Private Resource Pairing

Joseph A. Calandrino* and Alfred C. Weaver

University of Virginia
Charlottesville, Virginia
{jac4dt, acw}@cs.virginia.edu

Abstract. Protection of information confidentiality can result in obstruction of legitimate access to necessary resources. This paper explores the problem of pairing resource requestors and providers such that neither must sacrifice privacy. While solutions to similar problems exist, these solutions are inadequate or inefficient in the context of private resource pairing. This work explores private resource-pairing solutions under two models of participant behavior: honest-but-curious behavior and potentially malicious behavior. Without compromising security, the foundation of these solutions demonstrates significant performance benefits over a popular solution to the similar private matching problem.

1 Introduction

In privacy-critical scenarios, the need to protect information confidentiality can impede valid resource requests. Resource providers may refuse to even confirm possession of a resource to requestors that have not demonstrated a need to access the resource. Such a scenario would force requestors to first reveal their queries accompanied by justifications. As a request query alone may contain or imply confidential data, requestors need some assurance that a provider can satisfy a request before revelation of the request. If both entities refuse to compromise privacy, a reasonable request could go unfulfilled. Private resource pairing links resource providers and legitimate requestors while preserving privacy.

Several recent papers have explored the similar private matching problem, in which operators of two separate databases wish to establish common entries without revealing non-matching elements [2, 8, 11, 12]. By treating request queries as single-entry databases and forcing providers to maintain databases of resource metadata, existing solutions to the private matching problem can, with minor modification, solve the private resource-pairing problem for honest-but-curious participants. This paper presents schemes with two primary advantages over such a solution:

* This research was partially performed under an appointment to the Department of Homeland Security (DHS) Scholarship and Fellowship Program, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE.

– Efficiency: The unique constraints of private resource pairing allow the use of pre-computation and other techniques that significantly decrease the computational costs of searches over a popular private matching solution.
– Security: While a private matching solution exists that prevents participant dishonesty [12], its technique is incompatible with private resource pairing. This paper proposes several methods for thwarting dishonest behavior.

1.1 Motivating Scenarios

Under a number of circumstances, a solution to the private resource-pairing problem would allow organizations to come closer to the ideal of precisely pairing entities with needed resources. Two such scenarios arise in the medical and national intelligence domains.

Medical Scenario. Suppose that an incapacitated tourist with no identification arrives at a hospital in the United States. The safety of any treatment for the patient's condition is highly dependent upon her medical history. In addition, the patient's condition, while serious, will not dramatically deteriorate during the time a doctor would require to review the patient's record. Further, assume that some biometric or combination of biometrics could allow unique, perfectly reproducible identification of any human. Prior to administering treatment, the hospital may wish to use the patient's biometric to make an emergency request for relevant records from all health centers in the country or a particular region.

In the United States, no centralized repository exists for medical records, and security and medical data ownership issues presently preclude the use of such a repository [18]. Therefore, a searching party would need to approach numerous medical centers and inquire as to whether those centers possess records related to the patient. Given a reasonable alternative, most people would prefer not to disclose their hospital visits to unnecessary parties. To comply with federal medical privacy standards, health centers are also unlikely to disclose lists of their patients [14]. In this scenario, a system for privately pairing record requestors and possessors would be desirable to protect patient privacy. Such a system must enforce requestor need to know and prevent provider forgery of record possession.

National Intelligence Scenario. Presume that a security analyst determines that a particular landmark may be at risk. Numerous agencies may possess data related to the landmark or threat. To protect information confidentiality, agencies may have strict policies against revealing even metadata pertaining to resources they possess. For example, an agency may have records of related threats but wish to appear unaware of the threats by restricting access to both the records and data regarding the records. Similarly, the analyst may be reluctant to reveal the metadata that interests her. In this scenario, necessary privacy hampers necessary availability. A private pairing method would be desirable to link the analyst with resources essential to assess and respond to the threat.

1.2 Paper Overview

The remainder of this paper is organized as follows. Section 2 presents existing work related to private resource pairing. Section 3 sets forth security goals. Section 4 provides a system for privately pairing resource requestors and providers given that entities are honest but curious. Section 5 suggests extensions to the system to prevent malicious behavior. Section 6 evaluates the theoretical cost, applied performance, and security of the honest but curious protocol as compared to a private matching solution. Finally, section 7 presents a summary and recommendations for future work.

Throughout this paper, assume that all sets are totally ordered, are transmitted in order, and are initially ordered by element insertion time.

2 Related Work

2.1 Private Matching

In 2003, Agrawal, Evfimievski, and Srikant presented the notion of minimal information sharing across private databases [2]. Their paper establishes protocols to allow two entities maintaining separate databases to determine query results across both databases without revealing information beyond the result or requiring a trusted third party. Agrawal et al. address the intersection query problem, also known as the private matching problem, and several other query types. Assuming Alice wishes to learn the intersection between her and Bob's databases, the Agrawal, Evfimievski, and Srikant private matching solution (AgES) is:

