Application and Analysis of Private Matching Schemes Based on Commutative Cryptosystems

Zbigniew Kwecka, Prof William J. Buchanan, Duncan Spiers
Centre for Distributed Computing and Security, Napier University, Edinburgh, UK
z.kwecka@napier.ac.uk
w.buchanan@napier.ac.uk
d.spiers@napier.ac.uk

Abstract:

Privacy issues are becoming a key focus with software systems. Surveys show that the invasion of privacy is among the things people fear the most from the coming years. These fears seem to be justified, in the light of recent events involving the UK government. Thus, according to the EU Telecoms Commissioner the UK government breach European privacy laws by allowing a group of UK based Internet Service Providers (ISPs) to intercept communications of their users for behavioural advertising purposes. In this case it was complaints from the concerned public that made the EU Commission examine the privacy implications. Yet, on the contrary, popularity of various social networking portals, where users publish their personal and sensitive data publicly, is growing. Therefore, some argue that users should not expect any level of privacy in the digital world. Such climes are backed-up by the fact that majority of Internet users are unconcerned about the *digital footprint* they leave behind. What is overseen is the control factor. Users want to have the right to decide what information about their lives is in the public domain. Consequently, 'one-size fits all' solution to privacy concerns does not exist, as everybody perceives privacy in a slightly different way. Therefore, parties involved in data-handling, including social networking portals, need to research and implement privacy technologies that can keep their customers happy and make the operation comply with local security and privacy directives in many locations around the globe.

This paper gives an insight on how Privacy Enhancing Technologies (PETs) can be used to perform private matching operations in large datasets. These operations can be used by data-holders and individuals to compare or to retrieve information in a private manner in cases where trusted third party does not exist or trusted third party it is used trusted for authentication purposes only. Thus, they can provide users with greater control over how their data is used. They include equality tests, dataset intersections, dataset equijoins, and symmetric private information retrieval protocols. Application of such private operations lies in the area of pervasive computing, database interaction, auditing and data acquisition. Here it is shown that PETs based on commutative cryptosystems are most efficient in performing these operations. Therefore, these cryptosystems are examined in detail. Currently anyone wishing to implement PETs based on commutative cryptosystems will quickly notice that such cryptosystems cannot be found in any of the popular cryptographic suites. The reason for this is the fact that these cryptographic algorithms are expensive to run in comparison with other encryption technologies and have limited area of usage in security applications. Thus, the key contribution of this paper is a guide to implementing commutative cryptosystems, using common open-source cryptographic packages. Consequently, this should enable developers and researchers to further investigate the existing PETs and propose new systems employing the notion of the commutative cryptography.

Keywords: commutative cryptography, data acquisition, privacy enhancing technologies, data mining, private matching

1. Introduction

In September 2000, a poll asking Americans what they fear the most in the upcoming century was published by the Wall Street Journal (in Swire & Steinfeld, 2002). The respondents were given a number of scenarios to list in order of most feared. These scenarios included international terrorism, global warming and world war. It turned out that among these the most feared was invasion of personal privacy. Things changed slightly after the 9/11, however, once enough time will pass from these tragic events privacy will, most likely, become a major concern once again. This trend is shown in the survey conducted by the Washington Post in 2006 (Balz & Deane, 2006), where 32% of respondents agreed that they would prefer the federal government to ensure that privacy rights are respected rather than investigating possible terrorism threats. This was 11% increase from the similar survey conducted in 2003.

European Commission has requested UK government to stop the *Phorm* programme, which analyses behaviour of internet users at ISP level for targeted advertising purposes (RAPID, 2009). This was a response to a number of complains by UK internet users received by the EU Telecoms Commissioner. Privacy activist groups welcomed this move by the European Commission, since the *Phorm* programme indulges into deep packet analysis of user traffic, which is equivalent to tapping into Internet conversations of every customer from participating ISPs. Consequently, one of the larges on-line retailers in UK, Amazon, has decided to block *Phorm* from analysing traffic to and from their domains, and other businesses are likely to follow (Waters, 2009). This shows large groups of users are ready to fight their right for privacy, since it was their protests that triggered the actions of the EU Commission. However, Amazon itself tracks users in order to build their behavioural profiles, but it does it overtly and users can choose not to deal with this particular retailer. Thus, it can also be concluded, that it is the control element that is appreciated by the Internet users.

On contrary to the demands for privacy, a lot of the Internet users jeopardise their privacy and security by making their personal profiles openly available on social networking portals (Anderson, 2008). Some information posted on the Internet by unaware users is sensitive and can be used by adversary in identity theft and other social engineering exploits. However, this risk can be reduced by appropriate awareness campaigns and filtering mechanism integrated into web browsing or antivirus software. Where at the same time, relevant surveys show that users are becoming more relaxed about the *digital footprint*, i.e. traces of lives, they leave behind on the Internet (Stross, 2007). While this footprint can affect further lives of the users consciously choosing to publish their personal information on-line, it should remain their choice. Publishing information to social networking portals should not be treated as privacy or security risk, but more, as a lifestyle decision, as long as it is an informed decision.

