients of known values in public parameters of the analyzed schemes. Using the bilinearity property of pairings, and equations used in algorithms (e.g. the `PEKS` algorithm) of the analyzed PEKS schemes, the proposed attacks verify guesses of keywords used to generate search tokens. Each attack consists of two phases. First, a dictionary of potential keywords is built. Second, each potential keyword is used to verify whether a given search token was generated for this keyword.

In the second phase, no pre-computation is needed, and each keyword verification typically requires two bilinear pairing computations. For security levels comparable with 1024-bit RSA, bilinear pairings can be computed in about 85ms [18] on a Sparc Ultra 10 computer.[7] This suggests that, in the context of the attacks presented below, a dictionary of a thousand keywords can be exhausted within $1\frac{1}{2}$ minutes, and a dictionary of a million keywords within a day.

In the following sections, we present the details of each attack on various PEKS schemes. For the sake of brevity, we only present relevant portions of the analyzed PEKS schemes.

Since the analyzed schemes rely on bilinear pairing [7], we provide a brief definition of this mathematical construct. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order p. A bilinear pairing $\hat{e}$ between $\mathbb{G}_1$ and $\mathbb{G}_2$, is a map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfying the following properties:

1. *Bilinearity*: $\hat{e}(aP_1, bP_2) = \hat{e}(P_1, P_2)^{ab}$ for all $a, b \in \{1, 2, \cdots, p\}$ and $P_1, P_2 \in \mathbb{G}_1$.
2. *Computability*: there exists a polynomial-time algorithm to compute $\hat{e}(P_1, P_2)$ from any $P_1, P_2 \in \mathbb{G}_1$.
3. *Non-degeneracy*: if P is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.

When $\hat{e}$ meets these three properties, it is said to be *admissible*.

Gu et al. PEKS Scheme

We now consider the PEKS scheme of Gu et al. [17].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
3. Generate two cryptographic hash functions $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $\mathbb{H}_2: \mathbb{G}_2 \rightarrow \{0,1\}^k$.
4. Pick a generator P of $\mathbb{G}_1$.
5. Compute $\mu = \hat{e}(P, P)$.
6. Output $\pi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, h_1, \mathbb{H}_2, \mu, P)$.

`KeyGen` (π):

1. Pick $s \in_R \mathbb{Z}_p^*$.
2. Compute $S = sP$, and set $(A_{pub}, A_{priv}) = (S, s)$.
3. Output (A_{pub}, A_{priv}).

`Token` (A_{priv}, w, π): Output $t_w = (h_1(w) + A_{priv})^{-1} P \in \mathbb{G}_1$.

The attack takes input parameters (t_w, A_{pub}, π), and is as follows.

`Dictionary Building Phase`:

Generate and output a list $L = (w_1, w_2, \cdots, w_n)$ of words probable keywords.

[7] Clock speed of the benchmark Ultra Sparc 10 are not reported by Harrison et al. [18], but these computers typically come with a 440Mhz processor.

`Keyword Verification Phase` (L, t_w, A_{pub}, π):

For $i = 1, \cdots, n$, proceed as follows:

If $\hat{e}(t_w, h_1(w_i)P + A_{pub}) = \mu$, then output w_i and exit.

If no w_i is output, then t_w corresponds to no keyword in L. Since $t_w = (h_1(w) + s)^{-1}P$, $A_{pub} = sP$, and $\mu = \hat{e}(P, P)$, note that, if $w = w_i$, then:
$\hat{e}(t_w, h_1(w_i)P + A_{pub}) = \hat{e}((h_1(w_i) + s)^{-1}P, (h_1(w_i) + s)P)) = \mu$.[8]

Baek et al. PEKS Scheme

Here, we consider the scheme of Baek et al. [3].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
3. Generate two cryptographic hash functions $\mathbb{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathbb{H}_2 \colon \mathbb{G}_2 \to \{0,1\}^k$.
4. Pick a generator P of $\mathbb{G}_1$.
5. Output $\pi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, \mathbb{H}_1, \mathbb{H}_2, P)$.

`KeyGen-Server` (π):

1. Pick $x \in_R \mathbb{Z}_p^*$ and $Q \in_R \mathbb{G}_1^*$.
2. Compute $X = xP$, and set $(A_{(pub,serv)}, A_{(priv,serv)}) = ((X, Q), x)$.
3. Output $(A_{(pub,serv)}, A_{(priv,serv)})$.

`KeyGen-User` (π):

1. Pick $y \in_R \mathbb{Z}_p^*$.
2. Compute $Y = yP$, and set $(A_{(pub,user)}, A_{(priv,user)}) = (Y, y)$.
3. Output $(A_{(pub,user)}, A_{(priv,user)})$.

`Token` $(A_{(priv,user)}, w, \pi)$: Output $t_w = A_{(priv,user)}\mathbb{H}_1(w) \in \mathbb{G}_1$.

The attack takes input parameters $(t_w, A_{(pub,user)}, \pi)$, and is as follows.

`Dictionary Building Phase`:

Generate and output a list $L = (w_1, w_2, \cdots, w_n)$ of words probable keywords.

