Private Resource Pairing

Joseph A. Calandrino* and Alfred C. Weaver
University of Virginia
Department of Computer Science
{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 the state-of-the-art 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 [1, 8]. 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:

- Efficiency: The unique constraints of the private resource-pairing problem allow for the use of pre-computation and other techniques that significantly decrease the computational costs of searches over the state-of-the-art private matching solution.
- Security: While a private matching solution exists that prevents participant dishonesty [8], 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. The safety of any treatment for the patient's condition is highly dependent upon the patient's 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 use of such a repository [13]. 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 [10]. In this scenario, a system for privately pairing record requestors and record possessors would be desirable to protect patient privacy. Such a system must enforce requestor need to know and prevent provider forgery of record possession.

* This research was performed in part while on appointment as a U.S. Department of Homeland Security (DHS) Fellow under the DHS Scholarship and Fellowship Program, a program administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

National Intelligence Scenario: Presume that a national security analyst is studying a region and determines that a particular national landmark may be at immediate risk. Numerous agencies may possess classified intelligence related to the landmark or the threat. To protect information confidentiality, agencies may have strict policies against revealing even metadata pertaining to resources they possess. For example, a national intelligence agency may have records of several communications related to a given threat, but the agency may wish to appear unaware of the threat by restricting access to both those resources and data regarding those resources. Similarly, the analyst may be reluctant to reveal the metadata that interests her. In this scenario, necessary privacy hampers equally necessary availability. A private pairing method would be desirable to link the analyst with all 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 provides a system for privately pairing resource requestors and providers given that entities are honest but curious. Section 4 extends the system to prevent malicious behavior. Section 5 evaluates the theoretical cost, applied performance, and security of this paper's private resource-pairing solution as compared to the state-of-the-art private matching solution. Finally, section 6 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 [1]. Their paper establishes protocols to allow two entities maintaining separate databases to determine query results across both databases without revealing information beyond the result. A trusted third party solution to this problem is unreasonable, as it would require both entities to completely trust the third party. Agrawal et al. address not only intersection queries but also intersection size, equijoin, and equijoin size queries. The intersection query problem is also known as the private matching problem [8]. Assuming Alice wishes to determine the intersection between her database and Bob's database, the Agrawal, Evfimievski, and Srikant private matching solution (AgES) works as follows:

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

Agrawal et al. demonstrate the computation and communication benefits of their protocols over a solution based on secure multi-party communication.

AgES assumes semi-honest, or honest but curious, behavior of both entities participating in the protocol. This assumption means that entities will adhere to the protocol during the query process, but they may store and analyze intermediate results to derive additional information [1, 6]. For example, neither Alice nor Bob will falsely claim to possess elements, but Alice may scrutinize B_{e_B} in an attempt to learn more about Bob's data set. While many organizations may demonstrate semi-honesty at minimum to protect their reputations, a lack of tangible protection measures is insufficient in the case of sensitive data. In addition, entities may prefer a protocol in which they cannot lie to protect

themselves against damaging false accusations of impropriety. For example, if two countries, Atlantis and Babylon, have an agreement to share intelligence data and their relations sour, Babylon may accuse Atlantis of bogus data possession claims. To refute the accusation, Atlantis would need to demonstrate data possession, potentially revealing sensitive information in the process. If Atlantis chose not to refute the allegation, the country's reputation could be damaged irreparably.

Li, Tygar, and Hellerstein offer a comprehensive exploration of solutions to the private matching problem under both a semi-honest and a malicious model [8]. A malicious model assumes that entities may choose to lie given the opportunity. To prevent bogus data possession claims, Li et al. propose the use of data ownership certificates. However, as [8] proposes them, data ownership certificates are not directly applicable to the private resource-pairing problem. Since a requestor may not possess the desired resource, the requestor would not have an ownership certificate for that resource. Li et al.'s model requires both entities to have ownership certificates to verify each other's.[1] For the private matching problem, this property is desirable: Bob has no legitimate reason to verify possession of data he does not also possess. An alternate solution to the resource possession problem is necessary for private resource pairing.