The previous paragraphs show that users of software systems have various requirements as to their privacy. However, they all have one in common: all users want to maintain the control over the way their data is being used. In European Union appropriate legislations are there to safeguard this level of control over the private data. However, it is different in other parts of the world. Also, while the Western Society enjoys many liberties, people around the word still need to hide their beliefs and sexual preferences. Thus, there is a need for technologies that can allow like-minded people to find each other in the global village and communicate securely. In some cases soft security technologies can be used to improve privacy, these are policies and access control mechanisms that are more suitable than use of technologies supported by cryptography and number theory. However, these technologies can help only in the case where trusted third party exists. For example in the case of social networking portals soft security and training can be sufficient, since mutually trusted third party, the portal regulated by various privacy laws, is there to provide any level of privacy required by the users. However, the need for privacy enhancing solutions based on cryptography is still there. When parties cannot agree upon a trusted third party, or there is a need for secrecy of communication, mathematics can become a trustee of the secrets. In this paper following scenarios are used to help illustrate the need for private operations on datasets:

Scenario 1:
Pervasive computing enables electronic devices to assist people in their everyday lives, even in the most intimate scenarios. However, in a distributed network of pervasive computing devices, a common trusted third party does not always exist. Imagine avatars running on PDAs and mobile phones that scan the surroundings in search of an ideal match for a date or a social networking contact. Such avatars, based on their owner's profile, could compare profiles supplied by avatars of other users nearby. A simple solution would provide profiles openly to others users of this dating or social networking application. Nevertheless, this could make the user vulnerable to many well targeted scams and attacks. With the use of PETs described in this paper the profiles could be confidential and still the avatars would be capable of comparing them.

Scenario 2:
Security services investigate a terrorist suspect, Bob. They need to obtain the information on his recent Internet activity. Currently, in the UK the procedure for obtaining such data involves serving a notice to the ISP of the suspect. This notice would need to identify Bob as a suspect, which would breach Bob's privacy and possibly jeopardise the investigation (Kwecka, Buchanan, Spiers, & Saliou, 2008). On the other hand, it is possible to build a system based on PETs that would allow the ISP to give access to Bob's data, without security services having to reveal his identity, and without breaching the privacy of the other data subjects in the ISP's database.

This research claims that PETs can assist in the solution to a number of private matching problems similar to those described by the scenarios. It shows that use of commutative cryptosystems is beneficial to this task and then provides a guide to implementing secure Commutative Encryption (CE) protocols.

2. Commutative Encryption

This research argues that private matching operations of datasets are most efficient when based on CE. The reasoning behind this claim is provided in Section 3, while this section discusses the principals of operation of CE protocols. Thus, in conventional cryptography when a plaintext message is encrypted with two different cryptographic functions E_A and E_B, the resulting ciphertext will be different depending on the order of the key application. Thus, for two different conventional cryptographic functions E_A and E_B, and for any $value$ in the allowed input range following is true:

Equation 1: $E_A(E_B(value)) \neq E_B(E_A(value))$ for conventional cryptography

For most cryptographic application, this is desirable behaviour, since it improves the security of the plaintext. However, most protocols described in Section 3 require the underlying cryptographic protocol where opposite of Equation 1 is true:

Equation 2: $E_A(E_B(value)) = E_B(E_A(value))$ for commutative cryptography

This requirement is met only by commutative protocols. Consequently, such properties make CE protocols an interesting choice for private matching algorithms.

3. Private matching dataset operations

The privacy concerns described in Section 1 fall into a general area of private matching techniques. This problem should not be confused with statistical data mining, where efforts are focussed on providing privacy in systems that share data for the purpose of statistical analysis. Thus, in the later problem anonymisation techniques can be successfully deployed to guard privacy of the records, however, these techniques cannot assist in private matching problem. In this section few privacy-preserving protocols allowing data comparison are described.

3.1 Private Equality Test (PEqT)

A key primitive in the area of private dataset operations is PEqT that allows two parties to test their secret inputs for equality. One of the first such tests published was that described by Frikken & Atallah (2003) and shown in Figure 3.1. This scheme was designed to work with CE protocols and the operation of the protocol can be summarised as follows, for two parties, Alice and Bob, each holding different CE keys:

1. Alice encrypts her input and sends it to Bob,
2. Bob encrypts the ciphertext received from Alice and sends it back,
3. Bob encrypts his secret input and sends it to Alice,
4. Alice encrypts the ciphertext containing Bob's input,
5. Alice compares the two resulting ciphertexts, if they are equal then her and Bob's inputs are equal,
6. Alice may inform Bob about the result.

The above PEqT is the first solution to the problem of comparing information without leaking it that does not require trusted third party. In 1996 Fagin, Naor and Winkler have collected and published a number of real-life substitute solutions to this problem, however, most of the proposed solutions which did not require trusted third party involved Bob and Alice to monitor each other. Therefore, these solutions were not real alternative to CE PEqT. To this date, a number of different PEqT schemes have been proposed, however, complexity of the other schemes is usually higher than this of CE solution. Boa and Deng (2001) proposed quite an efficient method for equality testing based on homomorphic encryption. However, this method requires a series of multiplications, an exponentiation, as well as a round of homomorphic encryption and decryption. The homomorphic protocol proposed for their scheme is ElGamal (EG), which itself requires two exponentiations modulo a prime during the encryption process and another one for the decryption operation. Consequently, complexity of their protocol is slightly higher than complexity of the CE PEqT scheme illustrated in Figure 3.1 when it is based on Pohlig-Hellman (PH) algorithm described in Section 4. However, only the slight difference in performance means that the decision of using one or the other method should be based on factors other than efficiency alone.

