Sender and Recipient Anonymous Communication without Public Key Cryptography

Hiroaki Kikuchi
Tokai University

Abstract
We propose a new idea to archive an anonymous communication network which provides both sender and recipient anonymities under the condition that the overall traffic may be analyzed. The proposed protocol, which is based on a secret function computation of addition using secret sharing schemes, is estimated with regards to bandwidth and time delay. We show that the bandwidth spend in computation of the proposed protocol is linear to the number of entities, but is limited within a fraction of the total bandwidth.

1 Introduction

We study an anonymous communication in the Internet. An anonymous channel is one of primitive technique, which is used in broad protocols such as electrical cash or secret voting. Especially, in web browsing, a senders' privacy against web page provider is coming to current issues, because the web contents providers wish to know their customer privacy information as much as possible. To this privacy issues, there have been a couple of approaches attempted:

- Anonymizer[1]
- Crowds[6]
- Onion Routing[2, 3, 4]

With the level of anonymity to be required, several approaches are feasible. In fact, other than these explicit anonymous channels, we are surrounded by variant anonymous communications.

1. A proxy server, which is intended to relay HTTP protocol through firewalls, helps to prevent servers from identifying the client hostname, which may reveal certain personal information[1].
2. An IP multicast has interesting property that a sender of packets is hard to be recognized[9]. Thus, it can be considered as a kind of practical sender-anonymous communication.
3. An ordinary mass media such as newspaper, magazine and TV are one way recipient-anonymous communication where intended recipient is impossible to be identified (thus, it is useful for the perfect crime[10])

From this observation, we see the following four classes of anonymity.

1. Sender-anonymity over recipient
2. Sender-anonymity over someone else
3. Recipient-anonymity over sender (meaningless)
4. Recipient-anonymity over someone else

For example, Crowds is a typical attempt to ensure the sender anonymity, which improves as number of entity increases. However, at the same time, it reduces recipient anonymity, because every entity who forwards a packet would know the intended recipient.

Moreover, some applications like web requires a round trip anonymous communication, while an one way anonymous transfer is sufficient in electronic voting. There is a tradeoff between a strength of anonymity and efficiency. An appropriate anonymous technology should be chosen for particular application.

Chaum presented a well-known technique for anonymous communication called MIX-NET[16], where the route is specified by sender such that no router can learn neither source nor destination other than one hope forward and backward. A packet is encrypted multiple times with different routers public key, like onion[2]. This primitive idea satisfies sender-anonymity against intruder who taps the line. Since the MIX-NET was originally used in electronic voting that the authority can not violate voters privacy, the recipient (the authority) does not necessarily respond to the sender (the voters).

Unfortunately, MIX-NET is volunable by traffic analysis which identifies who is sending to whom especially when no congestion of packets happens. To cope with traffic analysis, pseudo packets may be randomly generated. Alternatively, the packet forwarding may be delayed until enough number of packets arrive.

We propose some protocols for anonymous communication using multiparty computation. In the proposed protocol, the traffic among parities is independent of the amount of information communicated with them. Hence, the anonymity for both sender and recipient archived against any traffic analysis.

2 Proposed Protocol

2.1 Multiparty Protocol for Addition

Shamir's secret sharing scheme enables that n users having each secret $s_1, \dots, s_n$ respectively can compute the total $S = \sum_{i=1}^{n} s_i$ without revealing each secret to anyone else.

(Protocol 1)

Step 1: User i composes degree t random polynomial $f_i(x) = s_i + a_1 x + \cdots a_t x^t \pmod p$ with his secret s_i at the constant. For $j \neq i \in \{1, \dots, n\}$, he sends $f_i(\alpha_j)$ in secure communication that ensures a confidentiality and an integrity. Where, $\alpha_1, \dots, \alpha_n \in Z_p$ are public information assigned with n users and previously distributed.

Step 2: User i $(i = 1, \dots, n)$ computes

$$c_i = \sum_{j=1}^{n} f_j(\alpha_i) \pmod p.$$

With commitment by an appropriate protocol, he publishes c_i.

Step 3: User i receives $c_1, \dots, c_n$ that are n points of the same polynomial defined by the sum of n polynomials $F(x) = f_1(x) + \cdots + f_n(x)$, and he uses LaGrange scheme to solve it. Thus, every user can agree on the same total of their secrets $S = s_1 + \dots + s_n$.

We know $t+1$ points are necessary to uniquely identify t-degree polynomial. Hence, Protocol 1 is said to be t-private, that is, it keeps secrets against at most t users collusion. This protocol is often used as a primitive protocol to compose more complicated distributed computation without revealing secrets[11].

