Privacy-Preserving Outsourcing of Brute-Force Key Searches

Ghassan O. Karame
Dept. of Computer Science
ETH Zurich, Switzerland
karameg@inf.ethz.ch

Srdjan Čapkun
Dept. of Computer Science
ETH Zurich, Switzerland
capkuns@inf.ethz.ch

Ueli Maurer
Dept. of Computer Science
ETH Zurich, Switzerland
maurer@inf.ethz.ch

ABSTRACT

In this work, we investigate the privacy-preserving properties of encryption algorithms in the special case where encrypted data might be brute-force decrypted in a distributed setting. For that purpose, we consider a problem where a supervisor holds a ciphertext and wants to search for the corresponding key assisted by a set of helper nodes, without the nodes learning any information about the plaintext or the decryption key. We call this a *privacy-preserving* cryptographic key search. We provide a model for privacy-preserving cryptographic searches and we introduce two types of privacy-preserving key search problems: *plaintext-hiding* and *key-hiding* cryptographic search. We show that a number of private-key and public-key encryption schemes enable the construction of efficient *privacy-preserving solvers* for plaintext hiding searches. We also discuss possible constructions of privacy-preserving solvers for key-hiding cryptographic searches.

Our results highlight the need to consider the property of enabling efficient privacy-preserving solvers as an additional criterion for choosing which cryptographic algorithm to use.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and Protection

General Terms

Design, Security, Theory.

Keywords

Distributed Computing, Brute-Force Key Search, Privacy-Preserving Solvers.

1. INTRODUCTION

The main premise of data encryption is to ensure the privacy and confidentiality of the data against passive and active adversaries. As a result of ongoing cryptanalysis of encryption schemes, the literature contains a large number of reported attacks on existing encryption schemes [11, 12]. These attacks commonly exploit the vulnerabilities of encryption schemes (e.g., differential cryptanalysis, weak password-based encryption, related-key attacks, etc.) and reduce the search space to brute-force the keys. Examples include attacks on DES [10], RC4 [40], small private RSA exponents [12, 21, 34, 36], etc..

Nowadays, the time, effort and cost of brute-force breaking encrypted messages is considerably reduced with the emergence of distributed computing platforms [1–3] that rely on the contribution of millions of volunteers. For example, DES (16-round, 56-bit key) was broken in a record time of 22 hours and 15 minutes [4] with the help of EFF's supercomputer "Deep Crack" [5] and approximately 100,000 volunteer nodes from the distributed.net platform [1]. Given that the security offered by current encryption schemes is only expected to decrease with the growth of computational resources[1], the messages—encrypted and stored in long-term storages today—are likely to be successfully brute-forced in the future.

In distributed computing platforms, the helper nodes that brute-force search a given ciphertext (e.g., when the encryption function is weak due to cryptanalysis or when the secret key is short) will typically acquire (and report) the corresponding plaintext and the decryption key. This provides the assurance/ability e.g., for platforms administrators and governments to control the type of tasks and cryptographic searches that are being executed in these platforms.

In this work, we show that distributed cryptographic searches can be performed without leaking information about the sensitive search instances to the parties that are involved in the search process. For that purpose, we consider a setting where an entity, the supervisor, has access to a ciphertext (and possibly its corresponding plaintext) and wants to brute-force search for the corresponding decryption key with the help of a set of helper nodes given the following constraint: the helpers should not learn (parts of) the input plaintext and/or the decryption key. We refer to the outsourced key searches in which the helpers do not learn the plaintext or the key by *plaintext-hiding* and *key-hiding* searches, respectively. We precisely define these notions and we show that a number of public-key cryptosystems enable a straightforward construction of plaintext-hiding and key-hiding searches. We also show that privacy-preserving solvers for plaintext-hiding searches can be efficiently con-

[1] NIST [6] deems that the current key length needs to be increased after 2010.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCSW'11, October 21, 2011, Chicago, Illinois, USA.
Copyright 2011 ACM 978-1-4503-1004-8/11/10 ...$10.00.

structed in most block cipher modes, and can be directly integrated in current distributed computing platforms. By leveraging on these plaintext-hiding solvers, we propose constructions of privacy-preserving solvers for key-hiding cryptographic searches in block ciphers.

Our results show that current cryptographic searches (e.g., the DES challenge [4], the WPA cracker [7]) can be effectively used to conduct massive brute-force decryption of sensitive ciphertexts (when such a search is feasible) while making the nature and purpose of an outsourced cryptographic search oblivious to the participating nodes.

The property of enabling efficient privacy-preserving solvers therefore emerges as a relevant property of encryption schemes. We argue that this property should be considered as an additional criterion in the choice of cryptographic algorithms and in the deployment of server-assisted decryption applications. In the special case when this property is required to enable both key-hiding and plaintext-hiding searches without the knowledge of any plaintext (e.g., in the case where users wish to recover their lost disk decryption keys), we show that plan-ahead encryption schemes can enable the efficient construction of the corresponding privacy-preserving solvers.

The rest of the paper is organized as follows. In Section 2, we introduce the notion of a privacy-preserving solver. In Section 3, we propose constructions of privacy-preserving solvers for plaintext-hiding searches based on public-key and private-key encryption schemes. In Section 4, we discuss variant constructions of private solvers for key-hiding problems. In Section 5, we propose plan-ahead encryption schemes that enable, by design, the efficient construction of privacy-preserving solvers. In Section 6, we survey the related work and we conclude the paper in Section 7.

2. PRIVACY-PRESERVING SOLVERS

In this section, we introduce the notions and the model that we will adopt throughout the paper. We first restate some known definitions: the search problem and problem solver.

Definition 1. **(Search problem)**
A search problem $\mathbb{S}$ *is characterized by the instance space $\mathcal{X}$ (and an implicit distribution over it), the solution space $\mathcal{Y}$, and the verification predicate $P_{\mathbb{S}} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, with $P_{\mathbb{S}}(x, y) = 1$ if and only if y is a solution for instance x.*

Definition 2. **(Solver)**
A solver *for a search problem $\mathbb{S}$ is an algorithm that takes as input an $x \in \mathcal{X}$ and returns a $y \in \mathcal{Y}$ if $P_{\mathbb{S}}(x, y) = 1$ and $\perp$ if no solution exists.*

We consider a setting where an entity wants to solve a search problem with the help of a computationally powerful server. Such a setting can be described as a pair (A, B) of interacting algorithms where A is run by the entity and B is run by the server (or a set of servers). Typically, A must be efficient (i.e., polynomial-time in an asymptotic definition of efficiency).

In the sequel, we assume $\mathbb{C}$ to be some super-polynomial or even exponential class of algorithms. Informally, our assumptions on the complexity class $\mathbb{C}$ capture the intuition that while some encryption functions are not $\mathbb{C}$-secure (and therefore might be broken by a "powerful" algorithm B' in $\mathbb{C}$), it is infeasible for any $B' \in \mathbb{C}$ to break other hard problems (e.g., the one-time pad encryption). We now capture the notion of computational independence relative to $\mathbb{C}$.

Definition 3. **(Computational Independence)**
Two random variables U and V are called computationally independent*, relative to complexity class $\mathbb{C}$, if there exists a pair (U', V') of statistically independent random variables such that every distinguisher in complexity class $\mathbb{C}$ has only negligible advantage in distinguishing the pair (U', V') from the pair (U, V).*

We introduce the notion of a search problem with privacy constraints.

Definition 4. **(Search Problem with Privacy Constraints)**
A search problem with privacy constraints *is a search problem where the instances and/or solutions consist of two parts, labeled public and secret (private). In other words, either $\mathcal{X} = \mathcal{X}_p \times \mathcal{X}_s$, or $\mathcal{Y} = \mathcal{Y}_p \times \mathcal{Y}_s$, or both.*

That is, we assume that the instances x and/or the solutions y are pairs[2], denoted by $x = (x_p, x_s)$ and $y = (y_p, y_s)$, respectively.

Definition 5. **(Privacy-preserving Solver)**
A pair (A, B) of interacting algorithms, where A is efficient, is a (computationally) privacy-preserving solver *for a search problem $\mathbb{S}$ with privacy constraints if two conditions hold:* (i) *The pair (A, B) is a solver for $\mathbb{S}$, and* (ii) *For every algorithm B' in complexity class $\mathbb{C}$, the transcript between A and B' is (computationally) independent relative to $\mathbb{C}$ of all the private parts of the instance and solution.*

The goal of the privacy-preserving solver is to solve a search problem with privacy constraints such that an entity running algorithm A learns the input instance and the solution to the problem, whereas a malicious entity running algorithm B (that assists in solving the problem) or any algorithm B' in $\mathbb{C}$ does not learn any information about the secret part of the instance (x_s) or of the solution (y_s). The notion of "learning nothing" is captured by the requirement that the transcript of interaction between A and B must be independent (or at least computationally independent relative to $\mathbb{C}$) of (x_s, y_s).

We now introduce two notions: *plaintext-hiding* and *key-hiding* key search problems.