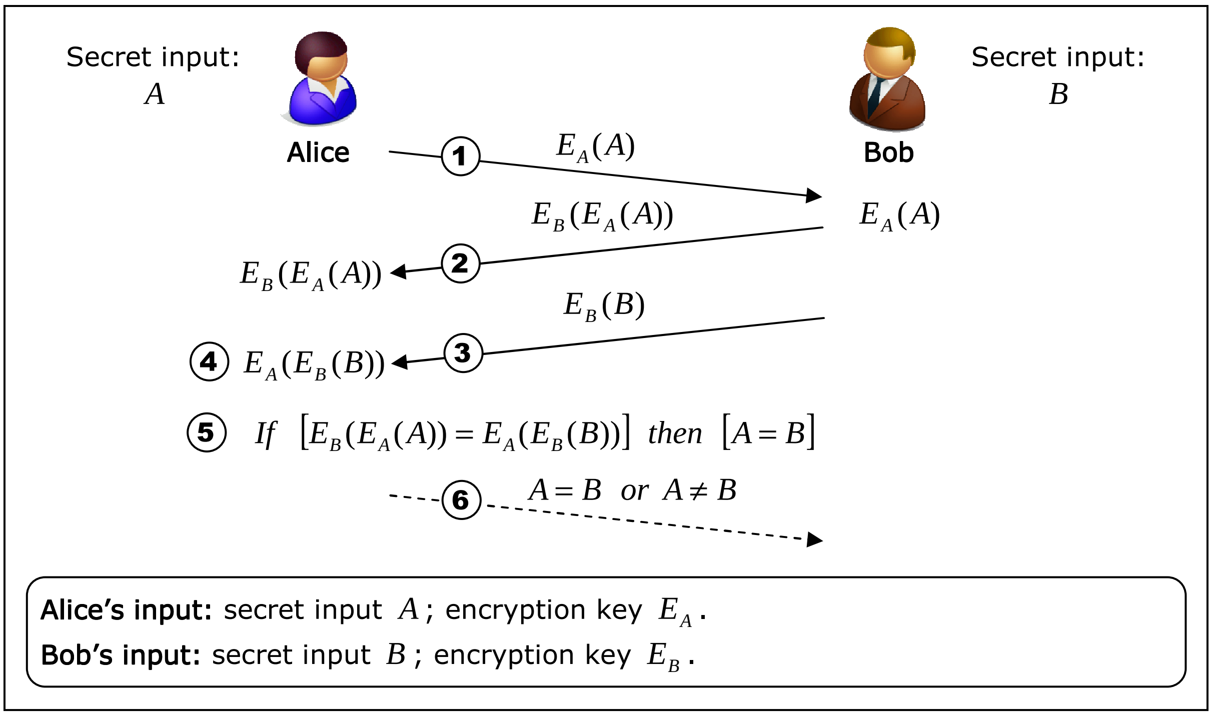

Secret input: A
Alice
Secret input: B
Bob
1 $E_A(A)$
$E_A(A)$
2 $E_B(E_A(A))$
$E_B(E_A(A))$
3 $E_B(B)$
4 $E_A(E_B(B))$
5 If $[E_B(E_A(A)) = E_A(E_B(B))]$ then $[A = B]$
6 $A = B$ or $A \neq B$
Alice's input: secret input A; encryption key E_A.
Bob's input: secret input B; encryption key E_B.

Figure 3.1 Private Equality Test

3.2 *1-to-n* Private Equality Test

It is rare for two parties to want to compare only a single value. More often, there are a number of inputs to be compared. Unfortunately, the complexity of most PEqT schemes cannot be lowered in case there are more inputs to the protocol. This applies also to the homomorphic scheme designed by Boa and Deng (2001). In the case of their protocol the whole PEqT would need to be run *n*, times in order for Alice to compare her input to *n* inputs held by Bob. Therefore, such operation would require O(*4n*) exponentiations in total. This is not the case with a slight modification of the CE based PEqT. Here Bob's inputs must be encrypted only once, using a single exponentiation operation, and Alice single input needs to be encrypted by her and Bob, and later decrypted by her. Thus, in total CE PEqT scheme requires O*(n+3)* exponentiations.

The CE PEqT protocol for Alice to compare her secret input with *n* inputs held by Bob involves the following steps:

1. Alice encrypts her input and sends it to Bob,
2. Bob encrypts the ciphertext received from Alice and sends it back,
3. Alice decrypts the ciphertext containing her input encrypted by her and Bob,
4. Bob encrypts all his secret inputs and sends them to Alice,
5. Alice compares the result form Step 3 with other ciphertexts received from Bob in Step 4, and if equal ciphertext are found, then she knows that Bob has got a common element with her set.
6. Alice may inform Bob about the result.

