Efficiency Improvements of the Private Message Service

Oliver Berthold, Sebastian Clauß, Stefan Köpsell, and Andreas Pfitzmann

Technische Universität Dresden
Institute for System Architecture
D-01062 Dresden, Germany
{ob2,sc2,sk13,pfitza}@inf.tu-dresden.de

Abstract. Based on the private message service described in [4] we show efficiency improvements of that private message service in the computational setting. Regarding an attacker which may control all but one of the queried servers we describe a private message service with a total communication complexity of blinded read between client and private message service of n bit upstream and k bit downstream, where n denotes the number of cells in the database and k the size of one cell. Apart from a registration mechanism, the communication complexity between client and service is independent of the number of queried servers. Our improvement of the private message service is not only extremely efficient in terms of communication, but also in terms of computation. Further we describe how to use the message service in case of messages which are addressed using visible implicit addresses. After that we prove that at least parts of messages which are addressed using invisible implicit addresses must be broadcasted.
We generalize the message service to operations in $\mathbb{Z}\!\!\mathbb{Z}_N$ ($N \geq 2$) and prove the security of blinded read.

1 Introduction

Techniques to gain privacy in computer networks become more and more important. One aspect of privacy in networks is fetching information privately. More concrete that means that only the client who wants to fetch that information knows which information he fetches. Not even the service, from which the information is fetched shall be able to know which information is fetched. [4] describes a message service which has this property.

Section 2 refers to previous papers related to private message services. In Sect. 3 we briefly describe the private message service of [4]. In Sect. 4 we show efficiency improvements of that message service in the computational setting.

After that we describe how the message service can be used to send messages to the intended recipients using visible resp. invisible implicit addresses. Further we prove that at least parts of messages which are addressed using invisible implicit addresses must be broadcasted.

In the last section we describe a generalized private message service. In that message service the operation XOR of the message service described in [4] is replaced by addition resp. subtraction in $\mathbb{Z}_N$ ($N \geq 2$).

2 Related Work

The private message service that uses multiple servers was first introduced in [4, 5]. Our work is based on that papers. In [1–3, 10] there are also shown efficiency improvements of the private message service with multiple servers. The attacker model used in [1, 3, 10] is weaker than the one we use. In those papers the client is able to privately retrieve information, so that each single server gains no information on the identity of the item retrieved. Our attacker may control all but one of the servers which the client queries. In [2] there is described a scheme regarding the attacker model we use in the information theoretical setting. The efficiency improvements we show are related to the computational setting. But the simple complexity related assumption we make is the existence of a secure pseudo-random number generator. In [3] Chor and Gilboa show a two-server approach, that is also based on the existence of pseudo-random number generators, with a communication complexity of $O(n^{\varepsilon})$, $\varepsilon > 0$. Our scheme is more general because it is not restricted to two servers. In contrast to their scheme our scheme uses minimal downstream communication, so it is more efficient than their scheme for databases with not so many, but large cells.

[6] describes how private information retrieval schemes can be adapted to the commodity-based cryptography model. In that model additional servers, called *commodity servers*, sell "security commodities" to their clients which can be later utilized by the clients to perform various cryptographic tasks more cheaply. Our efficiency improvements do not relate to this model. Although our improved scheme has the characteristics of a client-server model, it does not involve commodity servers. The attacker model (in multi-server schemes) in [6] is weaker than ours, because the privacy of schemes in that model bases on trust in at least one of the commodity servers in addition to at least one database server.

Another type of private message service was shown in [7–9]. There the message service only consists of one single database. The price to be paid for that is that the single database has to do quite heavy computations. Whereas in our scheme (as in most multi-server schemes) the database only needs to perform a small amount of computation. In contrast to our scheme, single-server schemes rely on more restrictive security assumptions than our scheme.