Definition 6. **(Plaintext-Hiding and Key-Hiding Key Search Problems)**
For a private-key encryption scheme, Enc*, with message space $\mathcal{P}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$, the* key search problem $\mathbb{K}^{Enc}$ *is defined by $\mathcal{X} = \mathcal{C} \times \mathcal{P}$, $\mathcal{Y} = \mathcal{K}$, and the predicate P, with $P((c, p), k) = 1$ if and only if c is the ciphertext for p created using* Enc *under key k. One can derive two key search problems with privacy constraints, as follows.*

- *The* plaintext-hiding *key search problem $\mathbb{K}_p^{Enc}$ is defined as the key search problem where $\mathcal{X}_s = \mathcal{P}$ and $\mathcal{Y}_p = \mathcal{Y} = \mathcal{K}$.*

- *The* key-hiding *key search problem $\mathbb{K}_k^{Enc}$ is defined as the key search problem where $\mathcal{X}_s = \mathcal{P}$ and $\mathcal{Y}_s = \mathcal{Y} = \mathcal{K}$.*

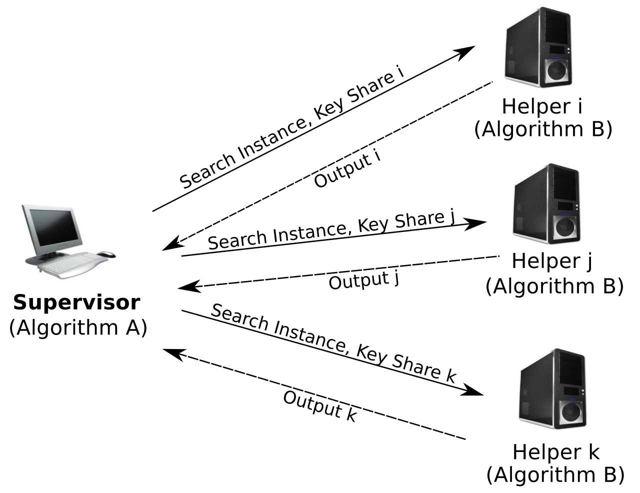

Figure 1: System model: the supervisor runs algorithm A that interfaces with the helper nodes. The supervisor distributes the key space among the nodes and gathers their results. In a privacy-preserving solver, the sensitive search instances and/or outputs are not revealed to the helpers.

We envision two possible use-cases for plaintext-hiding key search problems: *(i)* the supervisor possesses a ciphertext/plaintext pair and would like to acquire the decryption key without the helpers learning the plaintext and *(ii)* the supervisor only possesses a ciphertext and would like to acquire the key (and therefore to acquire the plaintext) without the nodes learning the corresponding plaintext, even if they acquire the correct decryption key (Section 5).

On the other hand, in key-hiding searches, the supervisor possesses a ciphertext/plaintext pair and would like to acquire the corresponding key without the helpers learning it. In these searches, the helpers cannot therefore be given the ciphertext unless they are not able to identify the output plaintext; we show that solvers for key-hiding searches can be constructed based on their plaintext-hiding counterparts.

2.1 System and Attacker Model

The cryptographic search that we consider is inherently parallelizable and its natural implementation is that each helper gets a share of the key space through which it searches. However, since a subset of the helper nodes might collude, the supervisor must ensure with high probability that no node can learn the sensitive parts of the search problem (i.e., the plaintext or the key).

To solve a cryptographic search problem, the helper nodes execute the solver algorithm B, whereas the supervisor executes algorithm A (Figure 1). We assume that algorithm A divides the key space among the helper nodes and, later on, gathers their individual results. We point out that the functionality and load of algorithm A are equivalent to that pertaining to the central server of existing distributed platforms (e.g., distributed.net [1]).

Each helper node runs algorithm B given the same input instances; their search is however performed on different shares of the key space. We assume that the helper nodes do not know any additional information about the outsourced ciphertexts besides the information that is meant to be released by the supervisor[3] (i.e., the inputs of the search problem). We further assume that the helper nodes can collude and execute any algorithm B' to acquire information about the instances that are meant to be kept secret during the search process. We assume that B' is in the complexity class $\mathbb{C}$. We further assume that there exist a $\mathbb{C}$-secure pseudorandom generator G and a $\mathbb{C}$-secure pseudorandom function F.

We assume that—except for the original plaintext and the resulting ciphertext—no intermediate results of the encryption/decryption algorithms are available to the supervisor. We denote by $\mathcal{C}$ the set of possible ciphertexts, by $\mathcal{P}$ the set of possible plaintexts, and by $\mathcal{K}$ the set of possible keys. Throughout this paper, we consider encryption functions (e.g., DES, AES) as a "black-box" that behaves as a pseudorandom permutation.

We focus on the case where the disclosure of non-sensitive parts of the plaintext is insufficient to identify the key[4]. We consider the scenarios where it is feasible to retrieve the key, e.g., when the key search space is reduced following cryptanalysis of an encryption algorithm, when the keys are short, or when the password of a password-based encryption scheme is weak.

In our privacy-preserving schemes, we assume that the outsourced search instances are public and therefore not only available to the helper nodes but also to other entities (e.g., the helpers are not trusted to keep the search instances confidential). Here, the advantage of any other entity, running an algorithm in $\mathbb{C}$, to acquire sensitive information from the search instances is, in the worst case, similar to that of the helper nodes (given that they at most have the same information at their disposal and they both run algorithms in complexity class $\mathbb{C}$). In analyzing the security of privacy-preserving solvers, we therefore do not treat this case separately, but we rather focus on the ability of the helper nodes to acquire the sensitive information from the search process. We point out, however, that both cases are captured by the requirement that the private search instances and solutions should not be acquired by any algorithm B' in $\mathbb{C}$.

In Section 4.3, we assume that up to N helper nodes can collude or share the search instances/solutions with an external attacker. In this case, we assume that the communication between the supervisor and the helpers is performed over a confidential channel.

3. PRIVACY-PRESERVING SOLVERS FOR PLAINTEXT-HIDING KEY SEARCHES

In this section, we describe constructions of privacy-preserving solvers for plaintext-hiding searches. First, we outline straightforward examples of privacy-preserving solvers for plaintext-hiding searches based on public key systems and additive stream ciphers. We then show how to construct

[2] In this case, $\mathcal{X}$ and $\mathcal{Y}$ refer to the sets of all possible pairs (x_p, x_s) and (y_p, y_s), respectively.

[3] Nevertheless, we assume that the helper nodes know some redundancy information about the plaintext (e.g., the plaintext is an English text).

[4] Otherwise, if the supervisor has access to a second ciphertext/plaintext pair in which the plaintext is not sensitive, the supervisor can then simply give the second ciphertext/plaintext pair to the helpers to find the key.

privacy-preserving solvers for plaintext-hiding searches in the CBC (and CFB, PCBC) mode encryption.

3.1 Examples of Plaintext-Hiding Searches

We start by outlining straightforward examples of solvers for plaintext-hiding searches.

The Case of Public-Key Cryptosystems.

Solving key searches in public-key cryptosystems can be achieved independently of the plaintext and/or the ciphertext that the supervisor possesses (when such a search is feasible[5]).

A supervisor interested in finding a private key to decrypt a given ciphertext can simply send the public key to the helpers that search for the corresponding private key; the helpers would have no information about the original plaintext. As a special case, the malleability property of some public-key cryptosystems could also be exploited to enable plaintext-hiding searches. Consider a malleable encryption scheme Enc that enables the construction of a transformed pair $(c_2, p_2) \in \mathcal{C} \times \mathcal{P}$ from any known ciphertext/plaintext pair $(c_1, p_1) \in \mathcal{C} \times \mathcal{P}$ such that *(i)* c_2 is the encryption of p_2 using the same unknown key that encrypts p_1 into c_1 and *(ii)* (c_1, p_1) and (c_2, p_2) are (computationally) independent. Following from Definitions 5 and 6, it is easy to construct a privacy-preserving solver for $\mathbb{K}_p^{Enc}$.

The Case of Stream Ciphers, CTR and OFB Mode Encryption.

We now consider the case of an additive stream cipher encryption scheme. Recall that, for such schemes, the plaintext p for a ciphertext c is computed as $p = c \oplus G(k)$, where $G(.)$ is a pseudorandom generator. In what follows, we show that there exist efficient privacy-preserving solvers for plaintext-hiding searches based on additive stream ciphers.

The plaintext-hiding key search problem based on stream ciphers $\mathbb{K}_p^{Stream}$ is defined by the predicate P with $P((c,p),k) = 1$ if $(c,p) \in \mathcal{C} \times \mathcal{P}$ is a matching ciphertext-plaintext pair created using the stream-cipher encryption scheme with key $k \in \mathcal{K}$. We construct a solver (A, B) for $\mathbb{K}_p^{Stream}$ as follows: *(i)* A computes $c \oplus p$ and sends it to B. *(ii)* On input $c \oplus p$, B searches for the key $\widehat{k}$ such that $G(\widehat{k}) = c \oplus p$. If found, B sends $\widehat{k}$ to A. *(iii)* A checks whether $P((c,p),\widehat{k}) \overset{?}{=} 1$. If so, it outputs $y = k = \widehat{k}$, otherwise it outputs $\perp$.

The transcript T of the interaction between A and B solely contains the values $c \oplus p$ and $\widehat{k}$. Since $c \oplus p = G(k)$, T only depends on k and is thus independent of p (hence also computationally independent of p irrespective of any algorithm B' in $\mathbb{C}$ running on the helpers). (A, B) is therefore a privacy-preserving solver for $\mathbb{K}_p^{Stream}$.