1. Alice and Bob agree on a commutative encryption function, f, and select appropriate secret keys, $e_A, e_B \in keyF$. Note that, for commutative encryption, $f_{e_A}(f_{e_B}(x)) = f_{e_B}(f_{e_A}(x))$ given $x \in domF$ and $e_A, e_B \in keyF$.
2. Alice and Bob, using a common one-way collision resistant hash function [11] from $domD$ (the domain of potential database entries) to $domF$, hash all entries in their databases: $A_h = \{h(a)|a \in A\}$ and $B_h = \{h(b)|b \in B\}$.
3. Alice and Bob encrypt the elements in A_h and B_h, producing $A_{e_A} = \{f_{e_A}(a_h)$ $|a_h \in A_h\}$ and $B_{e_B} = \{f_{e_B}(b_h)|b_h \in B_h\}$ then reorder A_{e_h} and B_{e_h} lexicographically. Alice maintains the dataset $\{(a, f_{e_A}(h(a)))|a \in A\}$.
4. Alice and Bob exchange A_{e_A} and B_{e_B}.
5. Alice computes $B_{e_B,e_A} = \{f_{e_A}(b_{e_B})|b_{e_B} \in B_{e_B}\} = \{f_{e_A}(f_{e_B}(b_h))|b_h \in B_h\}$. Bob computes $A_{e_A,e_B} = \{f_{e_B}(a_{e_A})|a_{e_A} \in A_{e_A}\}$ $(= \{f_{e_A}(f_{e_B}(a_h))|a_h \in A_h\})$ and uses the result to create the set $\{(f_{e_A}(a_h), f_{e_A}(f_{e_B}(a_h)))|a_h \in A_h\}$.
6. Bob returns $\{(f_{e_A}(a_h), f_{e_A}(f_{e_B}(a_h))|a_h \in A_h\}$ to Alice.
7. Alice joins $\{(a, f_{e_A}(h(a)))|a \in A\}$ and $\{(f_{e_A}(a_h), f_{e_A}(f_{e_B}(a_h))|a_h \in A_h\}$ on $f_{e_A}(h(a))$ to get $\{(a, f_{e_A}(f_{e_B}(h(a))))|a \in A\}$.
8. Alice extracts all $a \in A$ such that the corresponding $f_{e_A}(f_{e_B}(a))$ matches some value in B_{e_B,e_A}. These values comprise the intersection of A and B.

Agrawal et al. demonstrate the computation and communication benefits of their protocol over a solution using circuit-based protocols.

AgES assumes semi-honest, or honest-but-curious, behavior of protocol participants. This means that entities adhere to the protocol but may analyze data to derive additional information [2, 9]. For example, neither Alice nor Bob will falsely claim element possession, but Alice may perform cryptanalysis on B_{e_B}. Entities may demonstrate semi-honesty at minimum to protect their reputations, but a lack of protection measures is often inadequate. Entities may even prefer protocols in which they cannot lie to prevent false accusations of impropriety.

Li, Tygar, and Hellerstein explore private matching solutions under semi-honest and malicious models [12]. A malicious model assumes that entities may lie or deviate arbitrarily from the protocol. To prevent bogus possession claims, Li et al. propose data ownership certificates (DOCs). DOCs are not directly applicable to private resource pairing, however. A requestor may not possess a desired resource, so the requestor may not have its metadata's DOCs. Both entities must have DOCs to verify each other's[1]. While this property is desirable for private matching, an alternate solution is necessary for private resource pairing.

Li et al. present a hash-based alternative to AgES, but this alternative fails to ensure privacy without DOCs. Assume Alice and Bob agree on a hash function and trade hashed items. From that point on, Alice can guess and check for any item she desires, regardless of whether she possesses that item, in Bob's set.

Freedman, Nissim, and Pinkas as well as Kissner and Song have proposed private matching protocols based on homomorphic encryption [8, 11]. Future research may wish to explore private resource pairing using homomorphic encryption. Note that the notion of malicious behavior in [8] and [11] differs from that of [12] and this paper, which consider *ownership* of data. For example, assume that businesses have databases of customers indexed by unique consumer identifiers. If the space of identifiers is small, a business may falsely claim all consumers as customers. Ownership mechanisms can expose or prevent dishonesty.

2.2 Additional Work

Private information retrieval (PIR) allows parties to retrieve database entries without disclosing which entries they desire [7]. Unfortunately, PIR offers no assurance that parties need the data they retrieve. Nonetheless, an efficient PIR implementation would be useful to this paper's solutions (see sections 4 and 5).

Waters, Balfanz, Durfee, and Smetters present a method to allow searches on encrypted audit logs [21]. The scheme could protect provider privacy and

[1] For the certified hash and certified AgES protocols in [12], Bob provides Alice with $\sigma = \{b||B\}_{sk}$ for each value b he possesses, where B is a unique id for Bob. Alice must possess pk to verify σ (the verification method is VERIFY$(pk, b||B, \sigma)$). If everyone knew pk, the only unknown is b. If b's domain is small and Bob is honest, Alice could mount a brute force attack, running VERIFY for all possible values of b for the σ value until VERIFY returns true. When VERIFY returns true, Alice can be confident that she found the entry in Bob's database corresponding to the σ value. Repeating this process for all of Bob's σ values yields all values in Bob's database.

enforce need to know for resource requestors. Requestors would need to reveal potentially confidential search strings to a third party, however.

Song, Wagner, and Perrig present a means of searching on encrypted data [20]. Their work allows the use of encrypted queries to search encrypted data on untrusted servers. Unfortunately, to make an encrypted query, the entity that encrypted the data (or a third party) must learn the query. For private pairing, providers would learn requestors' searches.

Zero-knowledge proofs allow a prover to demonstrate possession of a piece of information to a verifier without revealing the information. For example, a prover could demonstrate possession of Alice's unique identity-confirming key without revealing the key itself [17]. Unfortunately, a verifier must know the precise information for which a prover will seek to demonstrate possession, even if the information itself can remain private. For example, a verifier must know that the prover is demonstrating possession of Alice's key. Thus, such proofs would require one party to publicly reveal its requests or possessions.