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

Devious parties also may choose not to disclose possessions. Li et al. leave this problem to future work. Allowance of nondisclosure is actually desirable under certain circumstances but not under others. This paper presents a method that allows individuals to periodically ensure that their resources are available (see section 4.2), a compromise that may be sufficient in some situations.

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

Li et al. consider threats that parties, including protocol participants, are curious, are dishonest, terminate the protocol early, or collude [8]. Malicious parties may mount additional attacks, such as denial-of-service attacks, against protocol participants or the implementation infrastructure, however. While a more efficient solution may be more resistant to denial-of-service attacks, this paper does not explicitly consider such additional threats.

2.2 Additional Work

Private information retrieval (PIR) seeks to allow parties to retrieve database entries without disclosing information about the entries they desire [5]. Unfortunately, PIR does not offer any assurance that parties accessing a database need to know the information that they privately retrieve. Nonetheless, an efficient implementation of PIR would be useful to this paper's private pairing solutions (see sections 3 and 4).

Waters, Balfanz, Durfee, and Smetters present a method to allow searches on encrypted audit logs [16]. The scheme could protect provider privacy and enforce need to know for resource requestors. Requestors would need to reveal search strings to a third party, however, potentially providing the third party with confidential information.

Song, Wagner, and Perrig present a means of searching on encrypted data [15]. Their work allows the use of encrypted queries to search encrypted data on untrusted servers. Unfortunately, to make an encrypted query, the entity that originally encrypted the data (or a third party) must learn the query and provide the ability to search. In the private resource-pairing problem, providers would therefore need to learn the requestors' searches. To hide queries from providers, a commutative encryption based scheme similar to that of private matching may be a necessary addition to their work.

Zero-knowledge proofs allow a prover to demonstrate to a verifier that the prover possesses a given piece of information without revealing that information. For example, a prover could demonstrate possession of Alice's unique identity-confirming key without revealing the key itself [12]. Although a prover does not need to reveal the information itself, it must reveal to the verifier what piece of information it possesses. In the example, the verifier must know that the prover is demonstrating possession of Alice's, not Bob's, key. While such proofs may enable demonstration that a requestor has a legitimate need for a resource or a provider has a resource, they would require one party to publicly reveal the requested or possessed resource, violating a requirement of private resource pairing.

3. SEMI-HONEST CASE SOLUTION

This paper first presents a protocol for private resource pairing under a semi-honest behavior model. As with private matching, this assumption is somewhat unrealistic in practice. Section 4 presents extensions to the semi-honest protocol that allow enforcement of need to know and proof of resource possession.

3.1 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. Resource providers choose random encryption keys and then hash and 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 more efficiently handle requestor searches later.

When a requestor wishes to search for and acquire resources tagged with a given piece of metadata, the requestor 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 encryption 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 openly requests resources related to the metadata. By decrypting the single item of search metadata rather than re-encrypting every published piece of provider metadata, requestors significantly lighten their computation load. A more rigorous explanation follows shortly.

3.2 Assumptions

This protocol makes several assumptions necessary to the scheme's security. First, a resource requestor's identity alone must imply nothing confidential to providers or servers. Second, resource providers must publish encrypted metadata all at once, or providers must be able to assume that other entities cannot draw undesirable conclusions from metadata publication order, modification, or removal. If providers use host servers, requestors must download all data from any given server. Private information retrieval is unreasonable since even a successful search will require requestors to download half of a server's data on average. If a requestor did not take this precaution, a server could infer whether and on what piece of encrypted data a search is satisfied and use this information to uncover search patterns. 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 determine whether the request went unsatisfied. Finally, this solution assumes that resource metadata is in no way "fuzzy."

3.3 Detailed Process

This paper's protocol for private resource pairing under the semi-honest model requires a setup process and a search and acquisition process.

For the remainder of this paper, shPRP denotes this semi-honest private resource pairing solution.