Note that since the Counter (CTR) and the Output Feedback (OFB) mode encryption are instances of additive stream ciphers, plaintext-hiding search problems based on CTR and OFB modes also have efficient privacy-preserving solvers.

3.2 Plaintext-Hiding Searches in the Cipher Block-Chaining (CBC) Mode Encryption

We show that plaintext-hiding key-searches based on the CBC mode encryption can also be solved using a privacy-preserving solver.

Recall that in a CBC encryption scheme: $p_i = D_k(c_i) \oplus c_{(i-1)}$, where D_k denotes the block-cipher decryption function (e.g., DES, AES) using key k, p_i and c_i denote the ith block of plaintext and ciphertext, respectively.

Now, consider the plaintext-hiding search problem $\mathbb{K}_p^{CBC}$. Recall that $\mathbb{K}_p^{CBC}$ is defined by $\mathcal{X} = \mathcal{C} \times \mathcal{P}$, $\mathcal{Y} = \mathcal{K}$ and by the predicate P with $P((c,p),k) = 1$ if (c,p) is a matching ciphertext-plaintext pair created using the CBC mode encryption with key k.

LEMMA 1. *If G is a $\mathbb{C}$-secure pseudorandom generator, and $M \in \{0,1\}^n$, then $C = G(s) \oplus M$, where s is chosen uniformly at random from $\{0,1\}^n$, is computationally independent of M relative to complexity class $\mathbb{C}$ (refer to Appendix A for the proof).*

THEOREM 3.1. *For any CBC encryption scheme, there exists an efficient privacy-preserving solver for the plaintext-hiding key search problem $\mathbb{K}_p^{CBC}$.*

Proof: We prove that $\mathbb{K}_p^{CBC}$ has an efficient privacy-preserving solver by construction; let (A, B) refer to the solver of $\mathbb{K}_p^{CBC}$. A takes as input $x = (c,p)$ and divides c and p into blocks of equal length conforming with the operation of block-ciphers. A interacts with algorithm B in the following way:

1. A computes $c_0 \oplus p_1$ and sends it to B.

2. On input c_1 and $c_0 \oplus p_1$, B searches for $\widehat{k} \in \mathcal{K}$ such that $c_0 \oplus p_1 = D_{\widehat{k}}(c_1)$, where D is the decryption function in block-ciphers (e.g., DES, AES). If found, B sends $\widehat{k}$ to A.

3. Given $\widehat{k}$, A checks whether $P((c,p),\widehat{k}) \overset{?}{=} 1$. In the case where more than one candidate $\widehat{k}$ satisfies the predicate function P (i.e., if a collision occurs), A checks the validity of each candidate on other (ciphertext/plaintext) blocks it possesses. Once a unique solution $\widehat{k}$ is found, A outputs $y = k = \widehat{k}$, otherwise it outputs $\perp$.

The transcript T of the interaction between A and B solely contains the values c_1, $c_0 \oplus p_1$ and $\widehat{k}$. Note that only c_1 is sent to B; all the remaining blocks in c (including $c_0 = \text{IV}$) are kept secret. In what follows, we show that T is independent of p. Here, two main cases emerge:

1. IV *is chosen uniformly at random:* Since IV is kept secret from B, $\text{IV} \oplus p_1$ acts as a one-time pad encryption of p_1. Hence, $\text{IV} \oplus p_1$ is independent of p_1 (and of p), even in the case when B (or B') is computationally unbounded.

2. IV *is a pseudorandom string:* Let $\text{IV} = G(s)$, where G is a $\mathbb{C}$-secure pseudorandom number generator and s is chosen uniformly at random, then $\text{IV} \oplus p_1$ is computationally independent of p_1 relative to $\mathbb{C}$, given Lemma 1.

Given this, c_1 is (computationally) independent of p_1; this is the case since c_1 encrypts $\text{IV} \oplus p_1$ using key k, and $\text{IV} \oplus p_1$ is (computationally) independent of p_1. Thus, T is (computationally) independent of p. (A, B) is therefore a privacy-preserving solver for $\mathbb{K}_p^{CBC}$. □

[5] This could be the case for instance when the private RSA exponents are small [12, 21, 34, 36].

We point out that A is efficient in the sense that it has only to perform a single decryption and one XOR operation to solve $\mathbb{K}_p^{CBC}$. Similar solvers can be constructed for search problems based on the PCBC and CFB mode encryption.

Remark 1. *Note that a privacy-preserving solver for $\mathbb{K}_p^{CBC}$ can be constructed given the knowledge of any two consecutive ciphertext blocks. This is the case since the randomness (or pseudorandomness) of the* IV *is propagated through the block encryption function to the subsequent blocks. That is, similar to the above analysis, it can be shown that any transcript T that contains c_i, $c_{(i-1)} \oplus p_i$ $(i \geq 1)$, and $\widehat{k}$ is computationally independent relative to $\mathbb{C}$ of p_j where $j \in [1..i]$ given a random (or pseudorandom)* IV*. This can be useful in the case where the supervisor only possesses part of ciphertext.*

Remark 2. *The above privacy-preserving solver can be easily integrated in existing distributed computing platforms since current cryptographic searches are often performed using a single pair of ciphertext/plaintext blocks*[6]*. That is, in our solver, B searches for the key that encrypts $c_{(i-1)} \oplus p_i$ into c_i, while in existing platforms it searches for the key that encrypts p_i into c_i. Clearly, both search problems require similar functionality and computational capability.*

Remark 3. *Note that the supervisor can detect malicious nodes that return incorrect results in a subset of their assigned tasks by embedding indistinguishable pre-computed tasks within the search tasks of the nodes [28, 35, 46]. Since the supervisor already knows the solutions of these embedded tasks, he can then ensure, with high probability, that malicious nodes are detected. Here, the probability of detecting misbehavior is given by $1-(1-p)^n$, where p is the probability that a node misbehaves per task and n is the number of pre-computed tasks assigned to each node.*

Large Key Space.

We now proceed to analyze the security of our scheme in the case when the key size is larger than the plaintext/ciphertext size (i.e., $|\mathcal{K}| > |\mathcal{P}|$). In this case, algorithm A might need to invoke several instantiations of the same search problem using multiple ciphertext/plaintext blocks that were encrypted using the same unknown key (and the same IV).

Recall that given the propagation of the (pseudo)randomness of IV through the chaining of block ciphers, it is infeasible for the helpers to acquire any information about p in a single instantiation of $\mathbb{K}_p^{CBC}$. However, when multiple different instantiations of the same $\mathbb{K}_p^{CBC}$—using the same IV—are invoked (Figure 2), the secrecy of p might be compromised. This is due to the fact that if the helpers have access to different (not necessarily consecutive) ciphertext blocks, then this reveals relations between their corresponding plaintexts (due to block-chaining); in the worst case, the helpers might be able to acquire these plaintexts.

We illustrate this by means of an example and we consider two different instantiations of $\mathbb{K}_p^{CBC}$ where the input to algorithm B consists of the pairs: $(c_{(i-1)} \oplus p_i, c_i)$ and $(c_{(j-1)} \oplus p_j, c_j)$, such that $i, j \geq 1, j \neq i$, $(j-1) \neq i$ and $j \neq (i-1)$.[7]

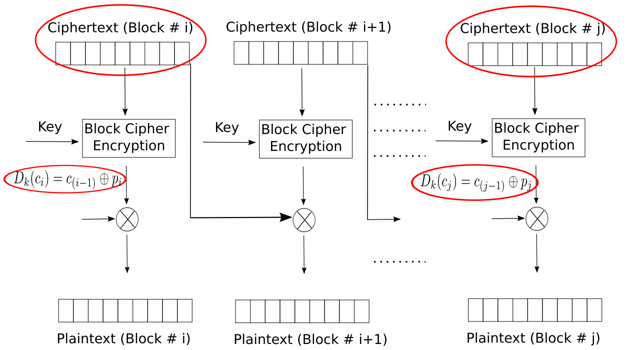

Figure 2: Outsourcing multiple instances of the same plaintext-hiding CBC-based search problem. The circled instances are the inputs to algorithm B. After recovering the key, the helpers might be able to acquire all the plaintext-blocks whose indexes fall between $[i+1, j]$.

Once the key k is found by B, the helpers can solve the (non-private) search problem $\mathbb{S}^{CBC}$ for the plaintext blocks that are located between p_i and p_j, given k, c_j and c_i. Note that if the helpers are able to solve $\mathbb{K}_p^{CBC}$ to acquire the correct key k, then they might be able to solve[8] $\mathbb{S}^{CBC}$ (if $|\mathcal{K}| \geq |\mathcal{P}^{|j-i|}|$).

In general, $\forall i, j \geq 1$ such that $j \neq i$, $(j-1) \neq i$ and $j \neq (i-1)$, the helpers might acquire the plaintext under different multiple instantiations of $\mathbb{K}_p^{CBC}$ if it is feasible to search in $|\mathcal{P}^{|j-i|}|$. This is the case if the helpers can execute an unbounded algorithm or if the search for a solution in a set of size $|\mathcal{P}^{|j-i|}|$ is feasible in $\mathbb{C}$.