3 Security Criteria

Li et al. [12] offer three goals for assessing the security of private matching solutions under semi-honest and malicious scenarios. This paper adopts two of the goals (the third is not applicable): (1) the protocol leaks no information beyond input size and (2) participants cannot lie regarding element possession. If parties cannot lie regarding possession, AgES leaks no information even in a malicious scenario [12]. Any protocol trivially prevents lying in a semi-honest scenario. Note that Li et al. do not address lying via omission. Allowance of nondisclosure may be desirable, so this paper presents a compromise: individuals can check that their resources are available (see section 5.2). With several exceptions (see section 4.2), [12] and this paper allow collusion.

Devious parties may mount other attacks, such as denial-of-service attacks against protocol participants. While more efficient solutions may be more resistant to some attacks, this paper does not explicitly consider these threats.

4 Semi-Honest Case Solution

This paper first presents a protocol for private resource pairing under a semi-honest behavior model. Section 5 presents extensions to the semi-honest protocol to allow enforcement of need to know and proof of resource possession.

4.1 Basic Scheme

A one-time setup process is necessary for participants in this private resource pairing protocol. Resource requestors and providers, which may be overlapping sets, agree on a common commutative encryption scheme and hash function. Providers choose random encryption keys and hash then encrypt metadata pertaining to their resources. Finally, providers publish the encryptions to potential requestors directly or to host servers. By maintaining constant keys and publishing encryptions a single time, providers can efficiently handle searches later.

When a requestor wishes to search for and acquire resources tagged with a given piece of metadata, it chooses a random encryption/decryption key pair and hashes then encrypts the metadata. The requestor gives the ciphertext to the provider, who encrypts the ciphertext again using its key and returns the result. The requestor decrypts the ciphertext and matches the result to provider-published records. If the requestor finds a match, it approaches the provider and requests resources related to the metadata. By decrypting a single item of metadata rather than re-encrypting every published piece of metadata, requestors decrease their computation. A more rigorous explanation follows shortly.

4.2 Assumptions

This protocol makes several assumptions. First, a requestor's identity alone must imply nothing confidential to providers or servers. Second, providers must publish encrypted metadata all at once (i.e., provider metadata must not change frequently), or others must be unable to draw undesirable conclusions from metadata publication order, modification, or removal. If providers use host servers, requestors must download all data from a given server. Otherwise, a server could infer whether and on what encrypted value a search is satisfied even if the underlying metadata is unknown. Private information retrieval may be unreasonable: even a search over sorted data will require $\log n$ values. Also, servers must be unable to collude to determine which servers a requestor checks. If servers colluded, they could identify the provider that satisfied a request or infer that a request went unsatisfied. Finally, this paper assumes that metadata is not fuzzy.

4.3 Detailed Process

This paper's protocol for private resource pairing under the semi-honest model (henceforth shPRP) requires separate setup and search processes.

Setup. The setup process for a resource provider, P, with resource metadata $M_P \subseteq M$, where M is the set of all possible metadata, is:

1. P, all other providers, and all potential requestors agree on a commutative encryption function, f, and a common one-way collision resistant hash function, h, that maps from $domM$ to $domF$.
2. P selects a random encryption key, e_P, such that $e_P \in keyF$.
3. P computes hashes of its resource metadata: $P_h = \{h(m_P) | m_P \in M_P\}$.
4. P encrypts the elements in P_h, producing $P_{e_P} = \{f_{e_P}(p_h) | p_h \in P_h\}$.
5. P reorders P_{e_P} lexicographically if others could infer private information from M_P's order.
6. P publishes P_{e_P} to potential requestors, host servers, or both.

If an escrow service is desirable, P may provide e_P, the related decryption key, or both to the service. If P publishes metadata to host servers, P must choose a signature scheme and accompany each published item with a signature.

Search and Acquisition The following process allows a resource requestor, R, to obtain access to P's resources with metadata m:

1. R generates a random encryption key, $e_R \in keyF$, and the corresponding decryption key, d_R.
   - R must generate a new random key pair each time it enters the search process. If R reuses a key, providers could determine whether R previously sought the same value, even if they cannot identity the value.
   - If R is also a provider, R must not use its provider key. Otherwise, R's published data would reveal whether R already possesses resources with the metadata it seeks.
2. R computes the hash of m: $m_h = h(m)$.
3. R encrypts m_h: $m_{e_R} = f_{e_R}(m_h)$.
4. R presents m_{e_R} to P.
5. P encrypts m_{e_R}: $m_{e_R,e_P} = f_{e_P}(m_{e_R}) = f_{e_P}(f_{e_R}(m_h)) = f_{e_R}(f_{e_P}(m_h))$.
6. P returns m_{e_R,e_P} to R.
7. R decrypts m_{e_R,e_P}: $m_{e_P} = f_{d_R}(m_{e_R,e_P}) = f_{e_P}(m_h)$.
8. If P hosts its data on a server, R downloads P_{e_P}, the accompanying signatures, and any items necessary to verify P's signatures (public key, etc.).
9. R searches P_{e_P} for a match to m_{e_P}.
   - If R finds a match and P_{e_P} is from a server, R may verify the corresponding signature.
   - If R finds no match and P_{e_P} is from a server, R may verify signatures to ensure that the server did not remove data.
10. If R finds a match, R asks P for resources with metadata m.

5 Malicious Case Extensions

As section 6.3 argues, shPRP does not leak information even under a malicious model. Therefore, malicious case extensions must protect against two forms of potential participant dishonesty without leaking information. First, dishonest requestors could request either metadata searches for or direct access to resources for which they have no valid need. Providers can eliminate this issue by forcing requestors to prove their need to search for metadata and access resources. Second, dishonest providers could falsely claim possession of resources to coax requestors to reveal secret search metadata. By forcing providers to prove possession of resources related to metadata, the protocol can prevent this issue. This section considers a number of possible scenarios and, for completeness, offers solutions under each scenario. Under several scenarios, the solution uses a trusted external party, which future work may be able to eliminate.