3 Functionality of the Private Message Service

First we briefly describe the model of the private message service of [4, 5]. The private message service consists of m servers, $m > 1$. Each server contains an identical copy of the message service' database. The database consists of n cells of capacity k bits. Each cell may hold one message.

When a client wants to fetch a message from the private message service, he chooses s servers ($1 < s \leq m$) which he wants to query. Then he creates $s-1$ random vectors $V_1, ..., V_{s-1}$ of length n bit. Each bit of a vector corresponds to one cell at the server's database. Now he creates another vector V_s by XORing the vectors $V_1, ..., V_{s-1}$. To fetch the message N_t from cell t ($0 < t \leq n$), the client flips the bit at $V_s[t]$.

The client encrypts these vectors and sends to each of the s servers the corresponding vector. Encryption is necessary to protect against attackers who are listening to all the communication between client and private message service. If an attacker gets all vectors $V_1, ..., V_s$, he knows which cell the client is fetching.

Each server XORs all these cells of its database, where the corresponding bit in the vector is set to 1. Now each server S_i ($0 < i \leq s$), which received vector V_i sends its result R_i back to the client. The results must also be encrypted, because an attacker who gets the results of all queried servers is able to calculate the contents of the cell being read.

Finally the client XORs all the results $R_1, ..., R_s$ of the servers and so he gets the message N_t.

Figure 1 shows an example of a private message service which consists of 4 servers and a database of 4 cells. The client queries three of the servers.

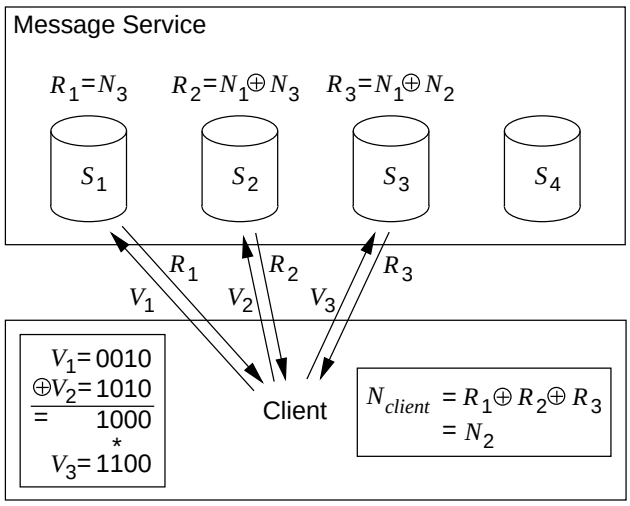

Fig. 1. Reading cell 2 from a private message service consisting of 4 servers and 4 cells. The client queries three of the servers.

In [4] it was shown that an attacker, which has access to no more than $s-1$ of the vectors, gains no information about which cell the client is reading.

The total communication complexity D of the communication between the client and the private message service that is needed to read one cell is as follows[1]:

$$D = D_{client \to service} + D_{service \to client} = s(n+k)\,\text{bit} \tag{1}$$

The communication complexity in dependence of the number of queried servers s is $O(s)$.

4 Efficiency Improvements in the Computational Setting

4.1 Communication from the Client to the Private Message Service

In [4] the vectors $V_1, ..., V_{s-1}$ are created randomly. That leads to an information-theoretical private information retrieval scheme. In the computational setting the vectors $V_1, ..., V_{s-1}$ may be created by pseudo-random number generators (PRNGs). The vector V_s is created by XORing the vectors $V_1, ..., V_{s-1}$ and flipping the bit $V_s[t]$, according to the model described in Sect. 3.

Now the client sends to $s-1$ servers only the random seeds of length p bits, which he used to setup the PRNGs. The servers create their vectors of length n bit using the PRNG. Only vector V_s must be fully transmitted.

The proof of the scheme of [4] holds, if each bit in the s vectors is set to 1 with probability $\frac{1}{2}$. In the computational setting this holds for vectors created by a PRNG. So the proof of [4] holds for the scheme described here, too.