This analysis also applies for plaintext-hiding search problems based on the PCBC and CFB mode encryption.

In the case of the CTR and OFB mode encryption (Section 3.1), A can safely invoke several instantiations of the same search problem based on these modes when $|\mathcal{K}| > |\mathcal{P}|$. In fact, in these modes, the transcript of interaction between A and B is independent of both c and p even if the helpers execute an unbounded algorithm.

Additional Properties: Deniability.

The aforementioned privacy-preserving solvers rely on the (computational) independence between p and its "one-time pad encryption" (with an unrevealed ciphertext block). It is well known that one-time pad encryption constitutes a shared-key deniable encryption scheme [17][9]. Therefore, algorithm A could conduct brute-force searches on sensitive ciphertexts while claiming to perform "legitimate" cryptographic searches (e.g., for academic purposes) when coerced.

As a simple example, algorithm A—equipped with a ciphertext/plaintext (c, p) pair created in the CBC mode encryption using a secret key k—can pick a "non-suspicious" string, str (e.g., str $\leftarrow$ "INNOCENT"), and sets $\overline{p_1} \leftarrow str$. A then computes $\overline{\text{IV}} = \text{IV} \oplus p_1 \oplus \overline{p_1}$ using the first genuine

[6] E.g., the DES distributed.net client *v. 20000229* takes as input a single block of ciphertext/plaintext pair.

[7] Otherwise, if $(j-1) = i$, the helpers can extract $p_j = c_{(j-1)} \oplus p_j \oplus c_i$. A similar case occurs when $j = (i-1)$.

[8] Recall that $\mathcal{P}$ refers to the set of all possible plaintexts. Given some "redundancy" information (e.g., p is English text, or p comprises of ASCII characters), $|\mathcal{P}|$ is not necessarily exponentional with respect to the length of p.

[9] An encryption scheme is deniable if the ciphertext can appear as an encryption of another plaintext [17].

plaintext block, p_1, of p and asks the helpers to search for the key k that encrypts the plaintext $\overline{p_1}$ into $c = (\overline{\text{IV}}, c_1)$. It is easy to see that k also encrypts the original plaintext p into c. We point out that this solver is a privacy-preserving solver for $\mathbb{K}_p^{CBC}$, since p_1 and $\overline{p_1}$ are independent. This privacy-preserving search will not also raise suspicion due to the fact that $\overline{p_1}$ is a plausible English block. This analysis also applies for plaintext-hiding search problems based on the CTR, OFB, CFB and PCBC mode encryption.

This deniability property could be prevented by requiring that the input of each outsourced search problem consists of several (> 3) consecutive blocks in the CBC, PCBC and CFB mode encryption. In this case, it is very hard for the supervisor to conceal all the consecutive sensitive plaintext-blocks from the helpers (see the related discussion in Section 3). This countermeasure is, however, ineffective in searches based on the CTR and OFB mode encryption.

4. ENABLING KEY-HIDING SEARCHES

In this section, we propose a number of constructions of privacy-preserving solvers for key-hiding searches. Namely, we show that key-hiding searches can be achieved by constructing Private Information Retrieval (PIR) solvers for plaintext-hiding searches. We also propose a privacy-preserving solver, that offers probabilistic guarantees, to enable key-hiding searches in symmetric encryption algorithms.

4.1 Exemplary Key-Hiding Searches in Public-Key Cryptosystems

Some public-key cryptosystems enable a straightforward construction of privacy-preserving solvers for key-hiding searches. For example, consider the case where the supervisor has the public key $c = g^x$ and would like to know the secret key $x \in \mathbb{Z}_p^*$ without revealing it to the helpers (e.g., ElGamal cryptosystem [24]). The supervisor chooses a random $y \in \mathbb{Z}_p^*$, sends $c \times g^y$ to the helpers and asks them to compute its discrete logarithm r to the base g. The helpers perform the search and send r to the supervisor. Since the latter knows y, it can simply extract $x = r - y$ modulo the group order; the helpers, however, cannot learn x from r (g^y is a random group element and therefore $c \times g^y$ is independent of c)[10].

4.2 Key-Hiding Searches using PIR Schemes

Key-hiding cryptographic searches can be achieved in theory by combining privacy-preserving solvers for the corresponding plaintext-hiding searches along with private information retrieval (PIR) schemes [16, 20, 26]. PIR schemes enable the supervisor to retrieve the key and the plaintext (that is known to the supervisor) relative to a ciphertext without revealing the retrieved values to the helpers.

PIR key-hiding solvers provide computational secrecy of the key and the plaintext; they, however, incur a large computational complexity on algorithm A.

PIR schemes can be used to solve the key-hiding problem in the following way: *(i)* A outsources a *plaintext-hiding* search problem[11] to B, *(ii)* B performs the search using all candidate keys in $\mathcal{K}$, *(iii)* A then uses a (computational) PIR scheme to query B for the decryption key that corresponds to the desired private input. In this case, (A, B) is a privacy-preserving solver for the above key-hiding (and plaintext-hiding) problem. Note that the computational complexity of A and the communication complexity between A and B correspond to those of the PIR protocol in use.

Example 1. *We consider the private block retrieval protocol of [26] as an example. This protocol is an extension of the PIR protocol in [16] and enables a user to retrieve the ith block of an n-bit database, without revealing to the database the value of i nor the content of the block that is being retrieved. The main intuition behind [26] is to divide the database into blocks of bits and associate each block with a distinct small prime. The database uses the Chinese Remainder Theorem to encode each database chunk modulo its corresponding prime power. Given an n-bit database divided into blocks of size l-bits, the communication complexity between the user and the database is* $O(\log(n))$ *and the computational complexity of the user is no more than* $4\sqrt{nl}$ *group operations. When used as a privacy-preserving solver for a key-hiding problem,* $n = |\mathcal{K}|$ *and* $l = |k|$*; the computational complexity of algorithm A is* $O(\sqrt{|\mathcal{K}|})$ *group operations and the communication complexity between A and B is* $O(\log(|\mathcal{K}|)) = O(|k|)$.

4.3 N-Private Solvers for (Symmetric) Key-Hiding Searches

Privacy-preserving solvers for key-hiding searches in symmetric cryptography are more challenging to construct when compared to their plaintext-hiding counterparts; if algorithm B is given a ciphertext c and can perform exhaustive search over the key space $\mathcal{K}$, then it can simply search for the key that results in a plausible decryption of c (e.g., an english plaintext). Therefore, B should not be able to identify the output plaintext in these searches. This also suggests that a privacy-preserving solver for a key-hiding search in symmetric cryptography will also be a privacy-preserving solver for the corresponding plaintext-hiding search problem. As shown in Section 4.2, these solvers are typically not efficient when dealing with an attacker that controls all the nodes in the network. In the sequel, we show that, if an adversary in class $\mathbb{C}$ can compromise at most N nodes in the network, then efficient privacy-preserving solvers—*N-private solvers*—for key-hiding searches can be constructed. We start by defining an *N-private solver*.

Definition 7. **(N-Private Solver)** *A pair (A, B) of interacting algorithms is an* N-private solver *for a search problem $\mathbb{S}$ with privacy constraints if two conditions hold:* (i) *The pair (A, B) is a solver for $\mathbb{S}$, and* (ii) *For every adversary M in $\mathbb{C}$ that controls up to N helper nodes, the probability that it acquires any meaningful information about the private parts of the instances and/or solutions is negligible.*

Given this definition, we present in what follows a construction for an efficient N-private solver for $\mathbb{K}_k^{CBC}$ based on the previously proposed privacy-preserving solver for $\mathbb{K}_p^{CBC}$. We point out, however, that variant N-private solvers can be also constructed to enable key-hiding searches in CTR, OFB, CFB and the PCBC mode encryptions; where appropriate, we will point to the differences between these solvers.

[10] Similar approaches are used in private signature computations [43] and in zero-knowledge proofs of knowledge protocols [27].

[11] Here, we assume that there exists an efficient solver for the variant plaintext-hiding search problem (e.g., in the CBC mode encryption).

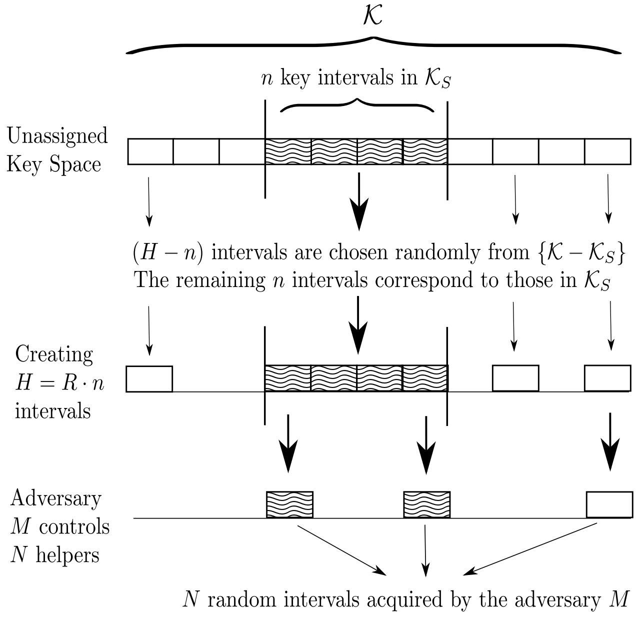
$\mathcal{K}$
n key intervals in $\mathcal{K}_S$
Unassigned Key Space
$(H-n)$ intervals are chosen randomly from $\{\mathcal{K}-\mathcal{K}_S$
The remaining n intervals correspond to those in $\mathcal{K}_S$
Creating $H = R \cdot n$ intervals
Adversary M controls N helpers
N random intervals acquired by the adversary M

Figure 3: $H = R \cdot n$ key intervals are assigned to the helpers. These key intervals contain the n intervals that comprise the reduced key set $\mathcal{K}_S$. The remaining $(H-n)$ intervals are chosen uniformly at random from $\{\mathcal{K}-\mathcal{K}_S\}$. In an N-private solver, the probability that an adversary derives any information about the key should be negligible if it acquires a maximum of N key samples. Here, we depict the worst-case scenario where the intervals of $\mathcal{K}_S$ are contiguous; our analysis however applies to any other distribution of these key intervals within $\mathcal{K}$.