Note that the modified protocol retains all assumptions of section 4.2.

5.1 Proving Need to Know

To prevent superfluous searches and resource accesses, resource providers must have the ability to verify the legitimacy of requests. To demonstrate the need to perform a search or to access a given resource, requestors present tickets to potential providers in steps four and ten of the shPRP search and acquisition process (see section 4.3). In step four, the ticket only verifies the right to search for the encrypted metadata, m_{e_R}; it does not reveal the metadata. In step ten, the ticket contains plaintext metadata, since the provider cannot confirm that m_{e_R} represents m. Note that, to generate tickets containing m_{e_R} and m, the ticket supplier must receive both items and verify that m_{e_R} represents m.

The process by which a requestor, R, may acquire tickets from a supplier, S, is as follows (the order of steps two and three is arbitrary):

1. R presents m and m_{e_R} to S.
2. S verifies that m_{e_R} represents m:
   (a) S generates a random encryption key, $e_S \in keyF$ for the common requestor/provider commutative encryption function.
   (b) Using the common hash function, S computes the hash of m: $m_h = h(m)$.
   (c) S encrypts m_h: $m_{e_S} = f_{e_S}(m_h)$.
   (d) S presents m_{e_S} to R.
   (e) R encrypts m_{e_S}: $m_{e_S,e_R} = f_{e_R}(m_{e_S}) = f_{e_R}(f_{e_S}(m_h)) = f_{e_S}(f_{e_R}(m_h))$.
   (f) R returns m_{e_S,e_R} to S.
   (g) S encrypts m_{e_R}: $m_{e_R,e_S} = f_{e_S}(m_{e_R})$ ($= f_{e_S}(f_{e_R}(m_h))$ if m_{e_R} is valid).
   (h) S checks that m_{e_S,e_R} matches m_{e_R,e_S}.
3. S verifies R's right to search for and acquire resources with metadata m (implementation specific verification process).
4. S returns tickets for m and m_{e_R}.

Tickets can be universal or restricted to a subset of potential providers if circumstances warrant only a limited search. A network of trust must connect ticket suppliers so providers can confirm the validity of any ticket. Various models exist for establishing trust, such as direct and distributed trust models [13]. This choice is implementation-specific; the use of any model is acceptable.

Two ticket supplier models exist: internal and external. Both models assume that ticket suppliers cannot initiate searches and will not collude with malicious requestors to allow illicit access to data or resources.

An internal supplier model assumes that potential requestors are part of larger organizations and that they may reveal searches to ticket-granting parties in their organizations. The ticket-granting party verifies that present conditions warrant a search. In the medical scenario, a set of trained hospital administrators could be on-call for search verification. When a doctor explains the situation, the verifier can determine, based on established standards, whether the situation warrants a search. If the verifier concludes that it does, she can provide the doctor with appropriately constrained tickets. This solution presumes the existence of robust audit mechanisms and severe penalties to deter and detect collusion.

In the event that no impartial party exists inside a requestor's organization, requestors and providers could form agreements, contractual or otherwise, with trusted external parties to verify the need to search. In this case, requestors must also trust the verification party with their search metadata. External verification is appropriate and perhaps necessary for cases such as business agreements in which parties agree to limited, circumstance-dependent resource sharing. Members of either business may possess bias in interpretation of the agreement, creating the possible need for an impartial arbitrator.

5.2 Proving Resource Possession

As with proving need to know, this section presents two models for proving resource possession. In one model, metadata implies an obvious owner of all associated resources. For example, a patient with a unique biometric could have legal control over medical records tied to her biometric [14], making her the effective owner of the records. Under the second model, metadata either does not imply an owner or implies numerous owners. For example, "explosives" may be applicable to many intelligence resources, but the word does not imply an owner of those resources. A solution under the second model is also applicable to the first, since entities can ignore implied ownership. A solution for the first scenario is preferable when possible, however, as it allows owners to better control their resources. In both cases, solutions rely on identity-based signatures.

Identity-Based Signatures. Identity-based cryptosystems and signature schemes, first proposed by Shamir, allow the use of one's identity as its public key [19]. For example, Alice may sign her messages using a private key associated with her unique identity ("alice@petworkshop.org"). To verify her signature, Bob can simply pass the message, the signature, a master public key, public parameters, and "alice@petworkshop.org" to a verification method. Bob does not need to acquire Alice's public key to verify her signatures. Alice needs to obtain her private key from a private key generator, however, unless she possesses the system's master secret, which allows the generation of private keys for all identities. Shamir presented the first identity-based signature (IBS) scheme in [19].

Key Privacy, IBC Privacy, and IBS Privacy. Bellare, Boldyreva, Desai, and Pointcheval [3] first formalized the property of key privacy in public-key cryptosystems. Given this property, an adversary that possesses a piece of ciphertext cannot gain a non-negligible advantage in determining which public key out of a given set produced the ciphertext. For example, RSA lacks key privacy because an adversary can gain an advantage based on the public modulus [3].

Calandrino and Weaver [6] extend the notion of key privacy to multiple identity-based cryptosystem instantiations. If a cryptosystem possesses IBC privacy, an adversary can gain no more than a negligible advantage in determining which instantiation produced a given piece of ciphertext. Instantiations may share common parameters if such a choice does not undermine security [6].