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: $P_h = \{h(m_P) \mid m_P \in M_P\}$.
4. P encrypts the elements in P_h, producing $P_{e_P} = \{f_{e_P}(p_h) \mid p_h \in P_h\}$.
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 publish e_P, the corresponding decryption key, or both to the escrow service during step two. In addition, 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 (see Figure 1):

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 pairing process with a potential provider. If R reuses a key, providers could look at previous requests and determine that R seeks the same value, even if they don't know what the value is.
   - If R is also a provider, R must choose a different key from its provider encryption key. Otherwise, other providers could look at R's published values to determine whether R possesses a resource with the metadata it seeks.
2. R computes the hash of m: $m_h = h(m)$.

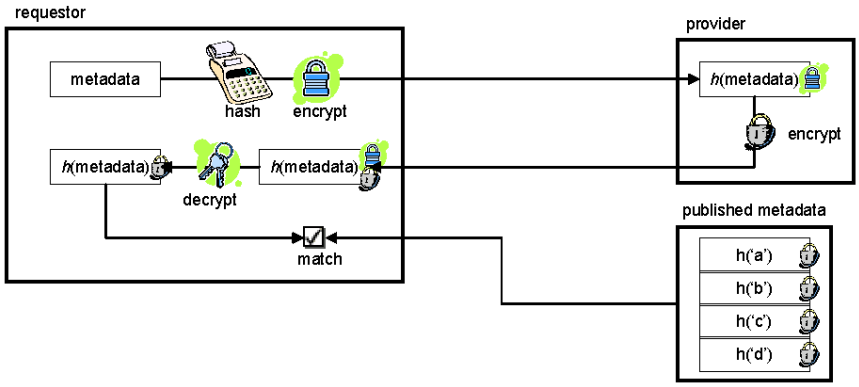

Figure 1: shPRP search and acquisition process. Commutative encryption allows requestors to get provider encryptions of data without revealing the data to the provider. See section 3.3 for details.

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}: $m_{e_P} = f_{d_R}(m_{e_R,e_P}) = f_{d_R}(f_{e_R}(f_{e_P}(m_h))) = f_{e_P}(m_h)$.
8. If *P* hosts its data on a server, *R* downloads P_{e_P}, the accompanying signature data, and any additional 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* approaches *P* and asks for resources with metadata *m*.

4. MALICIOUS CASE SOLUTION

A malicious case solution seeks to prevent two forms of potential participant dishonesty. 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.

Note that the modified protocol retains all assumptions of section 3.2.

4.1 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 3.3). 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 confirm that m_{e_R} represents *m*.

The processes by which a requestor, *R*, may acquire tickets from a supplier, *S*, is as follows:

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_R,e_S} matches m_{e_S,e_R}.

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}.

The order of steps two and three is arbitrary. They may occur in opposite order or in parallel.

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 tickets from any supplier. Various models exist for establishing trust, including direct, hierarchical, and distributed trust models [9]. This choice is implementation-specific; the use of any model is acceptable for private pairing.

Two ticket supplier models exist: internal and external. Both models assume that ticket suppliers will not collude with malicious requestors to allow illicit access to data or resources. In addition, ticket suppliers should be unable to initiate searches. Otherwise, a rogue supplier could access unlimited resources. An internal supplier model assumes that potential requestors are part of larger organizations and that requestors may reveal their searches to ticket-granting parties in their organizations. The ticket-granting party verifies that present conditions warrant a search for resources.

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 make a determination, 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 impartial external parties to verify the need to search. In this case, requestors must fully trust the verification party with their search metadata, and providers must trust the verification party not to collude with dishonest requestors. External verification is appropriate for cases such as business agreements in which parties agree to limited resource sharing. Members of either business may possess bias in interpretation of the agreement, creating the need for an impartial arbitrator.