Similar to Section 2.1, we consider a setting where the supervisor possesses a ciphertext-plaintext pair (c, p) created using an unknown key $k \in \mathcal{K}$. However, we assume that *(i)* $|\mathcal{K}|$ is too large to be brute-force searched by any algorithm B' in $\mathbb{C}$, but *(ii)* the supervisor possesses additional information by which it can reduce the key search space to a key interval $\mathcal{K}_S \subset \mathcal{K}$ in which it is feasible for $B' \in \mathbb{C}$ to brute-force search for that key (i.e., $\mathcal{K} \gg \mathcal{K}_S$). For example, although the key k could be 192 bits, the supervisor might recall the first 120 bits of k and would like to privately brute-force search for the remaining 72 bits, without the helpers learning any meaningful information about *all the bits of k*. Note that we do not consider any specific distribution of the bits of k that are known to the supervisor. That is, the supervisor could recall consecutive bits (e.g., a string) or even random bits of k. This setting also mimics well that of password-based schemes (e.g., for disk encryption purposes) where the encryption key is derived from a password by means of a one-way pseudorandom key derivation function. Here, although the output space of the key derivation function is large, additional knowledge about the password can significantly reduce the key space. We further assume that there is a confidential channel between the helpers and the supervisor (e.g., the helpers could establish a session key with the supervisor using his public key).

Our N-Private Solver.

The main intuition behind an N-private solver is that any adversary M in $\mathbb{C}$ that controls up to N helper nodes cannot acquire information about the "additional" knowledge that is in the possession of the supervisor (i.e., the set $\mathcal{K}_S$). Given that $|\mathcal{K}|$ is too large to be brute-force searched by M, this suggests that M cannot acquire, with high probability, *any meaningful information* (e.g., range, bound) about the correct decryption key k.

Our N-private solver unfolds as follows. Let (A, B) be a solver for $\mathbb{K}_k^{CBC}$ and let ϵ and r be security parameters. A and B interact as follows:

- Similar to the privacy-preserving solver for $\mathbb{K}_p^{CBC}$, A computes $c_0 \oplus p_1$.

- Given ϵ, A also computes $R = \lfloor \frac{2}{\epsilon} \rfloor$. Let $\mathcal{K}$ denote the key space and $\mathcal{K}_S$ refer to the reduced key space in possession of A. A divides $\mathcal{K}_S$ into n subspaces $\{\mathcal{K}_S^1..\mathcal{K}_S^n\}$ (such that $R \cdot n > \frac{|\mathcal{K}|}{|\mathcal{K}_S|}$). Furthermore, A randomly chooses $(R-1) \cdot n$ key subspaces from $\{\mathcal{K}-\mathcal{K}_S\}$; these subspaces have the same size of the n intervals of $\mathcal{K}_S$. These subspaces could also be used by the supervisor to create ringer problems—and therefore to verify the integrity of the outsourced search—as described in Section 3.2. In total, A therefore constructs $H = R \cdot n$ equal-sized key subspaces (Figure 3).

- Assuming that there are $H = R \cdot n$ helpers in the network (that run algorithm B), A assigns uniformly at random one of the H key subspaces to each helper and asks the helpers to search in their assigned space for the keys that encrypt c_1 into $\{c_0 \oplus p_1\}_{(\log|\mathcal{P}|-r)}$ (i.e., into the first $(\log|\mathcal{P}|-r)$ bits of $c_0 \oplus p_1$)[12].

- B returns all the keys $\bar{k}$ that satisfy the search conditions.

- A aggregates the output of B and extracts the correct decryption key k.

Security Analysis.

We show that the solver that we presented in the previous paragraph is a $0.59\sqrt{\frac{H|\mathcal{K}|}{\epsilon^3|\mathcal{K}_S|}}$-private solver for $\mathbb{K}_k^{CBC}$. For that purpose, we have to show that the probability that an attacker—that can compromise a maximum of $0.59\sqrt{\frac{H|\mathcal{K}|}{\epsilon^3|\mathcal{K}_S|}}$ nodes—acquires any information about the key is negligible, even if the key is within the allocated search space of the compromised nodes.

We first note that since the communication between the helpers and the supervisor is performed over a confidential channel, then the only information available to M consists of the public protocol parameters and the inputs/outputs that are revealed to the N helper nodes that are compromised by M (Figure 4). This is an essential requirement to ensure the security of our scheme. In fact, if the communication

[12] The inputs and outputs of the private solver depend on the encryption mode in use. For instance, in the CTR mode encryption and similar to its plaintext-hiding privacy-preserving solver, helpers search in the assigned space for the keys $\bar{k}$ such that $G(\bar{k}) = \{c_0 \oplus p_0\}_{(\log|\mathcal{P}|-r)}$, where $G(.)$ is $\mathbb{C}$-secure pseudorandom generator.

between A and B is not confidential, then M can monitor the interface of A and acquire a set of candidate keys from which it is feasible for M to acquire the key k.

Our analysis relies on a recent uniformity hypothesis testing result [41]: Paninski *et al.* lower-bounded the number of independent samples N that are required to decide that a discrete distribution d—supported on m points—is ϵ-distant from a uniform distribution. More formally, Paninski *et al.* proved the following theorem.

Theorem 4.1. *Let $H_0 : d_i \equiv \frac{1}{m}$ and consider the non-parametric alternative $H_A : \sum_{i=1}^{m} |d(i) - \frac{1}{m}| > \epsilon$. Furthermore, let N denote the number of independent and identically distributed samples that are required to distinguish H_0 from H_A. If $N^2\epsilon^4 < m \log 5$, then no test can reliably distinguish H_0 from H_A [41].*

In analyzing the advantage of an adversary M in acquiring information about k given our aforementioned private solver, the main observations are as follows:

- There are a total of $\frac{n|\mathcal{K}|}{|\mathcal{K}_S|}$ sets in $\mathcal{K}$ of the same size as the n sets in $\mathcal{K}_S$. Therefore, the distribution D supporting the aforementioned allocation of these key sets to nodes is given by:

$$D(i) = \begin{cases} \frac{1}{R \cdot n} & \text{if } i \in \mathcal{K}_S, \\ \frac{R-1}{R\left(\frac{n|\mathcal{K}|}{|\mathcal{K}_S|} - n\right)} & \text{otherwise} \end{cases}$$

- Given this, it follows that:

$$\sum_{i=1}^{\frac{n|\mathcal{K}|}{|\mathcal{K}_S|}} \left| D(i) - \frac{1}{m} \right| = n\left| \frac{1}{R.n} - \frac{|\mathcal{K}_S|}{n|\mathcal{K}|} \right| + \left(\frac{n|\mathcal{K}|}{|\mathcal{K}_S|} - n \right) \times \left| \frac{R-1}{R} \frac{1}{\frac{n|\mathcal{K}|}{|\mathcal{K}_S|} - n} - \frac{|\mathcal{K}_S|}{n|\mathcal{K}|} \right| = \left| \frac{|\mathcal{K}_S|}{|\mathcal{K}|} - \frac{1}{R} \right| + \left| \frac{R-1}{R} - \frac{|\mathcal{K}| - |\mathcal{K}_S|}{|\mathcal{K}|} \right|$$

Given that $|\mathcal{K}| \gg |\mathcal{K}_S|$, then:

$$\sum_{i=1}^{\frac{n|\mathcal{K}|}{|\mathcal{K}_S|}} \left| D(i) - \frac{1}{m} \right| \approx \frac{1}{R} + 1 - \frac{R-1}{R} \approx \frac{2}{R} \geq \epsilon.$$

This means that D and the uniform distribution are ϵ-distant.

Following from Theorem 4.1, if M acquires at most $N < \frac{\sqrt{n|\mathcal{K}| \log 5}}{\epsilon^2 \sqrt{\mathcal{K}_S}} \approx \frac{\sqrt{H|\mathcal{K}| \log 5}}{\epsilon^{\frac{3}{2}} \sqrt{2|\mathcal{K}_S|}} \approx 0.59\sqrt{\frac{H|\mathcal{K}|}{\epsilon^3 |\mathcal{K}_S|}}$ key interval samples chosen uniformly at random, then it cannot distinguish D from the uniform distribution supported on the key intervals of $\mathcal{K}$. This also means that M cannot acquire any information about $\mathcal{K}_S$—the reduced set of candidate keys—from D; that is, even if M compromises N helpers, it will not learn any additional information about the parts of the key that the supervisor can recall nor any $\mathbb{C}$-feasible bound on an interval where the decryption key is located.