When metadata implies an owner, the possession scheme's security relies on IBS systems with a novel property called IBS privacy. Suppose that multiple instantiations of an IBS scheme exist. Instantiations may share some parameters but have unique master secrets, meaning that each instantiation produces a unique mapping between identities and private keys. Assume that an adversary chooses an identity, and an arbitrary instantiation produces the identity's signature of a nonce. The adversary receives the signature but not the nonce. If, for some parameters, no adversary can reliably determine the instantiation that produced the signature, the scheme provides IBS privacy under those parameters. Appendix A offers a more formal description.

Metadata Implies an Owner. Assume that metadata implies an owner of associated resources. To allow proof of resource possession, some setup is mandatory. Owners must agree on an IBS scheme and parameters necessary for IBS privacy. Each owner generates a unique instantiation of the scheme with the common parameters. Owners publish parameters needed to verify their signatures. Either requestors and providers or public repositories must maintain lists of owners' public parameters. If a repository maintains the data, private information retrieval or total repository downloads must be reasonable so repository operators cannot infer which owner's resources a requestor seeks. Given a large number of non-colluding servers, PIR may be feasible: requestors will seek a small amount of data at a predetermined index, the owner's identity.

To prove possession, additional steps are needed between steps four and five of the shPRP setup process (see section 4.3). For each metadata item $m_P \in M_P$:

1. P determines the owner, O, that m_P implies.
2. P presents m_P and the corresponding value $p_{e_P} \in P_{e_P}$ to O.
3. O verifies that p_{e_P} represents m_P:
   (a) O generates a random encryption key, $e_O \in keyF$ for the common requestor/provider commutative encryption function.
   (b) Using the common function, O computes the hash of m_P: $m_h = h(m_P)$.
   (c) O encrypts m_h: $m_{e_O} = f_{e_O}(m_h)$.
   (d) O presents m_{e_O} to P.
   (e) P encrypts m_{e_O}: $m_{e_O,e_P} = f_{e_P}(m_{e_O}) = f_{e_P}(f_{e_O}(m_h)) = f_{e_O}(f_{e_P}(m_h))$.
   (f) P returns m_{e_O,e_P} to O.
   (g) O encrypts p_{e_P}: $p_{e_P,e_O} = f_{e_O}(p_{e_P})$ $(= f_{e_O}(f_{e_P}(m_h))$ if p_{e_P} is valid).
   (h) O checks that p_{e_P,e_O} matches m_{e_O,e_P}.
4. O verifies that P possesses resources related to metadata m_P (implementation specific verification process).
5. O signs p_{e_P} using its IBS scheme instantiation and the private key associated with P's identity.
6. O returns the signature of p_{e_P} to P.
7. P downloads the public parameters for O's IBS scheme instantiation.
8. P verifies the signature of p_{e_P} using P's identity as the public key.

The order of steps three and four is arbitrary.

Signing with the private key associated with the provider's identity prevents two providers from using the same encryption keys and sharing signed values. Because values in P_{e_P} are polynomial-time indistinguishable from random values and owners use IBS private signature schemes, an adversary will have at most a negligible advantage in determining the owner behind any given signature.

Following acquisition of signatures, P can reorder them lexicographically and publish them. If P reordered by the original encryptions, adversaries could estimate the pre-signed data ranges and attempt to infer the signing instantiations.

If owners can privately retrieve server data, they can verify at any time that servers host their data to detect malicious data removal.

Because only an owner possesses its master secret, only it can produce its private keys and generate its signatures. Owners can delegate signing responsibilities to a trusted party and can provide master secrets to an escrow system. If one resource owner's master secret is compromised, only that owner's data is compromised. Generating a new master secret and replacing associated published signatures would be straightforward, but this procedure could be problematic if others could infer confidential information from updates. If an owner updates its parameters at nearly the same time a provider updates its published data, an adversary can infer that the provider's published metadata related to the owner's resources. This paper leaves resolution of update issues to future work.

With two exceptions, the search process is the same for requestors as under shPRP. First, a requestor must obtain the owner's public parameters. Second, using the provider's identity as a public key, requestors must attempt to verify published values as the signature of m_{e_P} in step nine of the shPRP search process. If a value verifies, the provider possesses a desired resource.

In the medical scenario, patients could serve as owners of their medical records for the purpose of proving resource possession. When a patient receives medical care, she could provide her unique identifier to the medical center and authorize a delegated service to sign the encrypted hash of her identifier. If a hospital needs to retrieve the patient's records, it could use her identifier to retrieve the public parameters of her signature scheme instantiation and verify published signatures. Because medical centers would retain possession of records, such a system deliberately sidesteps disagreements over medical data ownership [18].

Metadata Does Not Imply an Owner. Assume that metadata does not imply a single owner of associated resources. Thus, no single party has a legitimate right to confirm or deny possession of resources associated with metadata. For requestors to accept possession claims, a trusted third party, centralized or distributed, seems necessary to validate provider possession based on established rules. Requestors can later verify possession without the third party, preventing the third party from collecting request data. The third party acts as a universal resource owner and maintains an identity-based signature system.

In this case, the publication process is the same as when metadata implies an owner, except the owner is always the trusted third party. The search and acquisition process is also the same, but requestors can store the single owner's parameters instead of retrieving parameters during each search. This scheme suffers from an issue common to identity-based cryptography: the key revocation problem. If any private key is compromised, the universal owner has two options:

– Publish a potentially huge exception list. In this case, the third party must maintain backup system(s) for the exceptions. Requestors would need to either store exception lists or have the ability to privately check the list.
– Change the master secret. This impractical option would entail reproducing all signatures. Gradual migration to a new master secret may be more reasonable. For example, if the key for "provider" is compromised, the party could immediately migrate all 'p' keys and gradually migrate other keys.