4.2 Proving Resource Possession

As with proving need to know, this section presents two models for proving resource possession. In one model, all resources described by a piece of metadata have a clear owner. The metadata therefore implies an owner in a manner obvious to providers and requestors alike. For example, a patient with a unique biometric would have legal control over the distribution of medical records tied to her biometric [10], making her the effective owner of those records. Under the second model, metadata either does not imply an owner or implies numerous owners. For example, the keyword "explosives" may be applicable to many intelligence resources, but the word alone does not imply a clear owner of those resources. A solution under the second model is also applicable to the first scenario, since entities can ignore implied ownership. A solution for the first scenario is preferable when applicable, however, as it allows owners to better control their resources. This paper's solutions in both cases rely on the use of identity-based signature systems.

Identity-Based Signatures: Identity-based signature systems, first proposed by Shamir, allow the use of one's identity as their public key [14]. For example, Alice may sign her messages using a private key associated with her unique identity ("alice@virginia.edu"). To verify her signature, Bob can simply pass the message, the signature, and "alice@virginia.edu" 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.

Key Privacy: Bellare, Boldyreva, Desai, and Pointcheval first formalized the property of key privacy in public-key cryptosystems. Given this property, an adversary that possesses a piece of ciphertext can gain no more than a negligible advantage in determining which public key out of a given set produced the ciphertext [2]. For example, RSA lacks key privacy because an adversary can gain an advantage based on the publicly known modulus [2]. When metadata implies an owner, the security of this paper's possession scheme relies on a principle similar to key privacy.

Metadata Implies an Owner: Assume the scenario in which metadata implies an owner of resources associated with the metadata. In this case, proof of possession relies on the use of identity-based signature systems with the novel property of system privacy, similar to key privacy.

Suppose that multiple instantiations of an identity-based signature scheme exist. Each instantiation has a different master secret, but other parameters may match across instantiations. The use of different master secrets means that each instantiation will produce a different mapping between identities and private keys. Now assume that an adversary chooses an identity, and an arbitrary

instantiation produces that identity's signature for a nonce. The adversary receives a copy of the signature but not the nonce. If, given some parameters, the adversary is unable to gain more than a negligible advantage in determining the instantiation that produced the signature, this signature scheme provides system privacy under those parameters. For example, Shamir's original identity-based signature scheme lacks system privacy. Each instantiation must use a different, publicly available value of n, and part of the signature is the result of a value modulo n [14]. Thus, the same technique for distinguishing between public keys in RSA systems is applicable to this identity-based signature scheme.

To allow proof of resource possession, some setup is mandatory. Owners must agree on a common identity-based signature scheme along with any parameters necessary to provide system privacy. Each owner then generates a unique instantiation of the scheme under the established constraints. Owners publish public parameters necessary to verify signatures that they produce. Either requestors and providers or public repositories must maintain lists of owner identities with their public parameters. If a repository maintains the data, private information retrieval or total repository downloads must be efficiently possible so that repository operators cannot infer which owner's resources a requestor seeks. Given a large number of non-colluding servers, PIR may be feasible, as requestors will seek a relatively small amount of data at a predetermined index, the owner's identity.

To demonstrate possession, additional steps are necessary between steps four and five of the shPRP setup process (see section 3.3). For each piece of metadata $m_P \in M_P$:

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 hash 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 identity-based signature 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 identity-based signature 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. They may occur in opposite order or in parallel.

Signing with the private key associated with the provider's identity prevents two providers from using the same commutative encryption keys and sharing signed values. Because values in P_{e_P} are indistinguishable from random values in polynomial time [1] and owners use system private signature schemes, an adversary will have at most a negligible advantage in determining the owner that produced any given signature.

Following acquisition of signatures, P can reorder the signatures lexicographically and publish the signed data. If P reordered by the original encryptions rather than signatures, adversaries could estimate the pre-signed data ranges to gain an advantage in guessing the signing instantiations.

If owners can privately retrieve data from servers, they can verify at any time that the servers continue to host their data, preventing providers from removing data without permission.