Figure 3.2 illustrates this protocol.

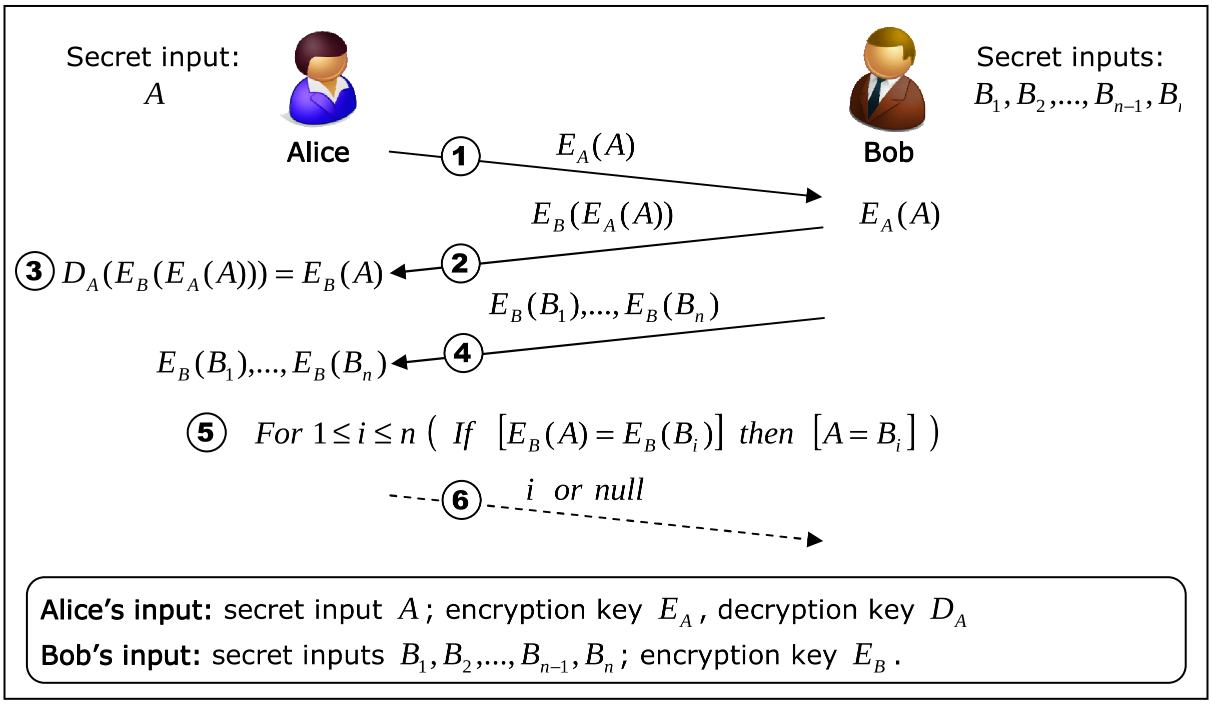

Secret input:
A
Alice
Bob
Secret inputs:
$B_1, B_2, \ldots, B_{n-1}, B$
1
$E_A(A)$
$E_A(A)$
$E_B(E_A(A))$
2
3 $D_A(E_B(E_A(A))) = E_B(A)$
$E_B(B_1), \ldots, E_B(B_n)$
4
$E_B(B_1), \ldots, E_B(B_n)$
5 For $1 \le i \le n$ (If $[E_B(A) = E_B(B_i)]$ then $[A = B_i]$)
6
i or null
Alice's input: secret input A; encryption key E_A, decryption key D_A
Bob's input: secret inputs $B_1, B_2, \ldots, B_{n-1}, B_n$; encryption key E_B.

Figure 3.2 *1-to-n* Private Equality Test

As far as the openly available literature goes the above protocol is the most efficient *1-to-n* PEqT protocol available. It can be used as a partial solution to the problem posed by the two scenarios provided in Section 1. Thus, in Scenario 1 an avatar belonging to Alice could communicate with other avatars and check do their owners like “music”, “horror movies” or “quantum physics”. However, each of these queries would require a separate run of the above protocol. In Scenario 2 the security services could use the protocol to find out is Bob a client of a given ISP, or index of Bob’s records in the ISP’s database, but they would be unable to request any more information about him in a private manner.

3.3 Private intersection, private intersection size

Section 3.1 concluded that PEqT can be efficiently performed using CE. Agrawal, Evfimievski and Srikant (2003) extended the above 1-to-n PEqT algorithm to construct a private intersection and private intersection size protocols. The operation of their private intersection size protocol can be described as follows:

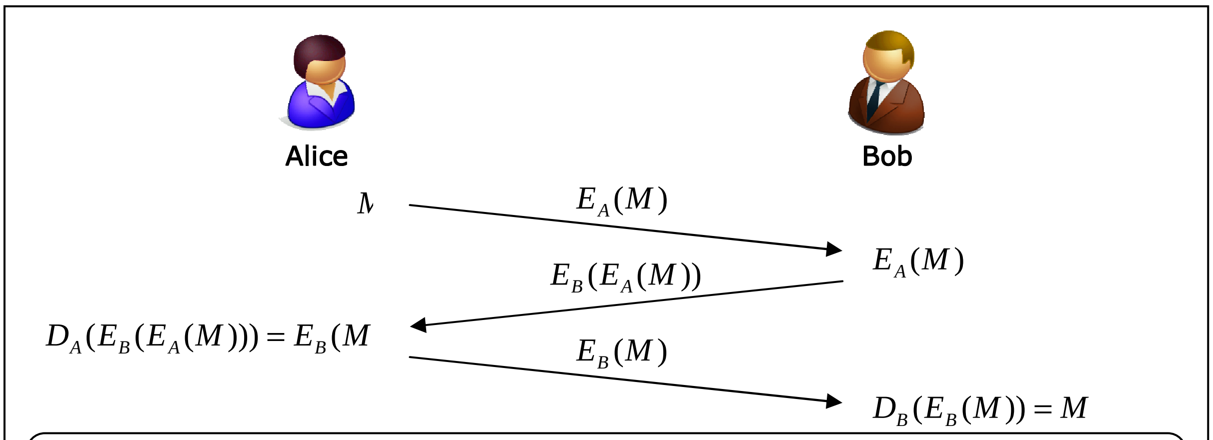
1. Alice and Bob apply hash function to their sets.
2. Both parties encrypt their hashed sets using their CE encryption keys.
3. Alice sends Bob her encrypted set in a lexicographical order.
4. Bob encrypts the set received from Alice in Step 3, and sends back the resulting ciphertexts reordered lexicographically.
5. Bob sends Alice his encrypted set in a lexicographical order.
6. Alice encrypts the set received from Bob in Step 5.
7. Alice compares the two sets, i.e. the one received in Steps 4 and the one produced in Step 6. The elements from one set that have a match in the other form the intersection of the sets, hence the count of the common elements is the intersection size.
8. Alice may inform Bob about the results.

Notice that in the above protocol Alice and Bob are unable to find out which elements are common. This protocol provides them only with the information about the size of the intersection, but it can be easily transformed into private intersection protocol. If the parties would like to identify which elements are common between them, then Bob would have to pair up inputs from Alice, with his encryption of these inputs in Step 4. This way, in Step 7 Alice could find out which of the elements in her set match Bob’s elements. Agrawal et. al. provide strong proofs of correctness and security of the above protocols.

Both private intersection size and private intersection protocols can be used to create proximity based social networking or dating application described in Scenario 1. Avatars could scan surroundings for other avatars, and then engage in the above protocols to find out how many things their users have in common or what exactly they have in common. No other information would be revealed. Consequently, this is a complete solution to the problems presented by Scenario 1. On the contrary, these protocols have limited benefits to Scenario 2, since they can help the security services to search for their suspect in the ISP's database, but they don't provide any information retrieval mechanism. Thus, the security services would be unable to gather any data about their suspects.

3.4 Three-Pass (3Pass) protocol

Before operation of other private dataset protocols can be described 3Pass protocol primitive needs to be explained. This protocol was intended so that two parties could share a secret without exchanging any private or public key. Thus, the protocol was aimed at providing an alternative to public-key encryption and DH-like key negotiation protocols. The 3Pass, though, have never been widely used in this way since, on its own, it is susceptible to the man-in-the-middle attacks (Shamir, Rivest, & Adleman, 1981) and is less efficient then Rivest-Shamir-Adleman (RSA), a common choice public-key algorithm (Rivest, Shamir, & Adleman, 1978).



Alice's input: secret message M ; encryption key E_A ; decryption key D_A .
Bob's input: encryption key E_B ; decryption key D_B .

Figure 3.3 Operation of the 3Pass protocol

The 3Pass protocol can be described using the following physical analogy:

1. Alice places a secret message in a box and locks it with a padlock,
2. The box is sent to Bob, who adds his padlock to the latch, and sent the box back to Alice,
3. Alice removes her padlock and passes the box back to Bob,
4. Bob removes his padlock, and this enables him to read the message inside the box.

A more formal notation of this protocol is shown in Figure 3.3. Using this protocol Alice and Bob can share a secret without sharing a key. This algorithm needs the underlying encryption to be commutative. The CE protocols allow for a ciphertext created using a number of keys to be decrypted with application of the decryption keys in an arbitrary order. This is not the case in non-commutative ciphers and a ciphertext message created using non-commutative protocols can only be decrypted in a reverse order to the encryption process that produced it, otherwise the decryption output would be gibberish, and not the plaintext message.