Fortunately, because the third party need not reveal or store private keys, private keys are nearly as difficult to compromise as the master secret.

6 Evaluation of shPRP

The AgES protocol offers the closest match to shPRP, making it the most logical comparison for theoretical cost, actual performance, and security. In a private resource-pairing scenario, AgES treats requestors as operators of one-entry databases containing the desired metadata. To fairly compare the protocols, several assumptions are necessary:

– Providers and requestors have settled on commutative encryption and hash functions prior to entering the protocol.
– Even if shPRP uses host servers, it does not create signatures. Signature costs would be dependent on implementation decisions.
– Once complete, AgES performs step ten of the shPRP search process.
– Providers publish lexicographically ordered encryptions.

shPRP has an inherent advantage over AgES because shPRP is a custom private resource-pairing solution. For example, the requirements of private matching prevent AgES's use of pre-computation. Nevertheless, as a leading private matching solution, AgES provides the most appropriate comparison.

6.1 Theoretical Costs

Assume that C_{g_e}, C_{g_d}, C_e, and C_d are the costs of generating public keys, generating private keys, encrypting, and decrypting for the chosen encryption scheme. C_h is the cost of hash computation with the chosen hash function. m and c are the metadata and metadata ciphertext lengths. A provider has p metadata items.

Under AgES, no setup procedure is necessary. For the search and acquisition process, the total computational cost is $2C_{g_e} + (C_h + 2C_e)(p + 1) + p \log p + p$, while the communication cost is $(p + 2)c + m$. Note that, with AgES, requestors and providers generate new private keys each time they enter the search and acquisition process. The setup process for shPRP has a total computational cost of $C_{g_e} + (C_h + C_e)p + p \log p$, while the communication cost is pc. The computational cost for the search and acquisition process is $C_{g_e} + C_{g_d} + C_h + 2C_e + C_d + \log p$. If requestors download metadata from a server, the communication cost is $(p + 2)c + m$. Otherwise, the communication cost is $2c + m$. Table 1 and Table 2 summarize these results in greater detail.

Table 1. Computational cost comparison. See section 6.1 for variable definitions.

Setup		
	AgES	shPRP
Provider	-	$C_{g_e} + (C_h + C_e)p + p \log p$
Requestor	-	-
Total	-	$C_{g_e} + (C_h + C_e)p + p \log p$

Search and Acquisition		
	AgES	shPRP
Provider	$C_{g_e} + C_h p + C_e(p+1) + p \log p$	C_e
Requestor	$C_{g_e} + C_h + C_e(p+1) + p$	$C_{g_e} + C_{g_d} + C_h + C_e + C_d + \log p$
Total	$2C_{g_e} + (C_h + 2C_e)(p+1) + p \log p + p$	$C_{g_e} + C_{g_d} + C_h + 2C_e + C_d + \log p$

Table 2. Communication cost comparison. See section 6.1 for variable definitions.

	AgES	shPRP	
		w/ Host Server	w/o Host Server
Setup	-	pc	pc
Search and Acquisition	$(p+2)c+m$	$(p+2)c+m$	$2c+m$

After the initial setup, shPRP significantly lightens the computational costs for requestors and providers while producing equivalent or better communication costs. Provider computational cost during the search and acquisition process is critical, as a reduction in cost allows providers to handle more requests per given time. The importance of reducing this cost underscores the value of performing pre-computation during the setup process. These theoretical results also suggest an improvement to the AgES private matching protocol. If, in the protocol of section 2.1, Alice has a smaller dataset than Bob and $C_e \approx C_d$, she should not encrypt Bob's set in step five. She should instead decrypt her $f_{e_A}(f_{e_B}(h(a)))$ values between steps seven and eight and match those against Bob's set.

6.2 Actual Performance

A series of tests compared the performance of AgES to shPRP. Java-based implementations of shPRP and portions of the AgES protocol allowed direct comparisons. SHA-1 and Pohlig-Hellman [15] with a common modulus served as the hash function and commutative encryption scheme respectively. The sorting algorithm was a modified mergesort with guaranteed $n \log n$ performance [10]. For the tests, providers maintained 10,000 metadata items. To achieve a fair comparison, the AgES implementation contained a straightforward optimization: requestors encrypt provider-published data line-by-line (instead of all at once), checking each result against the re-encryption of the desired metadata. When a match exists, the optimization reduces requestor encryptions by 50% on average. Tests ran on a 3.2 GHz Pentium 4 with 512 MB of RAM. Table 3 shows the results, and Appendix B offers additional details of the evaluation process.

Table 3. Comparison of actual computational costs. See section 6.2 for details.

		AgES	shPRP	Speedup
Setup	Provider	-	50,514 ms	-
	Requestor	-	-	-
	Total	-	50,514 ms	-
Search and Acquisition	Provider	50,530 ms	16 ms	3158
	Requestor	40,059 ms	116 ms	345
	Total	90,589 ms	132 ms	686

These results demonstrate a strong performance benefit for shPRP. After the setup process, requestor computation time decreases by 99.7%, and provider computation time almost entirely disappears. Also note that shPRP scales better than AgES (see Table 1).

The results also demonstrate shPRP's practicality. Providers in shPRP always perform a single encryption during the search process, so a provider's expected computational cost is a constant, reasonable 16 ms for any amount of published metadata. The quantity of metadata has a marginal impact on requestor computational costs, as requestors search an ordered list of the encryptions. This cost grows logarithmically with the number of published values and averages only 116 ms for 10,000 metadata items, so shPRP is also computationally viable for requestors. A requestor's work is parallelizable, making shPRP even more practical. Search communication costs are negligible if a requestor stores all provider-published data. With host servers, however, communications costs can become a constraining factor for large quantities of published encryptions.