In this scheme the amount of data transmitted from the client to the private message service is as follows:

$$D_{client \to service} = n + p(s-1)\,\text{bit} \tag{2}$$

It can further be decreased, if a client registers at first with the private message service. A registration consists of the following steps:

1. The client chooses $s-1$ servers, which generate their vectors by using a PRNG.
2. For each of the $s-1$ servers the client generates a random seed, which he exchanges with that server.
3. The client selects one server, which is not one of the servers of Step 1., to which he later sends the vector V_s.

So the total communication complexity of the registration mechanism is

$$D_{registration} = (s-1)p + u(s)\,\text{bit} \tag{3}$$

where u denotes the number of bits used to tell the message service, which server gets which random seed and which server will later get the vector V_s.

[1] We do not consider data that will be needed by the client and the private message service to address each other

If the client wants to fetch a message, he only needs to send V_s to the selected server. This server informs the other $s-1$ servers, that the client wants to read a message. These servers create their vectors using the PRNG. On the first query each server initializes its PRNG with the appropriate random seed.

If we do not consider the registration mechanism, the communication complexity of the communication from the client to the private message service in this scheme only depends on the number of cells n of the database. It does not depend on the number of servers queried.

4.2 Communication from the Private Message Service to the Client

As stated in Sect. 3, each server has to encrypt its reply. If the encryption algorithm that the servers use is chosen favourably, it is possible that the private message service XORs the replies of the servers without revealing information about the cell being read. Now the private message service sends only one message of length k bit to the client. So the communication complexity of the communication from the private message service to the client is independent of the number of servers queried.

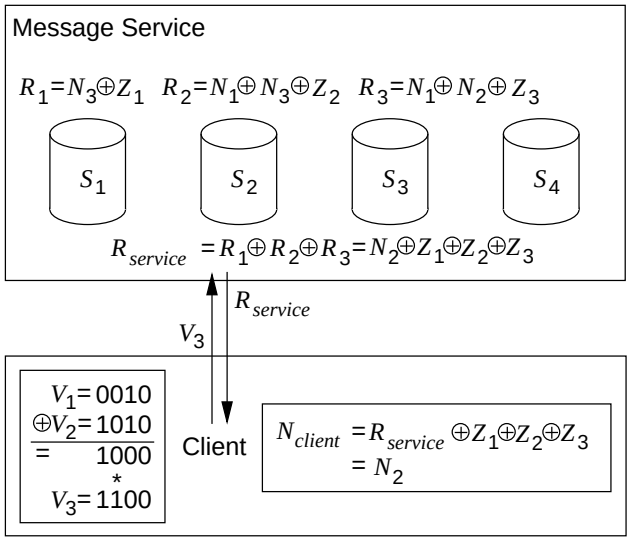

Fig. 2. Efficiency improvements described in Sect. 4.1 and 4.2, using the example of Fig. 1.

To gain this efficiency improvement each server encrypts its reply with a pseudo-one-time-pad. The client exchanges the keys for this encryption algorithm when he registers with the private message service. If the client wants to fetch a message, he first sends his query to the private message service. The

queried servers create their replies as stated in Sect. 3. Now one of the servers collects the replies of the other queried servers. The replies are XORed and the result is sent back to the client. The client receives the contents of the cell he requested, encrypted with the pseudo-one-time-pads of the queried servers. Finally the client decrypts this reply by sequentially decrypting it with the pseudo-one-time-pads of all queried servers.

The example shown in Fig. 2 illustrates the efficiency improvements described in Sect. 4.1 and 4.2. Z_i denotes the encryption pad that server S_i uses to encrypt its reply with a pseudo-one-time-pad.

Again, if we do not consider the registration mechanism, the total communication complexity D between the client and the private message service for fetching one message is as follows:

$$D = n + k \,\text{bit} \tag{4}$$

This communication complexity is independent of the number of servers queried.