3.5 *1-out-of-n* Oblivious Transfer (OT)

To OT protocols allow a client, often referred to as a *chooser*, to retrieve a record from a number of records hosted on a server, usually called a *sender*, in a way that the *chooser* gets only the record requested and the *sender* learns nothing apart from the fact that the *chooser* retrieved a record (Schneier, 1996). Generally, the *chooser* needs to provide the *sender* with an index *i*, of the record that needs to be retrieved. This selection is often random or based on publicly available catalogue of items held by the *sender*. Thus, an OT algorithm could be used for a purchase of digital goods in a way that the provider cannot identify the product purchased by the buyer, while the protocol guarantees that the buyer can only get access to one product. This is possible, since in this scenario the catalogue of products can be made public. Such solution was proposed by Bao and Deng in 2001, who designed an OT based on the 3Pass protocol. Many other efficient *1-out-of-n* OT algorithms exist, such as those described in Lipmaa (2003) and Aiello, Ishai, & Reingold (2001). On the contrary, OT on its own is not a viable solution to the privacy concerns described in the scenarios provided in Section 1. It has no use in Scenario 1, since the avatars are looking to compare data and not to retrieve a record. Also, it cannot be used in Scenario 2. Although, the OT primitive allows a retrieval of a record, there is no way for the security services to provide the index *i* of the ISP's dataset that would relate to their suspect.

The OT primitive needs to be combined with a form of PEqT, such as *1-to-n* PEqT or private intersection protocol in order to allow the *chooser* to identify an index of the interesting record in the data-holder's set. Such a composite solution was presented by Kwecka, Buchanan, Spiers, & Saliou (2008). The authors proposed a system that would be suitable, for security services from Scenario 2, to acquire a record from the Bob's ISP, maintaining the privacy of the suspect, other record, in the ISP's database, and also privacy of the investigation itself. However, this solution is not scalable, as for each suspect police would have to run a separate *1-to-n* PEqT and OT enquiries.

3.6 Private equijoin

The previous paragraph suggested that it is possible to create a viable solution to the problem presented in Scenario 2 from a mixture of PEqT and OT protocols. Such solutions usually do not blend well, and usually each requires its own computationally expensive preparation phase. However, Agrawal et. al. (2003) noticed that if CE is used for both PEqT and OT, an efficient protocol for private data acquisition can be formed. Thus, when the private intersection protocol described above is combined with the 3Pass primitive an ideal solution to Scenario 2 is created. The security services can retrieve a number of records relating their suspect with a single run of a protocol. Resulting protocol is referred to as a private equijoin protocol and is an extension of *k-out-of-n* OT protocol, where *k* records can to be retrieved over a single OT run. In such protocol the *chooser*, learns some extra information from the *sender* for the elements which form the intersection between the sets held by the two parties.

4. Implementing commutative cryptosystems

At first the only cryptographically strong commutative protocols were the ones based on modular exponentiation (Shamir, 1980). However, most theoretic applications of CE could benefit from being based on more efficient symmetric algorithms. Unfortunately, with the exception of the stream ciphers, which are unusable for dataset operations, all the current implementations of CE are still based on modular exponentiation, quite an inefficient process, often used in asymmetric encryption. This section explains the choice of CE to support the protocols described in Section 3 and provides description of implementation of the most viable solution.

4.1 Commutative algorithms

In his PhD thesis Weis proposed a method of building a commutative cryptosystem from an arbitrary cryptosystem that supports homomorphic multiplication of ciphertexts (Weis, 2006). Weis provides an example based on EG protocol that uses the original EG key specification and generation. However, his algorithm displays only one of two commutative properties that are required for most protocols discussed in Section 3, since **Equation 2** is not true for this algorithm. Consequently, it can be used in 3Pass-like protocols but not in the protocols that allow comparison of values, through comparison of ciphertexts.

The only protocol viable to perform PEqT, private equijoin and other protocols needed for the scenarios described in Section 1 is in fact an unpopular PH encryption scheme that is based on exponentiation modulo a large prime *p* (Pohlig & Hellman, 1978). This algorithm is commutative for keys with common prime *p* (Shamir, 1980; Shamir, Rivest, & Adleman, 1981). Interestingly, the PH algorithm, although similar to RSA algorithm, does not support public-key operations, since in PH the decryption key can be easily calculated from the encryption key. Nor it is a symmetric algorithm, as two different keys are used for encryption and decryption. Therefore, PH can be considered as an asymmetric private-key encryption algorithm. This is not a popular type of cryptographic protocol, and this explains why PH cannot be found in any openly available cryptographic suite.

4.2 Commutative PH algorithm

The PH scheme is similar to RSA. Equation 3 and Equation 4 show PH encryption and decryption functions respectively, where M stands for the plaintext message and C stands for the resulting ciphertext:

Equation 3: $C = M^{e} \bmod p$

Equation 4: $M = C^{d} \bmod p$

Both operations are performed modulo a large prime *p*, and different keys, exponents, are used for encryption (exponent *e*) and decryption (exponent *d*) in this algorithm. The encryption exponent *e* is randomly chosen so that:

Equation 5 $1 < e < p - 1$

Then, *e* is used to calculate the decryption exponent:

Equation 6 $de \equiv 1 \bmod (p-1) \Leftrightarrow d = e^{-1} \bmod (p-1)$

Unlike RSA, it is easy to calculate the decryption key form the encryption key, thus, *e* must remain secret.