We further note that little can be done by M to verify any guess that it makes. In fact, given that A only reveals c and the first $(\log|\mathcal{P}| - r)$ bits of $c_0 \oplus p_1$, this suggests that:

- Assuming that the encryption function in question behaves as a pseudorandom permutation, then there are $\frac{2^r \cdot |\mathcal{K}|}{|\mathcal{P}|}$ candidate solutions to this search problem. Since there are H helpers in the network, these helpers would output a set $\mathcal{S}$ comprising of $\frac{2^r \cdot R \cdot |\mathcal{K}_S|}{|\mathcal{P}|} \approx \frac{2^{r+1} \cdot |\mathcal{K}_S|}{\epsilon|\mathcal{P}|}$ candidate solutions—among which is the correct decryption key k (since it is easy to see that (A, B) solves $\mathbb{K}_p^{CBC}$)[13]. Each helper node then outputs a set comprising of $\frac{2^{r+1} \cdot |\mathcal{K}_S|}{\epsilon|\mathcal{P}|H}$ candidates. The probability that the correct decryption key is among those solution is bounded by $\frac{1}{H}$.

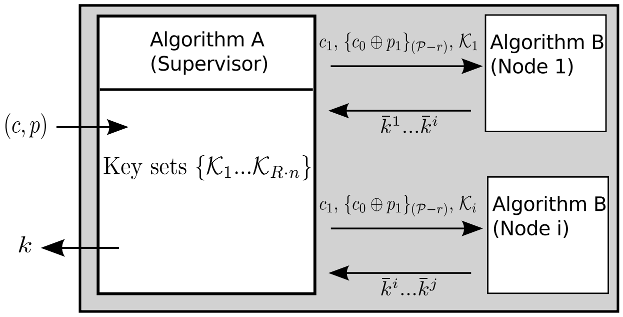
Algorithm A
(Supervisor)
Key sets $\{\mathcal{K}_1 ... \mathcal{K}_{R \cdot n}\}$
(c, p)
k
c_1, $\{c_0 \oplus p_1\}_{(\mathcal{P}-r)}$, $\mathcal{K}_1$
$\bar{k}^1 ... \bar{k}^i$
Algorithm B
(Node 1)
c_1, $\{c_0 \oplus p_1\}_{(\mathcal{P}-r)}$, $\mathcal{K}_i$
$\bar{k}^i ... \bar{k}^j$
Algorithm B
(Node i)

Figure 4: Sketch of an N-private solver for $\mathbb{K}_k^{CBC}$. Subscripts refer to the index of each block. The communication between the helpers and the supervisor is performed over a confidential channel.

- Similarly, the probability that the correct decryption key k is among the candidate keys returned by the N helpers compromised by M is bounded by $\frac{N}{H}$. These N helpers output a set comprising of $\frac{2^{r+1} N \cdot |\mathcal{K}_S|}{\epsilon|\mathcal{P}|H}$ elements. Note, here, that $2^{r+1} \cdot |\mathcal{K}_S| \gg \epsilon|\mathcal{P}|$. Otherwise, the helper node that finds a candidate solution can be certain—with considerable probability— that it holds the correct decryption key.

Recall that no algorithm $B' \in \mathbb{C}$ can acquire p_1 or p from the transcript of interaction between A and B (refer to Section 3.2). This therefore suggests that the attacker cannot verify which candidate key corresponds to the actual decryption key k.

We conclude that (A, B) is a $0.59\sqrt{\frac{H|\mathcal{K}|}{\epsilon^3 |\mathcal{K}_S|}}$-private solver for $\mathbb{K}_k^{CBC}$. Here, assuming that $\left(2^{r+1} \cdot |\mathcal{K}_S|\right) \gg \epsilon|\mathcal{P}|$, the probability that the correct decryption key can be acquired by M is bounded by $O\left(\frac{N}{H}\right)$, which is negligible if $H \cdot |\mathcal{K}_S| \gg |\mathcal{K}|$. Note that since $H = R \cdot n > \frac{|\mathcal{K}|}{|\mathcal{K}_S|}$, then $N < H$.

Example 2. *As an example, consider the case where $|\mathcal{K}| = 2^{96}$, $|\mathcal{P}| = 2^{64}$, $|\mathcal{K}_S| = 2^{72}$, $\epsilon = 1$, $r = 21$ and the network contains $H = 2^{28}$ helpers. In this case, the advantage of any attacker M in $\mathbb{C}$ in acquiring the correct decryption key is negligible if it controls a maximum of $N = 0.59\sqrt{\frac{H|\mathcal{K}|}{\epsilon^3 |\mathcal{K}_S|}} \approx 2^{25}$ helper nodes. In this example, the probability that the correct decryption key k is among the 2^2 candidate keys returned by any helper node is 2^{-28}. Similarly, the probability*

[13] Therefore, the computational complexity of algorithm A run by the supervisor is $O(|S|)$. This complexity approaches $O(H)$ if the number of candidate keys per node $\frac{2^{r+1} \cdot |\mathcal{K}_S|}{\epsilon|\mathcal{P}|H}$ is set to a constant.

that the key k is among the 2^{27} keys returned by the N nodes controlled by M correctly is approximately 2^{-4}. Nevertheless, since p is never revealed to any helper, we point out that no helper node can verify its guess. Note that, in this case, the helpers report a total of 2^{30} candidate solutions to the supervisor[14].

5. PLAN-AHEAD SCHEMES FOR KEY-HIDING SEARCHES

Plan-ahead key-hiding cryptographic schemes are encryption schemes that enable by design efficient (computational) key-hiding cryptographic searches. Although restrictive, since these schemes are "planned" at the time of encryption to exhibit such privacy-preserving properties, this notion can be useful, e.g., to users when encrypting their disks[15]; at any point in time, these users can recover their keys with the help of distributed nodes, without these nodes learning (in the computational sense) any information about the key and/or the plaintext—even if they all collude. In this section, we describe possible constructions for plan-ahead schemes that enable efficient key-hiding and plaintext-hiding searches based on *block ciphers* given *(i)* a known ciphertext/plaintext pair and *(ii)* given the sole knowledge of the ciphertext (i.e., the corresponding plaintext is not known to the supervisor).

5.1 Known Ciphertext/Plaintext Pairs

Our block-cipher plan-ahead scheme that enables the construction of efficient privacy-preserving solvers for key-hiding and plaintext-hiding search problems unfolds as follows.

We propose to encrypt each plaintext block at index $i \geq 1$ using the key $k_{“new“_i} = k \oplus F_{\mathrm{IV}}(i)$, where k is the genuine secret key, $F_{\mathrm{IV}}(i)$ is the output of the $\mathbb{C}$-secure pseudorandom function $F(.)$ for block index i given the input $\mathrm{IV} = c_0$. Note that this construction does not compromise the security of the block-cipher encryption process.

Let (A, B) be a privacy-preserving solver for the plaintext-hiding problem based on the above plan-ahead block-cipher encryption. A and B interact as described in Section 3.[16] At the end of their interaction, A verifies the validity of $\widehat{k}$ reported by the helpers and extracts the key $k = \widehat{k} \oplus F_{\mathrm{IV}}(i)$ (recall that IV is kept secret).

Note that the construction of this plan-ahead key-hiding scheme ensures forward and backward security. That is, the sole knowledge of $k_{“new“_i}$ does not leak information about $k_{“new“_j}$, k, nor about IV, $\forall j \neq i$. Therefore, no entity can make use of the obtained key to decrypt any other ciphertext that was created using the same secret key k.

This plan-ahead key-hiding encryption scheme enables, by design, the release of information that allows the brute-force search of a message key (i.e., the session key $k \oplus F_{\mathrm{IV}}(i)$), without revealing the information required to acquire the long term key k nor the plaintext p.

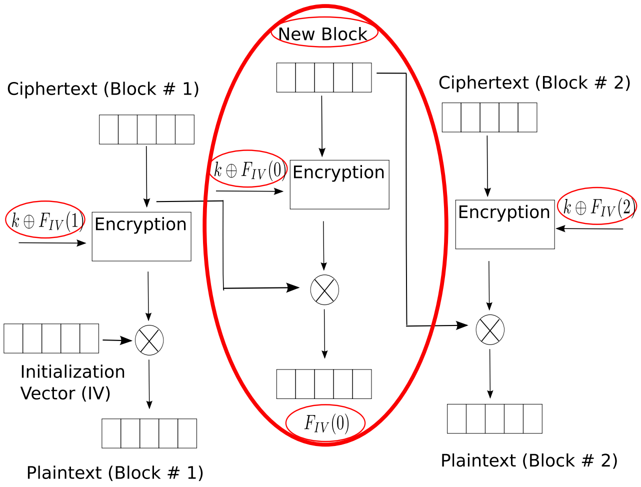

Figure 5: Example of a plan-ahead key-hiding and plaintext-hiding cryptographic scheme (with unknown plaintext) based on the CBC encryption scheme. At the time of encryption, *(i)* a new plaintext-block at index j, $p_{“new“_j}$, is inserted such that $p_{“new“_j} = F_{\mathrm{IV}}(0)$ and *(ii)* all plaintext-blocks are encrypted using the key $k_{“new“_i} = k \oplus F_{\mathrm{IV}}(i)$, where k is the genuine secret key, i is the block index and $F(.)$ is a pseudorandom function.