`Keyword Verification Phase` $(L, t_w, A_{(pub,user)}, \pi)$:

For $i = 1, \cdots, n$, proceed as follows:

If $\hat{e}(A_{(pub,user)}, \mathbb{H}_1(w_i)) = \hat{e}(t_w, P)$, then output w_i and exit.

If no w_i is output, then t_w corresponds to no keyword in L. Since $A_{(pub,user)} = yP$ and $T_{w_i} = y\mathbb{H}_1(w_i)$, note that, if $w = w_i$, then:
$\hat{e}(A_{(pub,user)}, \mathbb{H}_1(w_i)) = \hat{e}(yP, \mathbb{H}_1(w_i)) = \hat{e}(P, y\mathbb{H}_1(w_i)) = \hat{e}(t_w, P)$.[9]

[8] Since $\mathbb{H}_1$ is collision resistant, the probability that this equation holds when $w_i \neq w$ is negligible.

[9] Since $\mathbb{H}_1$ is collision resistant, the probability that this equation holds when $w_i \neq w$ is negligible.

Baek et al. PEKS Scheme

We now consider the PEKS scheme of Baek et al. [2].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
3. Generate four cryptographic hash functions $\mathbb{H}_1 : \mathbb{G}_1 \to \{0,1\}^{\ell_1}$ and $\mathbb{H}_2 : \{0,1\}^* \to \mathbb{G}_1$, $\mathbb{H}_3 : \mathbb{G}_2 \to \{0,1\}^{\ell_3}$, and $\mathbb{H}_4 \colon \{0,1\}^* \to \{0,1\}^{\ell_4}$.
4. Output $\pi = (p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, \mathbb{H}_1, \mathbb{H}_2, \mathbb{H}_3, \mathbb{H}_4)$.

`KeyGen` (π):

1. Pick $s \in_R \mathbb{Z}_p^*$, and a generator P of $\mathbb{G}_1$.
2. Compute $S = sP$, and set $(A_{pub}, A_{priv}) = ((P, S), s)$.
3. Output (A_{pub}, A_{priv}).

`Token` (A_{priv}, w, π): Output $t_w = A_{priv}\mathbb{H}_2(w) \in \mathbb{G}_1$.

The attack takes input parameters (t_w, A_{pub}, π), and is as follows.

`Dictionary Building Phase`:

Generate and output a list $L = (w_1, w_2, \cdots, w_n)$ of words probable keywords.

`Keyword Verification Phase` (L, t_w, A_{pub}, π):

For $i = 1, \cdots, n$, proceed as follows:

If $\hat{e}(S, \mathbb{H}_2(w_i)) = \hat{e}(t_w, P)$, then output w_i and exit

If no w_i is output, then t_w corresponds to no keyword in L. Since $S = sP$ and $t_w = s\mathbb{H}_1(w)$, note that, if $w_i = w$, then:
$\hat{e}(S, \mathbb{H}_2(w_i)) = \hat{e}(sP, \mathbb{H}_1(w_i)) = \hat{e}(P, s\mathbb{H}_1(w_i)) = \hat{e}(t_w, P)$.[10]

PEKS Scheme Derived from Anonymous IBE

Here, we consider a PEKS scheme derived from Boyen and Waters' anonymous IBE Scheme [10].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
3. Output $\pi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e})$.

`KeyGen` (π):

1. Pick $\omega, t_1, t_2, t_3, t_4 \in_R \mathbb{Z}_p^*$, $P_0, P_1 \in_R \mathbb{G}_1$, and a generator P of $\mathbb{G}_1$.
2. Compute $\Omega = \hat{e}(P, P)^{t_1 t_2 \omega}$, and $V_i = t_i P$ for $i = 1, 2, 3, 4$.
3. Set $(A_{pub}, A_{priv}) = ((\Omega, g, P_0, P_1, V_1, V_2, V_3, V_4), (\omega, t_1, t_2, t_3, t_4))$.
4. Output (A_{pub}, A_{priv}).

`Token` (A_{priv}, w, π):

1. Pick $r_1, r_2 \in_R \mathbb{Z}_p$.

[10] Since $\mathbb{H}_1$ is collision resistant, the probability that this equation holds when $w_i \neq w$ is negligible.

2. Compute $d_0 = (r_1t_1t_2 + r_2t_3t_4)P$, $d_1 = (-\omega t_2)P + (-r_1t_2)(P_0 + wP_1)$, $d_2 = (-\omega t_1)P + (-r_1t_1)(P_0 + wP_1)$, $d_3 = (-r_2t_4)(P_0 + wP_1)$, $d_4 = (-r_2t_3)(P_0 + wP_1)$.
3. Output $t_w = (d_0, d_1, d_2, d_3, d_4)$.

The attack takes input parameters (t_w, A_{pub}, π), and is as follows.

`Dictionary Building Phase`:

Generate and output a list $L = (w_1, w_2, \cdots, w_n)$ of words probable keywords.

`Keyword Verification Phase` (L, t_w, A_{pub}, π):

For $i = 1, \cdots, n$, proceed as follows:

$$\text{If } \hat{e}((P_0 + w_iP_1), d_0)\hat{e}(V_1, d_1)\hat{e}(V_3, d_3) = \Omega^{-1},$$

then output w_i and exit. If no w_i is output, then t_w corresponds to no keyword in L.