Protocol 1 can be simplified by the following protocol that reduces computation of polynomial.

(Protocol 2)

Step 1: User i picks n random numbers $b_{i1}, \dots, b_{in}$ such that

$$b_{i1} + \cdots + b_{in} = s_i \pmod p.$$

For each $j \neq i \in \{1, \dots, n\}$, he sends b_{ij} in secure communication.

Step 2: User i computes subtotal

$$c_i = \sum_{j=1}^{n} b_{ji} \pmod p$$

and publishes it in certain commitment protocol.

Step 3: Every users learn the total S by adding all published numbers as follows:

$$S = c_1 + \ldots + c_n = s_1 + \ldots + s_n$$

Note that user i does not reveal b_{ii} to anyone else. Hence, Protocol 2 is $(n-1)$-private protocol.

2.2 Sender Anonymity

Here we consider one way anonymous communication, which will be extended later.

Let suppose that sender i wishes to send a message to recipient j without being traced by the recipient.

Recall the secret computation of addition (Protocol 1 or 2), the result is the total of each secret. Consider only sender sets a message as his secret and all other users has set to be 0 in the secret computation. Then, the total result would recover the senders' message. We assume that users send message independently, and the probability that a user sends message at time T is considerable small. Then, in terms of the result of secret computation at time T, $S(T)$, the following three cases happens:

1. No one sends a message at T. The protocol results in $S(T) = 0$.
2. One user sends a message at T. The message M will be received by all users as $S(T) = M$, and we call successful transmission.
3. More than one users send messages simultaneously. The result $S(T)$ will be the sum of the different messages, and transmission fails. The senders can know the collision happens because $S(T)$ is not as they expected, thus, they go on re-sending after waiting independent period of time. Other than senders no one knows whether the transmission succeeded or failed.

All users hold on multiparty protocols constantly even if no message is communicated. Although this protocol is quite expensive in terms of bandwidth, it should be provided as a primitive protocol and will be revised to reduce its communication cost.

For optimal retransmission, the waiting time is decided depending of congestion. Binary exponential backoff algorithm states that resend packet with probability $q_r = 2^{-i}$, where i be a number of subsequent failures[13]. It can work with any multi-access communication including the proposed anonymous protocol.

2.3 Recipient Anonymity

Next, we consider a recipient-anonymity under assumption of computational complexly of a public key cryptosystem. Simply have message encrypted with an intended recipient's public key, and broadcast it through sender-anonymous channel. In broadcasting, everybody has a possibility to be the recipient, however, no one can read the encrypted message except for the intended recipient who has the corresponding private key.

2.4 Round-trip Anonymity

Finally, we should cope with response from the recipient. Not like the batch-style application including email and news, most current applications require interaction between a client and a server at least one round. However, it is a little hard to respond to the sender whom the recipient should not know. Furthermore, the fact of responding should not be known by anyone else.

In the forwarding message from the original sender to the recipient, the sender can contain an one-time symmetric encryption key, by which the recipient wraps her responding message so that only the unknown sender can read it. Then, she participates in the multiparty protocol for addition with the encrypted message. Note that public key encryption is not necessarily required in return trip.

Consequently, we have the following protocol for an anonymous round-trip communication:

(Protocol 3)

Step 1: User i participates in multiparty computation (Protocol 1 or 2) with his secret s_i defined by

$$s_i = \begin{cases} X & \text{willing to send} \\ 0 & \text{otherwise,} \end{cases}$$

where X is a concatenated two messages computed as

$$X = E_j(k)|E_k(M),$$

where E_j is an asymmetric encryption with recipient j's public key, and E_k is a symmetric encryption with shared key k.

Step 2A: Except the sender, every user tries to decrypt with each private key the result $S(T)$. If a user succeeds, she knows the message was sent to her, and go to Step 3. Otherwise, go back to Step 1.

Step 2B: Sender ensures $S(T) = X$, if it does not hold, that is, there is a collusion with someone else, cancel sending meaningful message at Step 1 in certain steps, and then resend X at time T' ($T' > T$). Repeat resending until he succeeds.

Step 3: Recipient j obtains M by decrypting both encryptions on message. If she need to respond message M_2, set her secret as $s_j = E_k(M_2)$ and go to Step 1. (The original sender who expects response uses not only his private key but also the stored one-time symmetric key k to attempt decryption at Step 2A.)

Step 4: Go to Step 1

2.5 Non-public key Protocol

In Protocol 3, a transmission fails when any two users happen to send message at the same time slot T. As the number of participants, n, increases, the chance to send successfully approaches 0.