It can be shown that (A, B) is a privacy-preserving solver for both plaintext-hiding and key-hiding search problems based on the proposed cryptographic construction.

5.2 Known Ciphertext Only (Unknown Plaintext)

So far, the privacy-preserving (and N-private) solvers that we proposed in this paper rely on the a priori knowledge of the plaintext. Here, we propose a plan-ahead encryption scheme that enables the construction of privacy-preserving solvers for plaintext-hiding and key-hiding block-cipher search problems *without requiring the knowledge of the original plaintext.*

The main intuition behind our construction is to embed, at the time of encryption, a plaintext block that is known to algorithm A, but cannot be acquired by any algorithm B' in $\mathbb{C}$; otherwise, B' can use the plaintext to search for the corresponding key.

Our scheme is a variant of the plan-ahead key-hiding construction that we described in Section 5.1. Let c_i and p_i be the ciphertext and plaintext blocks at index $i \geq 1$, respectively. At the time of encryption, *(i)* each plaintext block at index $i \geq 1$ is encrypted using the key $k_{“new“_i} = k \oplus F_{\mathrm{IV}}(i)$ and *(ii)* a new block $p_{“new“_j} = F_{\mathrm{IV}}(0)$ is embedded within the plaintext-blocks to encrypt[17]. Note that the index value j could be pre-defined and made public; for example, in a

[14] For comparison purposes, the collective computing power harnessed in the RSA DES project [4] enabled testing of approximately 2^{38} keys per second.

[15] Given the large amount of data stored on disks, users would typically rely on symmetric encryption to encrypt their disks. This motivates the need for efficient privacy-preserving solvers for key-hiding searches based on block ciphers.

[16] The described plan-ahead block-cipher encryption does not impact how block encryption is performed. Therefore, the privacy-preserving solvers for plaintext-hiding problems based on the CTR, OFB, CBC, PCBC, CFB block cipher mode encryption (Section 3) can be used to solve plaintext-hiding problems based on this variant plan-ahead encryption scheme.

[17] Recall that $F(.)$ is a $\mathbb{C}$-secure pseudorandom function.

	Computational Complexity of algorithm A	**Communication Complexity of algorithm B**
Privacy-preserving solver for $\mathbb{K}_p$ (known plaintext)	$O(1)$	$O(\|k\|)$
PIR-based solver for $\mathbb{K}_k$ & $\mathbb{K}_p$ (known plaintext)	$O(2^{\frac{\|k\|}{2}})$	$O(\|k\|)$
N-private solver for $\mathbb{K}_k$ & $\mathbb{K}_p$ (known plaintext)	$O(H)^{(*)}$	$O(\|k\|)$
Plan-ahead solver for $\mathbb{K}_k$ & $\mathbb{K}_p$ (known or unknown plaintext)	$O(1)$	$O(\|k\|)$

Table 1: Performance comparison of privacy-preserving solvers for plaintext-hiding and key-hiding searches based on block-ciphers (in the CTR, OFB, CBC, CFB, PCBC mode encryption). $|k|$ refers to the size (in bits) of the key and H denotes the number of helper nodes in the network. $\mathbb{K}_k$ and $\mathbb{K}_p$ refer to key-hiding and plaintext-hiding search problems, respectively. $^{(*)}$ for the reasoning why, refer to Section 4.3.

plan-ahead CBC encryption scheme, the second plaintext-block can be always set to $F_{\mathrm{IV}}(0)$.[18]

Let (A, B) be a privacy-preserving solver corresponding to the plaintext-hiding search problem based on this plan-ahead encryption scheme. Here, A only reveals $p_{``new``_j}$ and $c_{``new``_j}$ to the helpers. It is easy to see that (A, B) is also a privacy-preserving solver for the variant key-hiding search problem. In particular, similar to the analysis in Section 3, it can be shown that there exist efficient privacy-preserving solvers for this plan-ahead scheme based on the CTR, OFB, CBC, PCBC, CFB block cipher mode encryption.

In Figure 5, we sketch an example of this plan-ahead encryption scheme based on the CBC encryption scheme; we refer to this encryption scheme by $\overline{CBC}$. Here, the privacy-preserving solver (A, B) for $\mathbb{K}_k^{\overline{CBC}}$ and $\mathbb{K}_p^{\overline{CBC}}$ unfolds as follows: *(i)* A sends c_{j+1} and $c_j \oplus p_{``new``_j}$ to B, *(ii)* B searches for the key $\widehat{k}$ that encrypts c_{j+1} into $c_j \oplus p_{``new``_j}$ and sends it to A and *(iii)* A extracts the key $k = \widehat{k} \oplus F_{\mathrm{IV}}(j)$. Note that this privacy-preserving solver only requires the knowledge of the ciphertext c.

6. RELATED WORK

Our work shares similarities with the notion of "deniable encryption" [17, 23, 37]. An encryption scheme is "deniable" if the ciphertext can be seen as an encryption of another plaintext, thus keeping the real plaintext private. Our work differs from deniable encryption in the fact that the supervisor does *not* possess the secret key, and wishes to retrieve it while keeping it, along with the plaintext hidden from any third-party.

Canetti *et al.* [18] describe possible constructions for non-committing encryptions. In these constructions, only a "simulator" can generate "dummy ciphertexts" that could be used as encryptions for more than one value. Artzi *et al.* [9] present a scheme that enables encrypted keyword search over a set of encrypted documents.

Gennaro *et al.* [25] introduce the notion of a "verifiable computation scheme" and present a protocol that enables privacy-preserving outsourcing of computations to untrusted workers. The main intuition behind their scheme is to combine Yao's Garbled Circuit with a fully homomorphic encryption system to ensure the privacy of the input and output of the outsourced computations.

Hohenberger *et al.* [30] describe a scheme to outsource modular exponentiations to a set of untrusted helper nodes. Besides the basic differences in the properties of the outsourced tasks—which require different solution spaces—our approach ensures the privacy of the sensitive instances and/or solutions of the outsourced search problem even in the case when the helpers are malicious and collude.

Shaneck *et al.* [44] propose a scheme that enables privacy-preserving nearest neighbor search in the context of data mining applications. Other proposals include protocols for oblivious transfer [15, 42], privacy-preserving genomic computation [47]; proposals for other privacy-preserving or oblivious problems can be found in [8, 13, 14, 19, 31, 32, 38, 39, 45].

Other contributions in the literature addressed the integrity of outsourced computations. Jakobsson *et al.* [33] propose a scheme that enables secure outsourcing of a batch of signatures to a remote server. Golle *et al.* [28] propose to secure a specific class of parallel problems that are run on remote servers: inverse one-way functions (IOWF), where the helper nodes are required to compute the pre-images of several one-way functions. These correspond to the pre-computed images of randomly chosen elements, *ringers*, in each task. This solution has been extended to secure sequential problems [29, 35, 46].

Similarly, Du *et al.* [22] discuss a scheme that uses sampling techniques along with a Merkle-tree based commitment technique to secure non-sequential distributed problems in grid computing.

7. CONCLUDING REMARKS

In this work, we considered the problem of enabling privacy-preserving distributed searches of cryptographic keys without leaking sensitive information in the search process.

For this purpose, we introduced the notion of a "privacy-preserving solver" as a building block for solving "plaintext-hiding" and "key-hiding" cryptographic search problems. In these problems, a supervisor that holds a ciphertext (and possibly the corresponding plaintext) asks remote helper nodes to assist him in the search for the corresponding decryption key without the nodes learning any information about the plaintext or the secret key.

Our results show that, for most public-key and private-key encryption algorithms, privacy-preserving solvers for plaintext-hiding searches can be efficiently constructed. By lever-

[18] If $|F(.)| \leq |\mathcal{P}|$, then $p_{``new``_j}$ is padded with additional zeros to match $|\mathcal{P}|$; otherwise, the last few bits of $F(.)$ can be truncated.

aging on these solvers, we also showed how to construct privacy-preserving solvers for key-hiding searches. While the latter solvers are typically not efficient in private-key encryption, we showed that efficient private solvers for key-hiding searches based on block-ciphers can be constructed in the practical case where the adversary does not control the entire network. In the special case when (computational) privacy-preserving properties are required to enable both key-hiding and plaintext-hiding searches, plan-ahead encryption schemes can enable the efficient construction of the corresponding privacy-preserving solvers (even without the knowledge of the plaintext). Our findings are summarized in Table 1.