Given the definitions of V_1, V_3, d_0, d_1, d_3, and Ω presented above, note that, if $w = w_i$,[11] then

$$\begin{aligned}\hat{e}(V_1, d_1) &= \hat{e}(t_1P, (-\omega t_2)P + (-r_1t_2)(P_0 + w_iP_1)) \\ &= \hat{e}(P, (-\omega t_1t_2)P + (-r_1t_1t_2)(P_0 + w_iP_1));\end{aligned}$$

$$\begin{aligned}\hat{e}(V_3, d_3) &= \hat{e}(t_3P, (-r_2t_4)(P_0 + w_iP_1)) \\ &= \hat{e}(P, (-r_2t_3t_4)(P_0 + w_iP_1));\end{aligned}$$

$$\begin{aligned}\hat{e}(V_1, d_1)\hat{e}(V_3, d_3) &= \hat{e}(P, (-\omega t_1t_2)P) \cdot \\ &\quad \hat{e}(P, (-r_1t_1t_2 - r_2t_3t_4)(P_0 + w_iP_1));\end{aligned}$$

$$\begin{aligned}\hat{e}((P_0 + w_iP_1), d_0) &= \hat{e}((P_0 + w_iP_1), (r_1t_1t_2 + r_2t_3t_4)P) \\ &= \hat{e}(P, (r_1t_1t_2 + r_2t_3t_4)(P_0 + w_iP_1));\end{aligned}$$

$$\begin{aligned}\hat{e}((P_0 + w_iP_1), d_0)\hat{e}(V_1, d_1)\hat{e}(V_3, d_3) &= \hat{e}(P, (-\omega t_1t_2)P) \\ &= \hat{e}(P, P)^{-t_1t_2\omega} \\ &= \Omega^{-1}.\end{aligned}$$

A similar attack can be launched against the identity-based PEKS scheme derived from Boyen and Waters' anonymous hierarchical identity-based encryption (HIBE) scheme [10] using the generic construction proposed by Abdalla et al. [1]. This variant attack is not presented here for the sake of brevity, and because it follows from the attack presented above.[12]

[11] Since $\mathbb{H}_1$ is collision resistant, the probability that the equations hold when $w_i \neq w$ is negligible.

[12] In the variant attack, the attacker uses the formula presented in the decryption algorithm of Abdalla et al.'s anonymous HIBE scheme [1] with a ciphertext suitably constructed for a guessed second-level identity. This allows to confirm that the decryption key used in the aforementioned decryption procedure was generated for the guessed identity. Since keywords correspond to second-level identities, and second-level decryption keys to search tokens, search servers can confirm guesses of keywords used to generate given search tokens.

Kim et al.'s PEKS Scheme

Here, we consider the PEKS scheme of Kim et al. [20].

`Setup` (k):

1. Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p of bit-size k.
2. Generate an admissible (i.e. computable and non-degenerate) bilinear pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
3. Generate three cryptographic hash functions $\mathbb{H}_1, \mathbb{H}_2\colon \{0,1\}^* \to \mathbb{G}_1$ and $\mathbb{H}_3\colon \mathbb{G}_2 \to \{0,1\}^k$.
4. Output $\pi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, \mathbb{H}_1, \mathbb{H}_2, \mathbb{H}_3)$.

`KeyGen` (π):

1. Pick $s, t \in_R \mathbb{Z}_p^*$, and a generator P of $\mathbb{G}_1$.
2. Compute $S = sP$, $T = tP$, and set $(A_{pub}, A_{priv}) = ((P, S, T), (s, t))$.
3. Output (A_{pub}, A_{priv}).

`Token` (A_{priv}, w, π):

- Pick $m \in_R \mathbb{Z}_q^*$ such that $gcd(m + t, p) = 1$.
- Compute $M = s(t + m)^{-1}\mathbb{H}_2(w)$.
- Output $t_w = (M, m)$.

The attack takes input parameters (t_w, A_{pub}, π), and is as follows:

`Dictionary Building Phase`:

Generate and output a list $L = (w_1, w_2, \cdots, w_n)$ of words probable keywords.

`Keyword Verification Phase` (L, t_w, A_{pub}, π):

For $i = 1, \cdots, n$, proceed as follows:

If $\hat{e}(M, T + mP) = \hat{e}(S, \mathbb{H}_2(w_i))$, then output w_i and exit.

If no w_i is output, then t_w corresponds to no keyword in L. Since $M = s(t + m)^{-1}\mathbb{H}_2(w_i)$ if $w = w_i$, and $S = sP$ and $T = tP$, then:
$\hat{e}(M, T + mP) = \hat{e}(s(t + m)^{-1}\mathbb{H}_2(w_i), (t + m)P) = \hat{e}(sP, \mathbb{H}_2(w_i)) = \hat{e}(S, \mathbb{H}_2(w_i))$.[13]

[13] Since $\mathbb{H}_1$ is collision resistant, the probability that this equation holds when $w_i \neq w$ is negligible.