6.3 Security

Because shPRP is a modification of AgES, its security may rest on AgES's security as shown in [2] and [12] provided that, under the assumptions of section 4.2, the changes do not compromise security. The modifications are:

– Providers publicly reveal encrypted metadata.
– Providers may use an encryption key indefinitely.
– Providers may publish to host servers.
– Requestors decrypt re-encrypted data instead of re-encrypting provider data.

Appendix C argues that, under the given assumptions, none of these modifications adversely impacts security. This argument does not rely on semi-honest participant behavior, meaning that, like AgES [12], shPRP does not leak information even under a malicious behavioral model.

7 Summary and Conclusion

A chief concern of many privacy-critical organizations is protection of information against illegitimate access. This emphasis can result in restrictive systems that successfully thwart objectionable parties yet also deter privacy-constrained requestors with valid claims. Private resource pairing attempts to connect such resource requestors and providers without violating privacy. While existing work addresses similar issues, no known prior work directly addresses this issue in a satisfactory manner. Research on private resource pairing uncovered several interesting topics warranting further research, including weaknesses in the present system and extensions that would make the present system more useful.

Several weaknesses exist in the present private pairing model. During the search process, requestors receive indefinite search capabilities for a given piece of metadata. A provider's encrypted metadata is constant as long as its key remains constant. During that period, a provider may publish additional metadata that a requestor has no right to search. The present private resource-pairing scheme would also allow curious parties to make numerous undesirable inferences if a provider modifies its metadata set or an owner updates its key. In addition, the malicious case extensions rely on a trusted third party in several cases. Means of reducing or removing these weaknesses are desirable.

Additional research could also add functionality. For example, some entities may partition resources by classification levels and limit searches by requestor clearance level. If providers use multiple keys and verification tickets include clearance data, this paper's solutions are sufficient, but more elegant solutions may exist. Also, organizations may have valid reasons for revealing only a subset of metadata or resources related to metadata. A means of ensuring that providers reveal appropriate data would be helpful, particularly if owners do not exist or cannot monitor metadata. Finally, future projects may wish to examine cases where a requestor's identity is confidential, host servers may collude, or metadata is fuzzy ([16] may offer insight for working with fuzzy metadata).

This paper presents a practical semi-honest solution that, under the unique constraints of private resource pairing, offers a 686-time computational speedup over the similar AgES protocol without compromising security. In addition, this work suggests means of preventing malicious participant behavior. The shPRP protocol and its extensions for preventing malicious behavior provide a concrete basis for future work in private resource pairing.

8 Acknowledgments

We thank David Evans for helpful advice and assistance in finding related work.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, H. Shi. Searchable encryption 2: Consistency properties, relation to anonymous IBE, and extensions (Full version). Cryptology ePrint Archive, Report 2005/254, 2005. http://eprint.iacr.org/2005/254/.
2. R. Agrawal, A. Evfimievski, R. Srikant. Information sharing across private databases. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pages 86-97. ACM Press, 2003.

3. M. Bellare, A. Boldyreva, A. Desai, D. Pointcheval. Key-privacy in public-key encryption. In Proc. of Advances in Cryptology ASIACRYPT 01. Springer-Verlag, 2001. LNCS 2248.
4. D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano. Public key encryption with keyword search. In Proc. of EUROCRYPT 2004, pages 506-522. Springer-Verlag, 2004. LNCS 3027.
5. D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In Proc. of CRYPTO 2001, pages 213-229. Springer-Verlag, 2001. LNCS 2248.
6. J. A. Calandrino, A. C. Weaver. Identity-based cryptosystem privacy. University of Virginia Technical Report CS-2006-15. 2006.
7. B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan. Private information retrieval. In Journal of the ACM, Vol. 45, No. 6, pages 965-982. ACM Press, 1998.
8. M. J. Freedman, K. Nissim, B. Pinkas. Efficient private matching and set intersection. In Proc. of EUROCRYPT 2004, pages 1-19. Springer-Verlag, 2004. LNCS 3027.
9. O. Goldreich. Secure multi-party computation. Manuscript, version 1.4. 2002. Available at http://www.wisdom.weizmann.ac.il/ oded/pp.html
10. Java 2 Platform Standard Edition 5.0 API Specification. 2004. Available at http://java.sun.com/j2se/1.5.0/docs/api/
11. L. Kissner, D. Song. Privacy-preserving set operations. In Proc. of CRYPTO 2005, pages 241 - 257. Springer-Verlag, 2005. LNCS 3621
12. Y. Li, J. D. Tygar, J. M. Hellerstein. Private matching. In Computer Security in the 21st Century, pages 25-50. Springer, 2005.
13. Liberty Alliance Project. Liberty Trust Models Guidelines. Version 1.0. 2003. Available at http://www.projectliberty.org/specs/liberty-trust-models-guidelines-v1.0.pdf
14. Office for Civil Rights, U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act (HIPAA). Available at http://www.hhs.gov/ocr/hipaa/
15. S. C. Pohlig, M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. In IEEE Transactions on Information Theory, IT-24, pages 106-110. 1978.
16. A. Sahai, B. Waters. Fuzzy identity based encryption. In Proc. of EUROCRYPT 2005, pages 457-473. Springer-Verlag, 2005. LNCS 3494.
17. B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1994.
18. R. Schoenberg, C. Safran. Internet based repository of medical records that retains patient confidentiality. In British Medical Journal, Volume 321, pages 1199-1203. 11 November 2000.
19. A. Shamir. Identity-based cryptosystems and signature schemes. In Proc. of CRYPTO 84, pages 47-53. Springer-Verlag, 1985. LNCS 196.
20. D. X. Song, D. Wagner, A. Perrig. Practical techniques for searches on encrypted data. In Proc. of 2000 IEEE Symposium on Security and Privacy. 2000.
21. B. R. Waters, D. Balfanz, G. Durfee, D. K. Smetters. Building an encrypted and searchable audit log. In Proc. of 11th Annual Network and Distributed System Security Symposium. 2004.