5 Efficiently Fetching Messages from More than One Cell

Here we use a message service, where old messages are not explicitly deleted, but they are overwritten by new ones. Often a client needs to query more than one cell frequently to get new messages, which arrived in these cells.

If the client has to do one query for each single cell, the communication is independent of the number of messages received. If there are only a few cells that actually hold messages, the communication cost can be very high compared to the messages received.

We describe a technique, which may have significantly lower communication cost depending on the number of messages fetched. The technique is based on the following ideas:

- If a client flips more than one bit in the vector V_s, he fetches the XOR of the messages that are located in the cells corresponding to the flipped bits.
- If a message contains a redundancy predicate, a client is able to check whether he received one and only one message or a XOR of more than one message.

Thereby it is possible that the XOR of two or more messages gives exactly one message where the check of the redundancy predicate is successful. If the size of the redundancy predicate is chosen sufficiently large, the probability of this case is extremely low. So we do not consider this case in the described algorithm.

The technique can be described using a recursive algorithm. First, we describe a procedure $REC(R, Q)$ which is called recursively. Q denotes a set of cells of the database. R denotes the response of the message service when it is queried for the cells contained in Q.

Procedure $REC(R, Q)$**:**

Check (using the redundancy predicate), whether the response R of the private message service contains no message, one message, or more than one message.

1. The response R contains no message:
   In this case no cell of Q contains a message.
   End REC.
2. The response R contains one and only one message:
   Save the message.
   End REC.
3. The response R contains the XOR of more than one message:
   Create a set of cells Q_1, which contains $\lceil \frac{1}{2}|Q| \rceil$ of the cells of Q.
   Query the cells of Q_1 at once (using the described idea). The received response is called R_1.
   Call $REC(R_1, Q_1)$.
   Call $REC(R \text{ XOR } R_1, Q - Q_1)$.
   End REC.

End REC**.**

The procedure $QUERY(Q)$ that is used to query a set of cells Q consists of the following steps:

Procedure $QUERY(Q)$**:**

1. Query the cells of Q at once (using the described idea). The received response R consists of the XOR of the messages that are contained in the queried cells.
2. Call $REC(R, Q)$.

End $QUERY$**.**

Figure 3 shows an example of an execution of this algorithm. Depending on the number of messages located in Q, the algorithm needs different numbers of queries. If the message service knows which query belongs to which execution of the algorithm, the service would be able to gain information about the queried cells from the different numbers of queries. Using the data of the queries, the service cannot decide which query belongs to which execution of the algorithm, but it could do so by timing attacks, i.e. a close sequence of queries from one client may belong to one execution of the algorithm with significant probability. We have two solutions to prevent timing attacks:

- The client can use dummy queries to ensure that his sequence of queries does not vary in time. So the queries do not depend on the number of messages requested.
- The queries can be done anonymously, possibly using a Mix-network. So the service does not know which query comes from which user. Thus the service cannot decide, which queries belong to one execution of the algorithm.

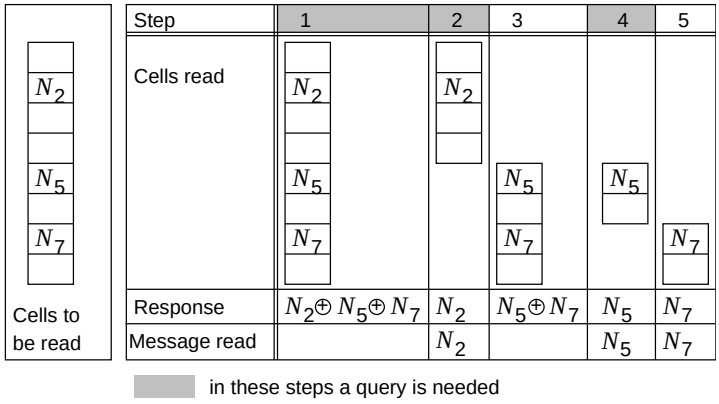

Fig. 3. Example of the algorithm for efficiently fetching messages from groups of cells. In the example three messages are fetched out of eight cells.