In Step 2 of Protocol 2, the subtotal is broadcasted through a commitment protocol in order to prevent cheating users with bogus subtotal that is modified accordingly as other published subtotals. We use $n-1$ pier-to-pier connections for broadcasting messages. In this scheme, the sender is allowed to change the subtotal according to recipients without revealing it. The collusion happens only if multiple users send the same recipient at the same time slot. This property reduces the probability of collusion, which improves the bandwidth of anonymous channel. Moreover, with different pair of sender and recipient, multiple communications can be involved at the same time.

Notice that we need no asymmetric cryptography in the revised protocol, because the message is sent only the intended recipient without revealing true sender. Instead, we need any check digit to see if the message is valid or spoiled with some simultaneous senders. Appropriate hash function can be used to this verification.

Unfortunately, in the revised protocol, senders can not learn from their total $S(T)$ whether their transmission attempts succeed or not. Hence, we develop an acknowledge protocol that recipient tells anonymous sender the result of transmission without revealing sending the acknowledge other than the recipient and the senders.

At subsequent time slot $T+1$, the recipient sends acknowledge random numbers that sum up to non-zero for the successful transmission; sum to zero for failure. Since the recipient does not know who is the sender, she should send to all possible senders the same random number by which only the sender can identify the result of his transmission. We use the hash of the message as the successful acknowledgement (Step 4), which should be discarded secretly so that no one except the sender and recipient can notice it is the acknowledge message (Step 5).

Suppose recipient i send a successful acknowledge message $h(M)$ to everybody who takes it as i's random number at time $T+1$. Her subtotal is $nh(M) \pmod p$, which is then added to the true total $S(T+1)$. There must exist exactly one sender, who can notice that the recipient is sending invalid random number. Hence, only the sender can cancel her invalid numbers by picking $xinZ_p$ which satisfies

$$h(M) + (c_i(T+1) + x) = 0 \pmod p.$$

Consequently, we have the revised protocol:

(Protocol 4)

Step 1: User i picks n random numbers $b_{i1}(T), \ldots, b_{in}(T)$ such that

$$b_{i1}(T) + \cdots + b_{in}(T) = 0 \pmod p.$$

For each $j \neq i \in \{1, \ldots, n\}$, he sends $b_{ij}(T)$ at time slot T in secure communication.

Step 2: User i computes subtotal

$$c_i = \sum_{j=1}^{n} b_{ji}(T) \pmod p,$$

and sends $c_i(T) + M|h(M)$ to j to which he is willing to send message M and c_i to other users. The hash function $h()$ will be used by recipient j to detect a collusion.

Step 3: User j computes the total $S_j(T)$ of all received numbers. If $S_j(T) = 0$, no one is sending her, and go to Step 1; otherwise someone is sending message to her. Check if the message satisfies $S_j(T) = M|h(M)$, which mean the transmission is successful without no collusion.

Step 4 (Step 1'): In successful transmission, recipient j sends an acknowledge message to anonymous sender by setting random number as

$$b_{j1}(T+1) = \cdots = b_{jn}(T+1) = h(M) \pmod p$$

at Step 1. Otherwise, a collusion arises, and recipient j sends random numbers that sums up to 0 as usually at Step 1.

Step 5 (Step 2'): The sender i learn whether his message is sent correctly or not by checking $b_{ji}(T+1) = h(M)$. If this holds, he fixes his subtotal $c_i(T+1)$ as follows:

$$c_i(T+1)' = c_i(T+1) - h(M) \pmod p.$$

Note that this cancels recipient j's invalid numbers and opens the channel for other users.

Otherwise, his attempt to send message to j fails, and accordingly waits a chance for resending.

Note that this improvement does not apply to Protocol 1 that is based on Shamir's secret sharing scheme.

3 Estimate

We estimate the bandwidth consumption and the latency in our proposed protocol.

In Protocol 3, a message involves $n(n-1)$ pier-to-pier connections at Step 1, and n broadcasts at Step 2. The channel efficiency, a fraction of bandwidth consumption over the normal network, is given by

$$\text{Channel efficiency} = (n(n-1) + Bn)$$

where B is a cost of broadcasting. The order of $O(n^2)$ is not scalable to number of entities.

For simply estimate protocols, we consider the capacity of anonymous channels, that is, the maximum bits to be communicated in the channel in a second. We assume a constant size of message, M bits, which takes M/C seconds to transfer between two entities in the network. At step 1, an entity distributes random numbers to $n-1$ entities, so it takes $M/C(n-1)$ seconds per entity. Since we assume a centralized switch network, multiple connections can be established at same time slot. For instance of $n = 4$, while entity 1 communicates with entity 2, entity 3 and 4 can communicate, too; so both connections $(1,2)$ and $(3,4)$ are in the first slot. As the same way, $(1,3),(2,4)$ are in second, $(1,4),(2,3)$ are last. Thus, distributing random number spends three time slots. More generally, n entities spend at least $2\lfloor n/2 \rfloor - 1$ time slots to completely exchange random numbers.