4.3 Security of PH algorithm

In PH both, encryption and decryption, keys should be kept secret. However, there is no harm in making the large prime *p* public, while this is required in order to make PH scheme commutative. An adversary with the knowledge of the ciphertext C and the prime *p* would need to solve the following hard problem to break the commutative PH protocol (Schneier, 1996):

Equation 7 $e = \log_{P} C \bmod p$

Just like RSA, the ciphertext created using the PH protocol may leak some information about the input plaintext message. Therefore, this algorithm is mainly suitable for situations where the input is formed from random data, such as secret encryption or hashed signatures. It is also advised to use padding schemes together with any RSA implementations (Kaliski & Rivest, 2003), consequently implementations of PH should employ a padding scheme as well. Therefore, if PH algorithm is to be employed in implementing protocols described in Section 3, then the input data can be hashed to form blocks of data that would not require padding (Agrawal, Evfimievski, & Srikant, 2003).

These issues show that implementation of the PH should not be treated lightly. The operations involved in the protocol itself can be easily performed using any package enabling operations on large integers. However, there are other components to the protocol, such as hashing and random number generation. Many tried-and-tested cryptographic suites exist to support these operations. Thus, for security reasons, it is recommended to build the PH algorithm into an existing cryptographic suite that allows modification of its source code. Since, PH is similar to RSA, in operation, an RSA implementation can be easily altered to support PH algorithm and the transformations required are shown next.

4.4 RSA based implementation of PH

Parties wanting to employ commutative behaviour of PH must use a common *p*. Consequently, *p* can be decided upon during initialisation phase by the parties involved in the protocol, or it can be written into protocol specification. It is advisable to use a strong prime for PH, just like it is in case of DH algorithm (Agrawal, Evfimievski, & Srikant, 2003; Ferguson & Schneier, 2003; Silverman & Rivest, 2001). A prime *p* is strong if there is such a prime *q* that the following is true:

Equation 8 $p = 2q + 1$

Most cryptographic suites have build-in methods for generating strong primes, often hidden in DH key generator code, this methods should be used for obtaining *p* (and *q*) for this PH implementation. Once *p* is agreed upon the PH key generation may begin.

As previously mentioned RSA implementations can be used to implement PH algorithm. The main difference between the two is that, in RSA, encryption and decryption are performed modulo a product of two large primes *p* and *q*, whereas in PH these operations are performed modulo a prime *p*. Thus, there are no changes required to the actual RSA engine, since its main task is to perform exponentiation modulo a number. However, there is a slight difference in the key generation phases of these two algorithms. It has more to do with the way most RSA key generators have been implemented, rather than the differences in algorithms themselves. Some RSA implementations available in the public domain do not choose *e* randomly, but use values 3 or 17 just as long as *d* generated from *e*, *p* and *q* is large enough for the RSA Problem to hold. Using small exponent *e* speeds up key generation and encryption, but it could make RSA implementations employing this technique weaker (Kaliski & Rivest, 2003). In PH, knowing the encryption exponent *e* and the prime *p* is enough to calculate the decryption key *d*. Since *p* is public, *e* needs to be kept private. Consequently, using small values for *e* would break this cryptosystem, and so an ordinary RSA key generator is unsuitable to produce valid PH keys.

The RSA key generator needs to be altered so that it accepts *p* (and *q*) as an input, and generates random *e* and then calculates *d*. The exponent *e* should be randomly chosen odd integer from the range defined in Equation 5 (Bao & Deng, 2001) and the Greatest Common Divisor (GCD) of *e* and $(p-1)$ should be one, i.e. *e* needs to be relatively prime with $(p-1)$. Once the appropriate *e* is generated the decryption exponent *d* can be calculated according to Equation 6. This is usually easily done in cryptographic suites, since methods for calculating modulo inverse of a number are used in RSA key generation. After such modifications the PH protocol can be invoked just like an ordinary RSA algorithm would be invoked.

4.5 Licensing

Both PH and RSA were patented in early 80's, consequently these patents have now expired and the above solution can be used without a licence (Schneier, 1996). However, attention should be given to the licence of the cryptographic suite that is being modified with PH scheme, since licences of some open-source packages prohibit source-code modification.

5. Conclusions

This paper has reviewed and analysed applications of PETs to privacy problems in matching operations on datasets. Section 1 provided usage scenarios for such operations. The findings show that protocols based on commutative cryptosystems are most viable private matching algorithms, when efficiency is the key consideration. This is the case when more than one value is to be compared, since other efficient privacy enhancing algorithms to perform a singe value equality test exist.

To answer the issues raised by Scenario 1. Private intersection and private intersection size protocols described in Section 3 can be used in pervasive computing applications where avatars scan surrounding of their owner for services and other avatars that match certain selection criteria that should remind secret in order to maintain the privacy of their owner. Thus, the future users of proximity based social networking and dating application can be more protected against various social engineering attacks and targeted advertising campaigns if these protocols are deployed.

Scenario 2 describes a case where secret service's investigation can be jeopardised together with suspect's privacy during a lawful communication data acquisition process in the UK. In Section 3 it was shown that these concerns can be mitigated by a composite system build from different 1-to-n PEqT and OT protocols. However, such solutions are usually inefficient, since both 1-to-n PEqT and OT protocols require certain preparation phase before the protocols can run. Also, these processes are often expensive in terms of computation needed. Therefore, a complete scheme based on commutative cryptography, i.e. private equijoin protocol by Agrawal et. al. (2003), is suggested for performing symmetric private information retrieval tasks.