Appendix A: IBS Privacy

Define an identity-based signature scheme as a set of four algorithms: $IBS = (Setup, KeyExtract, Sign, Verify)$. $Setup$ accepts a security parameter, k, and any given common parameters, $commonParams$, and generates a set of public parameters, $params$; a random master secret, s; and the corresponding master public key, pk. $KeyExtract$ accepts a master secret, public parameters, and an identity, id, for which the private key, v_{id}, is to be extracted. $Sign$ accepts a user's secret key, public parameters, and a message, m, and it outputs the signature, sig. $Verify$ accepts a master public key, public parameters, an identity, a message, and a signature. It outputs true or false. If an identity-based signature scheme, IBS, possesses IBS privacy, an adversary, A, is unable to gain more than a negligible advantage at guessing the value b in the following experiment $(\mathrm{Exp}_{IBS,A}^{ibs-priv-b}(k))$:

1. The challenger computes $(params_0, s_0, pk_0) = Setup(commonParams, k)$ and $(params_1, s_1, pk_1) = Setup(commonParams, k)$ and presents $params_0$, pk_0, $params_1$, and pk_1 to A.
2. A may use an oracle to derive secret keys for any identities under either instantiation. Eventually, A must choose a valid identity, id. A must not have queried for the secret keys corresponding to id. A returns id to the challenger and may save any state information.
3. The challenger randomly selects a bit $b \in \{0, 1\}$ and a random nonce, n, within the message space, computes $sig = Sign(v_{id,b}, params_b, n)$, and returns sig to A.
4. A may use the oracle again to derive private keys but may not derive private keys associated with id. Eventually, A must submit a guess, b', for b based on all known information, including saved state data.

A's advantage is defined as:

$$\mathrm{Adv}_{IBS,A}^{\text{ibs-priv}}(k) = \Pr[\mathrm{Exp}_{IBS,A}^{\text{ibs-priv-1}}(k) = 1] - \Pr[\mathrm{Exp}_{IBS,A}^{\text{ibs-priv-0}}(k) = 1]$$

A's advantage is negligible if $\mathrm{Adv}_{IBS,A}^{\text{ibs-priv}}(k)$ is a negligible function over k.

Shamir's original identity-based signature scheme lacks IBS privacy. Each instantiation must use a different, publicly available modulus [19]. Thus, the same technique for distinguishing between public keys in RSA systems is applicable to this identity-based signature scheme.

Appendix B: Evaluation Process

Only shPRP providers have a setup process. Therefore, the setup duration for requestors and AgES providers is trivially zero. Fourteen trials, with the two highest and two lowest results excluded, established the average setup duration of shPRP providers. An equivalent procedure assessed shPRP provider performance during the search and acquisition process. In AgES, providers are active at two points during the search process: to supply encrypted metadata and to encrypt requestor metadata. These tasks precisely correspond to the shPRP provider setup and search processes. Thus, the AgES provider average is the sum of the shPRP averages for each task. AgES and shPRP requestors underwent two rounds of testing. In the first round, requestors performed fourteen searches for existing metadata. In the second round, requestors searched for fourteen nonexistent metadata items. The overall average was the mean of all results, excluding the two highest and two lowest results from each round. Results do not include time waiting on providers or downloading data.

Appendix C: Security of shPRP

Recall that the modifications of AgES for shPRP are:

- Providers publicly reveal encrypted metadata.
- Providers may use an encryption key indefinitely.
- Providers may publish to host servers.
- Requestors decrypt re-encrypted data instead of re-encrypting provider data.

This section argues that, given the assumptions of section 4.2, none of these changes result in shPRP leaking more information than AgES, which does not leak information in the semi-honest or malicious case.

Through public revelation of metadata, a provider, P, allows any curious entity (requestors, other providers, host servers, etc.) to acquire and analyze the provider's encrypted metadata hashes. This set of encrypted hashes is equivalent to the set that P, with metadata M_P, would provide to a curious party, C, with metadata set $M_C = \emptyset$, under the AgES protocol. Agrawal et al. demonstrate that C can learn only $|M_P|$ and $M_C \cap M_P = \emptyset$ from this data [2].

Similarly, shPRP's use of constant provider keys makes cryptanalysis no less difficult. C could store and perform cryptanalysis on the equivalent set of encrypted hashes it receives from P under the AgES protocol. In both cases, security against cryptanalysis is dependent on choice of commutative encryption function, hash function, and key length. The use of constant provider encryption keys does mean that the encryption of a piece of metadata will remain constant, however. Because encryptions remain constant and providers publicly disclose encrypted data, curious parties may observe and draw inferences from the publication time of data if providers do not publish data all at once. Also, a curious party could trivially observe modification or removal of encryptions. To avoid issues with publication, modification, and removal, section 4.2 states that either providers must publish all data in unison or inferences must reveal no confidential data. Future work may establish a more satisfactory solution.

Provider signatures and the assumptions of section 4.2 prevent host servers from imperceptibly modifying data or drawing undesirable inferences. Beyond attacks that this paper explicitly does not consider (see section 3), host servers introduce no additional known weaknesses.

Finally, a requestor's choice to decrypt data rather than re-encrypt it has no impact on security. Nothing prevents entities from decrypting legitimately acquired data from the AgES protocol.