In Protocol 3, at most one message can be transferred in one round (Step 1 to 3). By assuming an efficient broadcast which takes one time slot per one message, which is repeated for n times, we have the capacity of channel of Protocol 3 as follows:

$$C_3 = \frac{M}{(M/C)(2\lfloor n/2 \rfloor - 1 + B)} = \frac{C}{2\lfloor n/2 \rfloor - 1 + n}.$$

Instead of the efficient broadcast, n connections are involved for each entity in Protocol 4, so, the total of $B = 2\lfloor n/2 \rfloor - 1$ time slots are necessary at Step 3 in the same way as C_3 is resolved. On the other hand, it is possible to multiplex at most $\lfloor n/2 \rfloor$ connections. Since an extra acknowledge round is involved per a message, the maximum total message contains $M\lfloor n/2 \rfloor$ bits for two rounds (Step 1 to 5). Consequently, we have the capacity of anonymous channel of Protocol 4 as

$$C_4 = \frac{M\lfloor n/2 \rfloor}{4(M/C)(2\lfloor n/2 \rfloor - 1)} = \frac{C}{8 - 4/\lfloor n/2 \rfloor}$$

Figure 1 demonstrates the improvement of Protocol 4 in terms of capacity. Letting $C = 1$, it shows the two capacities depending on number of entities, n.

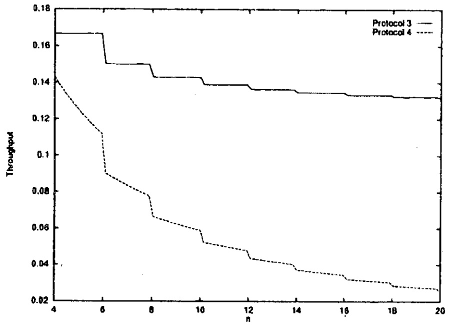

Figure 1: Capacities of Anonymous Channels

Note that the capacity is the maximum information transferred through a given communication channel, and the capacity is achieved under a perfectly scheduled communications among n entities. In practice, as network is congested, more collisions happen and the bandwidth would be saturated with retransmissions. It is because the model is equivalent to a general multi-access communication such as Ethernet. Hence, we have to take into account the collisions.

4 Conclusion

We present a new anonymous communication protocols using multiparty computation technique, which keeps both sender and recipient anonymity against any traffic analysts attack. We examines the communication cost of proposed protocol.

References

[1] Anonymoizer Inc. Web page, `http://www.anonymizer.com`

[2] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed, "Anonymous Connections and Onion Routing," IEEE Symposium on Security and Privacy, pp44-54, 1997

[3] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson, "Hiding Routing Information," Workshop on Information Hiding, 1996

[4] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag, "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference,1996

[5] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, Vol.24, No2, pp84-88, 1981

[6] Michael K. Reiter, Aviel D. Rubin, "Crowds: Anonymity for Web Transactions," DIMACS Technical Report 97-15,1997

[7] *InternetWatch*, 03, 1998-1-7

[8] Debby M. Wallner, Eric J. Harder, and Ryan C. Agree, Key Management for Multicast: Issues and Architectures, Internet Draft, 1997

[9] A. Ballardie, Scalable Multicast Key Distribution, RFC 1949, 1996

[10] S. von Solms and D. Naccache, On Blind Signatures and Perfect Crimes, Computers and Security, v.11, 1992, pp.58-583

[11] Ben-Or, M., Goldwasser, S. and Wigderson, A., Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, ACM STOC, pp.1-10, 1988

[12] E. Fujisaki and T. Okamoto, "Practical Escrow Cash Systems", Security Protocols, M. Lomas (Ed.), Lecture Notes in Computer Science 1189, Springer-Verlag, pp.33–48 (1997).

[13] D. Bertsekas, R. Gallager, Multiaccess Communication, *DATA NETWORKS*, Prentice Hall, pp.277-287, 1992

[14] Kikuchi, et. al., Privacy-oriented Web Audience Rating in the Internet, SCIS96-7A, 1996

[15] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, " Communications of the ACM, Vol.24, No.2, pp.84-88, 1981

[16] D. Chaum, "Security without Identification: Transaction systems to Make Big Brother Obsolete," Communications of the ACM, Vol.28, No.10, pp.1030-1044, 1985