Because only an owner possesses its master secret, only it can produce private keys and generate signatures. Owners can delegate signing responsibilities to a trusted third party, such as a contracted service. Owners also can provide master secrets to an escrow system if desired. 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, though this procedure could present a problem if others could infer confidential information from the update process. If an owner updates its public parameters at nearly the same time several providers update their published data, an adversary can infer that the providers published metadata related to the owner's resources. This paper leaves resolution of update issues to future work.

With two exceptions, the search and acquisition process remains the same for resource requestors as under shPRP. First, a requestor must obtain the owner's public parameters. Second, using the provider's identity as a public key, the requestor must attempt to verify provider-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. Whenever 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 later, 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 [13].

Metadata Does Not Imply an Owner: Assume that metadata does not imply a single owner of associated resources. Because metadata is not irrefutably linked to owners, no party has a legitimate right to confirm or deny possession of resources associated with metadata. For requestors to accept claims of possession, a trusted third party, centralized or distributed, seems necessary to validate provider possession based on established rules. Requestors can later verify provider possession without the intervention of a third party, however, 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 scenario, the publication process is exactly 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 signature system's parameters rather than retrieving parameters during each search. This scheme suffers from an issue common to identity-based cryptosystems: 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 exception list.
- Scrap all keys and move to a new master secret. As this option would entail reproducing all signatures, it is impractical. Gradual migration to a new master secret may be acceptable, however. For example, if the private key for "provider" is compromised, the third party could immediately migrate all keys starting with 'p' and publish a schedule for migrating 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.

5. EVALUATION

The AgES protocol offers the closest match to this paper's private resource pairing solutions, making AgES 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.
- Private resource pairing occurs under the semi-honest model, as AgES has no comparable scheme for a malicious model. Thus, the comparison is AgES versus shPRP.
- Even if shPRP uses host servers, it does not create signatures. Signature costs would be dependent on implementation decisions.
- Once finished, the AgES protocol 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 solution to the private resource-pairing problem. Due to the requirements of the private matching problem, AgES does not use pre-computation, for example. Nevertheless, as the leading existing solution for the private matching problem, AgES provides the most appropriate comparison.

5.1 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 respectively for the chosen commutative encryption scheme. C_h is the cost of generating a hash with the chosen hash function. m is the length of the desired metadata, while c is the metadata ciphertext length. A provider has p items of metadata.

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

		AgES	shPRP
Setup	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	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 1: Computational costs of AgES and shPRP. See section 5.1 for variable definitions.

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

Table 2: Communication costs of AgES and shPRP. See section 5.1 for variable definitions.

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 $2c + m$. Tables 1 and 2 summarize these results in greater detail.

After the initial setup, shPRP significantly lightens the computational cost for both 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.

5.2 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 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 [7]. For the tests, providers maintained 10,000 items of metadata. To achieve a fair comparison, the AgES implementation contained an obvious optimization: requestors do not immediately encrypt all provider-published data but instead encrypt provider data line-by-line, attempting to match each result to the re-encryption of the desired metadata. When a match exists, this optimization reduces requestor encryptions by approximately 50% on average. All tests ran on a 3.2 GHz Pentium 4 with 512 MB of RAM. Table 3 shows the results of these tests.

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

		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

Table 3: Actual computational costs of AgES and shPRP. See section 5.2 for implementation details.

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.

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).

These results also demonstrate the practicality of shPRP. 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 number of published encryptions. The quantity of encryptions 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 encryptions and averages only 116 ms for 10,000 metadata items, so shPRP is also computationally viable for requestors. A requestor's work is highly parallelizable, making shPRP even more practical. Search communication costs are also 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.

5.3 Security

Because shPRP is a modification of AgES, its security may rest on the security of AgES as demonstrated in [1] and [8] provided that, under the assumptions of section 3.2, the modifications do not adversely impact security. The modifications of interest are:

- Providers publicly reveal encrypted metadata.
- Providers maintain a constant encryption key, potentially through multiple pairing processes.
- Providers may publish to host servers.
- Requestors decrypt the re-encrypted data they receive rather than re-encrypting provider data.