Let q be the number of cells queried and r the number of messages, which the queried cells hold. The described algorithm needs a minimum of r queries to fetch these messages. If the messages are spread over the q queried cells in a unfavourable way, additional queries are needed. Such an unfavourable case occurs, if in procedure $REC(R, Q)$ the set Q is divided into Q_1 and Q_2 in such a way, that all cells of Q which actually hold a message are located in one of the subsets whereas no cell holding a message is located in the other.

If there is no message in the queried cells ($r = 0$), one query is needed to prove this. If $r > 0$, a maximum of $\lceil \log_2(q) + 1 \rceil$ queries is needed to get the first message. To fetch all the messages a maximum of q queries is needed. Consequently the worst case is equal to doing one query for each single cell.

6 Adressing Clients Using the Private Message Service

If a sender wants to use the private message service to send messages to a given recipient, the sender needs to address the message to the recipient. Usually the sender will have no information about the recipient that could be used to identify that recipient. In the ideal case, only the recipient itself is able to check, whether a message is addressed to him or not. Addresses that comply to this description are called implicit addresses.

In the simplest case such an address is a random number, that is appended to the original message. Only sender and recipient know that random number. Using this information, the recipient is able to check which messages of a given set of messages are addressed to him. This type of implicit address is called visible implicit address[2] . To avoid linkability of messages that are sent using the same

[2] This type of address corresponds to the "labels" used in [4].

visible implicit address, such an address may only be used once. A visible implicit address cannot be stored in public directories, because only sender and recipient should know this address. Hence it cannot be used for a first contact between sender and recipient.

A fundamental drawback of an implicit address is the fact that the recipient must check all the messages which are potentially dedicated to him. So an attacker, who controls the network, cannot discover the relationship between sender and recipient.

Using the private message service, this drawback can be avoided in the case of visible implicit addresses. Some of the bits of the visible implicit address are used as a pointer to a cell of the database of the private message service. Now the recipient only needs to query the cell the implicit address points to. The private message service enables a recipient to privately retrieve the contents of cells. So an adversary who controls the network and even some of the servers of the private message service gets no information about the recipient of a message.

If a recipient uses more than one visible implicit address at one time, he may efficiently fetch the contents of all corresponding cells using the algorithm described in Sect. 5.

6.1 Invisible Implicit Addresses

Invisible implicit addresses are another type of implicit addresses. These addresses can be stored in public directories in connection with some information of the recipient (e.g. memberships, interests ...) without allowing an adversary to link messages that are sent using this type of address. So it can be used to establish a first contact between sender and recipient. Invisible implicit addresses have the following properties:

1. Using a public information, a message can be addressed to an anonymous resp. pseudonymous recipient. With a high probability third parties cannot link the addressed message to the public information, which was used to address the message.
2. With a high probability for each set of messages it cannot be detected, whether these messages are addressed to the same recipient or not. (Only the recipient itself can do that, if one or more messages are addressed to him.)

If the messages shall not be linkable to previous messages sent to the same recipient, the sender has to use invisible implicit addresses.

In Sect. 6.2 we describe a technique that realizes invisible implicit addresses using public key cryptography. Further we give a general definition of invisible implicit addresses. In Sect. 6.3 we prove that invisible implicit addresses require a broadcast of at least parts of every message to all recipients. In Sect. 6.4 we show a technique which reduces the total communication complexity when using invisible implicit addresses in case of the private message service.

6.2 Invisible Implicit Addresses Using Public Key Cryptography

Invisible implicit addresses can be created using anonymous indeterministic encryption functions. An encryption scheme is defined as anonymous, if no third party is able to know, whether a given cipertext is created using a given public key or not. The indeterministic behaviour is achieved, if a message is encrypted together with a random number. So if a message is encrypted more than once, each two ciphertexts differ with a high probability.