Protocols described in Section 3 can form a key part of a number of PETs, where there is not trusted third party, or such party is trusted only for authentication purposes but not to handle sensitive data. These protocols are most efficient when based on commutative cryptosystems. However, CE cannot be performed using the openly available cryptographic suites, and thus, fast development strategy for forming secure implementations of CE is required. This paper presents a method of modifying common implementations of the RSA scheme, so that PH algorithm with commutative properties can be achieved as a part of any cryptographic suite that allows modification of the source-code. The reason for implementing PH algorithm as a part of a cryptographic suite is that it should always be used in conjunction with hashing or padding schemes and well tested random number generators. Such solution allows building a complete infrastructure for private matching in most secure and cost effective way.

This research focused on private matching dataset operations and showed that there is a large field of use for the presented algorithms. However, they have certain limitations. In all the protocols described here the match must be perfect. Thus, it is not possible to match fuzzy numbers and patterns. For example the techniques described could not be used to create a system for private DNA searches, since the object of the comparison in DNS analysis is a pattern and not a set value. Currently there is no way to compare patterns or fuzzy numbers in a private manner, and thus, this is the area of the further research in private matching algorithms.

6. Acknowledgments

The authors would like to thank Dr Sandra Cairncross, Dean of Faculty of Engineering, Computing and Creative Industries, for making funding available for research, as well as Sally Smith, Head of School, for assigning grants to this research program.

7. References

Agrawal, R., Evfimievski, A., & Srikant, R. (2003). *Information sharing across private databases.* Paper presented at the ACM SIGMOD international conference on Management of data, San Diego, California.

Aiello, B., Ishai, Y., & Reingold, O. (2001). Priced Oblivious Transfer: How to Sell Digital Goods. In B. Pfitzmann (Ed.), *Advances in Cryptology — EUROCRYPT 2001* (Vol. 2045, pp. 119-135): Springer-Verlag.

Anderson, R. J. (2008). *Security Engineering: A guide to building dependable distributed systems* (2nd ed.). Indianapolis, In: Wiley Publishing, Inc.

Balz, D., & Deane, C. (2006). Differing Views on Terrorism*. The Washington Post*

Bao, F., & Deng, R. (2001). Privacy Protection for Transactions of Digital Goods. In *Information and Communications Security* (pp. 202-213).

Fagin, R., Naor, M., & Winkler, P. (1996). Comparing information without leaking it. *Commun. ACM, 39*(5), 77-85.

Ferguson, N., & Schneier, B. (2003). *Practical Cryptography*. Indianapolis, Indiana: Wiley Publishing Inc.

Frikken, K. B., & Atallah, M. J. (2003). *Privacy preserving electronic surveillance.* Paper presented at the Proceedings of the 2003 ACM workshop on Privacy in the electronic society, Washington, DC.

Kaliski, B., & Rivest, R. (2003). RSA Problem. *ACM SIGKDD Explorations*

Kwecka, Z., Buchanan, W., Spiers, D., & Saliou, L. (2008, 30 June – 1 July). *Validation of 1-N OT Algorithms in Privacy-Preserving Investigations.* Paper presented at the 7th European Conference on Information Warfare and Security, University of Plymouth.

Lipmaa, H. (2003). Verifiable Homomorphic Oblivious Transfer and Private Equality Test. In C. S. Laih (Ed.), *Advances on Cryptology - ASIACRYPT 2003* (Vol. 2894 pp. 416-433). Taipei, Taiwan: Springer-Verlag.

RAPID. (2009). Telecoms: Commission launches case against UK over privacy and personal data protection [Electronic Version]. *Europa Press Releases RAPID*. Retrieved 15 April 2009 from http://europa.eu/rapid/pressReleasesAction.do?reference=IP/09/570&format=HTML&aged=0&language=EN&guiLanguage=en.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM, 21*(2), 120-126.

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*: John Wiley & Sons, Inc.

Shamir, A. (1979). How to share a secret. *Commun. ACM, 22*(11), 612-613.

Shamir, A. (1980). *On the Power of Commutativity in Cryptography.* Paper presented at the Proceedings of the 7th Colloquium on Automata, Languages and Programming, London.

Shamir, A., Rivest, R. L., & Adleman, L. (1981). Mental Poker. In D. A. Klarner (Ed.), *The Mathematical Gardner* (pp. 37-43). Boston, MA: Prindle, Weber & Schmidt.

Silverman, R., & Rivest, R. (2001). Are 'Strong' Primes Needed for RSA.

Stross, R. (2007). How to Lose Your Job on Your Own Time [Electronic Version]. *The New York Times*. Retrieved 30 Dec 2007 from http://www.nytimes.com/2007/12/30/business/30digi.html.

Swire, P., & Steinfeld, L. (2002). *Security and privacy after September 11: the health care example.* Paper presented at the 12th annual conference on Computers, freedom and privacy, San Francisco, California.

Waters, D. (2009). Amazon blocks Phorm adverts scan [Electronic Version]. *BBC News*. Retrieved 15 April 2009 from http://news.bbc.co.uk/2/hi/technology/7999635.stm.

Weis, S. A. (2006). *New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing.* Unpublished PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.