Through public revelation of metadata, a provider allows any curious entity to acquire and analyze the provider's encrypted metadata hashes. This set of encrypted hashes is equivalent to the set that a curious party, C, with metadata set $M_C = \varnothing$ would receive if it approached the provider, P, with metadata set M_P and entered the AgES protocol (given that the provider used the same encryption key in both cases). Agrawal et al. demonstrate that C can learn only $|M_P|$ and $M_C \cap M_P = \varnothing$ from this data [1]. Similarly, C would gain no advantage in performing cryptanalysis from shPRP's use of constant provider encryption keys. 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 means that the encryption of any piece of metadata will remain constant. 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 3.2 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 to this problem.

Provider signatures and the assumptions of section 3.2 prevent host servers from imperceptibly modifying data or drawing undesirable inferences. Beyond attacks that this paper explicitly does not consider (see section 2.1), 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.

6. 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. Unfortunately, these systems can 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, extensions that would make the present system more useful, and issues of relevance beyond private resource pairing.

Several weaknesses exist in the present private resource pairing model. During the search process, requestors receive indefinite search capabilities for a given piece of metadata. While tickets may expire, a provider's encrypted metadata is constant as long as its encryption key remains constant. During that period, a provider may publish additional metadata that the requestor has no right to search. In addition, the present private resource-pairing scheme would allow curious parties to make numerous undesirable inferences if a provider modifies its metadata set or an owner updates its key. Means of better constraining search capabilities and of preventing unwanted inferences are desirable.

Additional research could also lead to a more useful system. For example, in some cases, entities partition resources by classification levels. Parties with high-level clearances might have an entitlement to search low-level resources but not vice versa. If providers use different encryption keys for different clearance levels and verification tickets include the requestor's clearance level, solutions in this paper are applicable to multi-level secrecy. A more elegant solution may be possible, however. Also, as organizations may have valid reasons for not sharing metadata or for only revealing a subset of resources related to a given piece of metadata, attacks such as hiding attacks are particularly tricky. A means of ensuring that providers publish all appropriate data and reveal all appropriate resources would be helpful, particularly if owners either do not exist or are unable to monitor metadata related to their resources. Finally, future projects may wish to examine cases where a requestor's identity is confidential, where host servers may collude, or where metadata is fuzzy.

The concept of system privacy may be of importance beyond private resource pairing. This paper presents system privacy exclusively with regards to identity-based signature systems. A more comprehensive exploration of that concept in terms of both signature systems and general identity-based cryptosystems would be useful. For example, several businesses may wish to maintain separate master secrets, but they may want to prevent adversaries from determining the business destinations of encrypted messages.

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 offers a 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.

7. ACKNOWLEDGEMENTS

We thank the Department of Homeland Security for providing funding for this effort. We would like to thank David Evans for his assistance in the discovery of related work as well as his helpful comments on this paper. We thank Alexandre Evfimievski for his clarification of the performance of the AgES protocol. We also thank Brent Waters for his useful advice during the early stages of this research.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. of CRYPTO 2001*, pages 213-229. Springer-Verlag, 2001. LNCS 2248.

[5] 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.

[6] O. Goldreich. Secure multi-party computation. Manuscript, version 1.4. 2002. Available at http://www.wisdom.weizmann.ac.il/~oded/pp.html

[7] Java 2 Platform Standard Edition 5.0 API Specification. 2004. Available at http://java.sun.com/j2se/1.5.0/docs/api/

[8] Y. Li, J. D. Tygar, J. M. Hellerstein. Private matching. In *Computer Security in the 21st Century*, pages 25-50. Springer, 2005.

[9] 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

[10] 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/

[11] 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.

[12] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1994.

[13] 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.

[14] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of CRYPTO '84*, pages 47-53. Springer-Verlag, 1985. LNCS 196.

[15] 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.

[16] 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.