These properties can be used to create invisible implicit addresses. First the recipient creates a key pair of a public key encryption algorithm. Then he stores the public key together with other information in a public directory. Now the public key can be used as a pseudonym of the recipient. If a sender wants to send a message to a recipient, he encrypts a commonly agreed value using the public key that he got from the public directory. A third party is not able to check, whether two messages are sent to the same recipient, because of the indeterminism of the encryption algorithm. If the right recipient decrypts the message using his private key, he gets the commonly agreed value. So he knows that the message was sent to him. If another recipient decrypts the message (using another private key), he only gets a random number that differs from the commonly agreed value with a high probability.

This technique is very costly to the recipient. He must get every message that was sent, and he must apply a costly cryptographic operation to every message.

Now we give a general definition of invisible implicit addressing:

Definition 1. *Invisible implicit addressing system.*

1. *Each recipient of the system generates a key pair (p_k, s_k), where third parties cannot derive the secret key s_k from the public key p_k, and publishes the public key p_k.*
2. *There exists a public addressing algorithm $E(p_k, z)$, where z denotes a random number $0 \leq z < z_{max}$ with $z, z_{max} \in \mathbb{Z}$. E generates addresses $\alpha = E(p_k, z)$.*
3. *There exists a secret detection algorithm $D(s_k, \alpha)$, which decides, whether an address α belongs to a secret key s_k or not.*

The scheme is secure, if only the holder h of the secret key s_{k_h} can decide, whether a message is addressed to him or not. The security of the system is defined as follows:

1. *Generate two keypairs (p_{k_1}, s_{k_1}), (p_{k_2}, s_{k_2}) at random and give the public keys p_{k_1} and p_{k_2} to an adversary.*
2. *Let an oracle choose a random z and let it produce an address $\alpha = E(p_{k_b}, z)$ for a randomly choosen $b \in \{1, 2\}$, and give α to the adversary.*
3. *Let the adversary produce a guess b' for b. The scheme is secure, if the probability $P(b = b') = \frac{1}{2} \pm \varepsilon$, where ε is sufficiently small.*

Definition 2. *A message consists of two parts:*

- *An address α and*
- *the message data d, which cannot be used to gain information about the recipient.*

6.3 Use of Invisible Implicit Addresses Requires Broadcast

For the proof shown in this section, we assume that all parties are restricted to the computational setting and that the sender of a message m keeps the assignment of the message m to the public key p_k, which he used to generate the address part α of the message m, private. So the network, which must decide to which recipients a message is to be sent, cannot gain that information from the sender.

Lemma 1. *When a message is addressed to a recipient using invisible implicit addressing, at least parts of that message must be broadcasted to all recipients of the system.*

Proof. by contradiction:
If there exists a message m of which no parts need to be broadcasted to all recipients, there exists a public function $G(\alpha)$ that can be applied to the part α of message m. This function G is able to determine for at least one recipient, that m is not addressed to this recipient[3].

Because messages are addressed using invisible implicit addresses, each message contains $\alpha = E(p_k, z)$. As of Def. 2 only α can be used to get information about the recipient of a message.

From this it follows that if G is able to determine for at least one recipient h, that the message m is not addressed to h, G is able to decide that the part α of the message m is not created by $E(p_{k_h}, z)$. This contradicts to Def. 1.