Given that our proposed solvers can be directly integrated in existing distributed computing platforms, our work highlights the need to consider these privacy-preserving properties and their implications in the design and deployment of cryptographic searches. In this respect, we argue that the property of enabling efficient privacy-preserving solvers should be considered as an additional criterion for choosing which cryptographic algorithm to use; for example, while this property is present in most block-cipher mode encryption, we believe that it is not a feature of the (insecure) ECB block cipher mode. We plan to investigate a potential relationship between the security of an encryption algorithm and its "privacy-preserving" properties; more specifically, we are interested in investigating whether an encryption algorithm that supports classes of privacy-preserving solvers beyond what is presented in this work should be insecure (with respect to the ciphertext indistinguishability property).

8. ACKNOWLEDGMENTS

The authors would like to acknowledge the anonymous reviewers for their helpful feedback and comments. The work presented in this paper was supported (in part) by the Swiss National Science Foundation under Grant 200021-127294.

9. REFERENCES

[1] Distributed.Net, Available from: `http://distributed.net/`.

[2] Electronic Frontier Foundation, Available from: `http://www.eff.org`.

[3] M4 Project, Available from: `http://www.bytereef.org/m4_project.html`.

[4] RSA's DES Challenge III, Available from: `http://www.rsa.com/rsalabs/node.asp?id=2108`.

[5] DES Cracker, Available from: `http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html`.

[6] NIST Report on Cryptographic Key Length and CryptoPeriod, Available from: `http://www.keylength.com/en/4/`.

[7] WPA Cracker, Available from: `http://www.wpacracker.com/index.html`.

[8] Applebaum, B., Ringberg, H., Freedman, M. J., Caesar, M., and Rexford, J. Collaborative, Privacy-Preserving Data Aggregation at Scale. In *Proceedings of PETS* (2010).

[9] Artzi, S., Kiezun, A., Newport, C., and Schultz, D. Encrypted Keyword Search in a Distributed Storage System. MIT CSAIL Tech Report MIT-CSAIL-TR-2006-010, 2006.

[10] Biham, E., and Shamir, A. Differential Cryptanalysis of the Full 16-Round DES. In *Proceedings of CRYPTO* (1992).

[11] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., and Shamir, A. Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds. In *Proceedings of EUROCRYPT* (2010).

[12] Boneh, D. Twenty Years of Attacks on the RSA Cryptosystem. In *Notices of the American Mathematical Society (AMS)* (1999).

[13] Brickell, J., Porter, D., Shmatikov, V., and Witchel, E. Privacy-Preserving Remote Diagnostics. In *Proceedings of ACM CCS* (2007).

[14] Brickell, J., and Shmatikov, V. Privacy-Preserving Classifier Learning. In *Financial Crypto* (2009).

[15] Cachin, C. On the Foundations of Oblivious Transfer. In *Proceedings of EUROCRYPT* (1998).

[16] Cachin, C., Micali, S., and Stadler, M. Computationally Private Information Retrieval with Polylogarithmic Communication. In *Proceedings of EUROCRYPT* (1999).

[17] Canetti, R., Dwork, C., Naor, M., and Ostrovsky, R. Deniable Encryption. In *Proceedings of CRYPTO* (1997).

[18] Canetti, R., Feige, U., Goldreich, O., and Naor, M. Adaptively Secure Multi-party Computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC)* (1996).

[19] Chaum, D. Blind Signatures for Untraceable Payments. In *Proceedings of CRYPTO* (1982).

[20] Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. Private Information Retrieval. In *Journal of the ACM* (1998).

[21] Coppersmith, D., Franklin, M., Patarin, J., and Reiter, M. Low-exponent RSA with related messages. In *Proceedings of EUROCRYPT* (1996).

[22] Du, W., Jia, J., Mangal, M., and Murugesan, M. Uncheatable Grid Computing. In *Proceedings of ICDCS* (2004).

[23] Duermuth, M., and Freeman, D. M. Deniable Encryption with Negligible Detection Probability: An Interactive Construction. To Appear in EUROCRYPT, 2011. Available from `http://eprint.iacr.org/2011/066.pdf`.

[24] ElGamal, T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory* (1985).

[25] Gennaro, R., Gentry, C., and Parno, B. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Proceedings of CRYPTO* (2010).

[26] Gentry, C., and Ramzan, Z. Single-Database Private Information Retrieval with Constant Communication Rate. In *Automata, Languages and Programming* (2005).

[27] Goldwasser, S., Micali, S., and Rackoff, C. The Knowledge Complexity of Interactive Proof-Systems.

In *Proceedings of 17th Symposium on the Theory of Computation* (1985).

[28] Golle, P., and Mironov, I. Uncheatable Distributed Computations. In *Proceedings of the RSA Conference* (2001).

[29] Goodrich, M. T. Pipelined Algorithms to Detect Cheating in Long-Term Grid Computations. In *Theoretical Computer Science, LNCS, Springer* (2008).

[30] Hohenberger, S., and Lysyanskaya, A. How To Securely Outsource Cryptographic Computations. In *Theory of Cryptography Conference, LNCS, Springer* (2005).

[31] Huang, Q., Wang, H. J., and Borisov, N. Privacy-Preserving Friends Troubleshooting Network. In *Proceedings of NDSS* (2005).

[32] Huanga, Y., Malka, L., Evans, D., and Katz, J. Efficient Privacy-Preserving Biometric Identification. In *Proceedings of NDSS* (2011).

[33] Jakobsson, M., and Wetzel, S. Secure Server-Aided Signature Generation. In *Proceedings of the 4th International Workshop on Public Key Cryptography (PKC), LNCS, Springer* (2001).

[34] Jochemsz, E., and May, A. A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$. In *Proceedings of CRYPTO* (2007).

[35] Karame, G., Strasser, M., and Capkun, S. Secure Remote Execution of Sequential Computations. In *Proceedings of ICICS* (2009).

[36] Katzenbeisser, S. Recent Advances in RSA Cryptography. Volume 3 of Advances in Information Security, Springer, 2001.

[37] Klonowski, M., Kubiak, P., and Kutylowski, M. Practical Deniable Encryption. In *Proceedings of SOFSEM: Theory and Practice of Computer Science* (2008).

[38] Lincoln, P., Porras, P., and Shmatikov, V. Privacy-Preserving Sharing and Correlation of Security Alerts. In *Proceedings of USENIX Security* (2004).

[39] Lindqvist, J., Aura, T., Danezis, G., Koponen, T., Myllyniemi, A., Maki, J., and Roe, M. Privacy-Preserving 802.11 Access-Point Discovery. In *Proceedings of ACM WiSec* (2009).

[40] Mantin, I., and Shamir, A. A Practical Attack on Broadcast RC4. In *Proceedings of FSE* (2001).

[41] Paninski, L. A Coincidence-based Test for Uniformity given very Sparsely-Sampled Discrete Data. In *IEEE Transactions on Information Theory* (2008).

[42] Rabin, M. O. How to Exchange Secrets by Oblivious Transfers. Technical Report, Harvard University, 1981.

[43] Schnorr, C. Efficient Identification and Signatures for Smart Cards. In *Proceedings of CRYPTO* (1990).

[44] Shaneck, M., Kim, Y., and Kumar, V. Privacy Preserving Nearest Neighbor Search. In *Proceedings of the International Conference on Data Mining* (2006).

[45] Sion, R. Towards Secure Data Outsourcing. M. Gertz and S. Jajodia (editors), Springer Verlag, ISBN: 978-0-387-48532-4, 2008.

[46] Szajda, D., Lawson, B., and Owen, J. Hardening Functions for Large Scale Distributed Computations. In *Proceedings of the IEEE Symposium on Security and Privacy* (2003).

[47] Wang, R., Wang, X., Li, Z., Tang, H., Reiter, M., and Dong, Z. Privacy-Preserving Genomic Computation Through Program Specialization. In *Proceedings of ACM CCS* (2009).

APPENDIX

Appendix A: Proof of Lemma 1

In this appendix, we restate and prove Lemma 1.

Lemma: *If G is a $\mathbb{C}$-secure pseudorandom generator, and $M \in \{0,1\}^n$, then $C = G(s) \oplus M$, where s is chosen uniformly at random from $\{0,1\}^n$, is computationally independent of M relative to complexity class $\mathbb{C}$.*

Proof: Let G be a $\mathbb{C}$-secure pseudorandom generator. That is, G is an efficient deterministic algorithm that, when given a randomly chosen input $s \in \{0,1\}^n$, it outputs a pseudorandom string $G(s)$ of length $l(n) \geq n$ such that the advantage of any distinguisher D in $\mathbb{C}$ in distinguishing $G(s)$ from a randomly chosen $r \in \{0,1\}^n$ is negligible.

To prove that M and $G(s) \oplus M$ are computationally independent (relative to $\mathbb{C}$), according to Definition 3, we need to exhibit a pair (U', V') of statistically independent random variables such that the pairs $(M, G(s) \oplus M)$ and (U', V') are computationally indistinguishable. This is achieved by defining U' to be distributed according to the distribution of M and defining V' to be an independent uniform n-bit string. In other words, we claim that $(M, G(s) \oplus M)$ and (M, V') are computationally indistinguishable.

To prove this, note that a distinguisher D for $(M, G(s) \oplus M)$ and (M, V') can be used to distinguish $G(s)$ from a uniform random n-bit string U, simply by choosing a random variable M (according to the right distribution) and feeding the pairs $(M, G(s) \oplus M)$ and (M, U) to the distinguisher D. The advantage is the same, thus concluding the proof. □