This proves that such a function G does not exist. So at least the part α of a message m must be broadcasted to all recipients of the system, because only the intended recipient h is able to check whether α was generated by $E(p_{k_h}, z)$. □

6.4 Private Message Service and Invisible Implicit Addresses

Normally the private message service is used to enable private reading of messages without the need to broadcast each message. We now give a basic approach how to use the private message service in connection with invisible implicit addresses to lower the bandwidth needed for broadcast. The approach is as follows:

Only part α of each message is broadcasted together with a number of a cell of the private message service where the complete message is stored. Using that part of a message a recipient is able to check whether the message is addressed to

[3] In this context the term "public function applied to a message" means that such a function may be executed by third parties that are neither sender nor recipient of that message.

him or not. If a recipient detects a message that is addressed to him, he fetches the corresponding cell of the private message service. This technique also needs broadcast, but – depending on the ratio $\frac{|\alpha|}{|m|}$ – it uses a much smaller bandwidth compared to broadcast of complete messages.

Another approach is to use a trusted third party which is able to check the messages and sends them to the intended recipients. Therefore this third party must be able to successfully execute the algorithm D and so it must know the secret keys of the recipients. This contradicts to Def. 1.

7 Generalization of the Private Message Service Using Operations in $\mathbb{Z}_N$ ($N \geq 2$)

In this section we describe a generalization of the private message service. In this generalized private message service the operation XOR of the private message service described before is replaced by addition resp. subtraction in $\mathbb{Z}_N$ ($N \geq 2$). All mathematical operations in the following description are operations in $\mathbb{Z}_N$.

The generalized private message service has the following structure:

Vectors and cells of databases consist of sequences of blocks of size $\log_2 N$ bit. Each block of a vector corresponds to one cell of the database.

Like in the model described before, $s-1$ vectors are chosen randomly. Vector V_s is created as follows:

$$V'_s = -(V_1 + V_2 + ... + V_{s-1})$$

V_s is created from V'_s by addition of 1 to the block that corresponds to the cell, which will be queried.

Now the vectors are sent to the appropriate servers of the private message service. For each cell C_j each server S_i does the following computation, in which it also uses its vector V_i:

$$R_i = \sum_{0<j\leq n} C_j \cdot V_i[j]$$

Every server sends its reply R_i back to the client. The client adds all the replies and so he gets the contents of the desired cell.

In Fig. 4 an example of a generalized private message service is shown, which operates in $\mathbb{Z}_{10}$.

Security of blinded read. We show a general proof of the security of blinded read of exactly one cell or a sum in $\mathbb{Z}_N$ of more than one cell. This proof is derived from the proof of [4].

Lemma 2. *If each of the blocks in the vectors $V_1, ..., V_{s-1}$ holds every value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$ then an attacker which has access to at most $s-1$ of the requests/responses associated with the vectors will gain no information about the cells being read.*

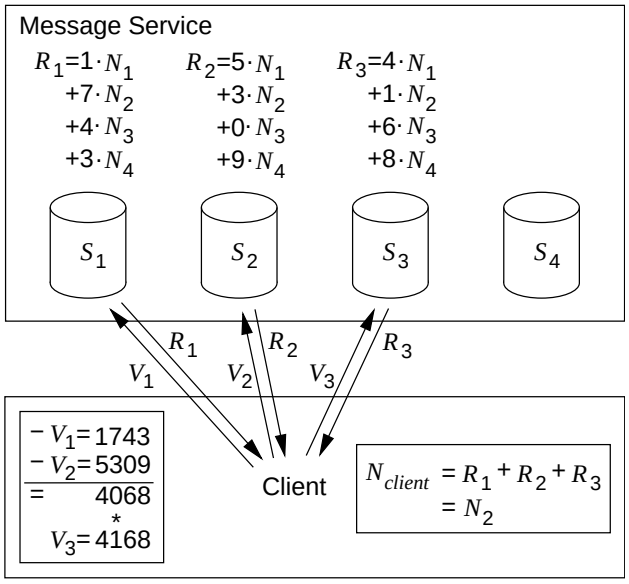

Fig. 4. Example of a generalized private message service.

Proof. Since the first $s-1$ vectors are chosen independently of the cells being read, an attacker will gain no information unless he has access to the vector V_s.

So we will assume that the attacker knows the vector V_s and $s-2$ of the other vectors. Let $V'_1, V'_2, ..., V'_{s-1}$ be the vectors that the attacker knows and V'' the vector that the attacker doesn't know.

Say that the client is reading a set of cells Q. We show for each block of the vectors that the attacker doesn't gain any information about the cells that the client is reading. C_j denotes the cell at position j.

Case 1: $C_j \in Q$. Since this is a cell being read, we know that

$$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j] + V''[j] = 1.$$

Since V'' holds each value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$ and

$$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j] = 1 - V''[j],$$

$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j]$ also holds each value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$.

Case 2: $C_j \notin Q$. Since this is not a cell being read, we know that

$$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j] + V''[j] = 0.$$

Since V'' holds each value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$ and

$$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j] = -V''[j],$$

$V'_1[j] + V'_2[j] + ... + V'_{s-1}[j]$ also holds each value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$.

Since, for each block, the value of $V'_1[j] + V'_2[j] + ... + V'_{s-1}[j]$ holds each value of $\mathbb{Z}_N$ with probability $\frac{1}{N}$ whether it corresponds to a cell being read or not, the attacker gains no information about which cells are being read. □

The efficiency improvements that are shown in Sect. 4 can also be adapted for the generalized private message service. The proof also holds for the generalized private message service with the efficiency improvements in the computational setting.

Let n be the number of cells in the database and k be the number of blocks of one cell. The total communication complexity D is as follows:

$$D = (n + k) \log_2 N \text{ bit} \tag{5}$$

It can be seen that the communication complexity increases for larger N. So private message services with $N > 2$ are less efficient than a private message service with $N = 2$.

8 Conclusions and Open Problems

We have shown improvements of the private message service which permit to efficiently fetch messages from one or more cells of the service' database. We have described how to gain efficiency improvements when using visible resp. invisible implicit addresses together with the private message service. We have generalized the private message service to operations in $\mathbb{Z}_N$ ($N \geq 2$). Thereby we have shown that the communication complexity increases for larger N.

Further research has to be done regarding the generalized private message service. It may be possible to use the communication overhead, which appears for larger N, to create more efficient protocols for fetching messages from more than one cell of the service' database.

References

1. A. Ambainis. Upper Bound on the Communication Complexity of Private Information Retrieval. 24th International Collocquium on Automata, Languages and Programming (ICALP), LNCS 1256, Springer-Verlag, Berlin 1997, 401-407
2. B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan. Private Information Retrieval. 36th Annual Symposium on Foundations of Computer Sience (FOCS) 1995, IEEE Computer Society, 1995, 41-50
3. B. Chor and N. Gilboa. Computationally Private Information Retrieval. 29th Symposium on Theory of Computing (STOC) 1997, ACM, New York 1997, 304-313

4. D. A. Cooper, K. P. Birman. Preserving Privacy in a Network of Mobile Computers. 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 26-38
5. D. A. Cooper, K. P. Birman. The design and implementation of a private message service for mobile computers. Wireless Networks 1, 1995, 297-309
6. G. Di Crescenzo, Y. Ishai, R. Ostrovsky. Universal Service-Providers for Private Information Retrieval. Journal of Cryptology 14, 2001, 37-74
7. O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Proc. 19th Annual ACM Symp. Theory Comp., 1987
8. O. Goldreich, R. Ostrovsky. Software protection and simulation by oblivious RAMs. JACM, 1996
9. R. Ostrovsky. Software protection and simulation on oblivious RAMs. M.I.T. Ph. D. Thesis in Computer Sience, June 1992. Preliminary version in Proc. 22nd Annual ACM Symp. Theory Comp., 1990
10. R. Ostrovsky, V. Shoup. Private Information Storage. 29th Symposium on Theory of Computing (STOC) 1997, ACM, New York 1997, 294-303
11. A. Pfitzmann, M. Waidner: Networks without user observability. Computers & Security 6/2, 1987, 158-166