To appear in Proc. Eurocrypt '94, LNCS, Springer-Verlag

Breaking an Efficient Anonymous Channel

Birgit Pfitzmann

Universität Hildesheim, Institut für Informatik,
Marienburger Platz 22, D-31141 Hildesheim, Germany
pfitzb@informatik.uni-hildesheim.de

Abstract. At Eurocrypt 1993, Park, Itoh, and Kurosawa presented an "all/nothing election scheme and anonymous channel". The schemes are based on the mix-net and the election scheme constructed from this anonymous channel (Chaum 1981). One of the two main improvements is that the messages sent by normal participants are significantly shorter in the two new anonymous channels. However, we show several successful attacks on these channels and thus on the secrecy of the votes in the election scheme. They break the first, more efficient channel completely. For the second channel and the election protocol, we present some countermeasures against all our attacks. Note, however, that we do not guarantee security even then, and that the specification of that channel is somewhat weaker than that of the original mix-net.

1 Introduction

The article [8] proposes two cryptographic protocols to realize an anonymous channel and one protocol for multi-party elections based on them. The structure of these protocols is adapted from the mix-net and the corresponding election scheme [3]. It was disputed at once [2] if the election scheme is really more efficient than others like [5, 1]. Here, however, we concentrate on the anonymous channels, which are indeed quite efficient, but unfortunately not very secure.

1.1 Purpose of the Given Anonymous Channels

First, we briefly sketch what "anonymous channel" means in this context. (A precise definition is not necessary in the following — it will be clear at once that our attacks are successful.) An anonymous channel is a multi-party protocol. Each of the participants has a secret input, called its message (such as a vote). At the end of the protocol, each participant obtains a result. This should be the list of all the secret messages in alphabetical order. However, the participants should not learn which of these messages originated from which of the other participants. Thus the message contents become known, but the senders of the messages are anonymous, or, in other one words, untraceable.

Anonymous channels can be considered in different fault scenarios. In [8], the scenario is as follows:

- There are k distinguished participants, the "shuffle machine agents". We use Chaum's original and shorter word mix instead. Anonymity should be guaranteed as long as at least one mix is honest.
- Anonymity is computational. (That is, it may rely on an assumption such as that computing discrete logarithms is infeasible.)

- Correctness of the result is only guaranteed if all the mixes are honest. (This is the difference to a full election protocol, where stronger correctness requirements are made.)
- A non-anonymous reliable broadcast channel, called a public board, is given.

This is the same scenario as for the original mix-net, except that the latter does not require reliable broadcast [3].[1]

Both protocols in [8] and the original mix-net are of the following structure: First, each participant sends an encrypted form of its secret message to the public board or the first mix, respectively. The rest of the protocol is just between the mixes. With the original mix-net, the size of this first ciphertext is proportional to the number k of mixes used. (Recall that increasing k increases the security.) In [8], this ciphertext is always just two ElGamal blocks long. Taking special care about the length of this first ciphertext is reasonable, since normal participants may have to rely on normal communication infrastructure, whereas mixes may have special broadband connections.[2]

The difference between the two channels in [8] is that the first one is completely non-interactive between the mixes, i.e., each message is passed through each mix exactly once. This protocol does not even make use of reliable broadcast between the mixes.

1.2 Overview of the Attacks

We found two types of attacks on the anonymous channels in [8]. The first one is a simple passive attack that is successful against both protocols. Passive means that the attacker only observes the participants. However, this attack can be countered with a small change to the protocols. Secondly, there is an active attack, i.e., the attacker deviates from his protocol. This attack also works against both channels. With the first channel, we do not see any successful countermeasures (unless one destroys the advantage over the second channel or the original mix-net). With the second channel, the attack can be countered; however, one really seems to need reliable broadcast for this purpose.

Historically, active attacks on mix-nets were first considered in [9]; in particular, the direct implementation of the mix-net as described in [8] with RSA as the cryptosystem was broken there. However, countermeasures were also proposed in [9], so that the mix-net as such can be considered secure. Such attacks are also mentioned in [10].

2 The Proposed Anonymous Channels

The two protocols for anonymous channels in [8] are called “Type 1 channel” and “Type 2 channel”. The former is more efficient, the latter more suitable for the election protocol. The protocols have a common beginning and different endings.

---

[1] An additional disadvantage over the original mix-net, after the countermeasures that we introduce to retain any security, is that it becomes possible for any participant to disrupt the protocol, not just for any mix.

[2] However, normal participants usually do not have physical broadcast channels, and a cryptographic solution would destroy the efficiency intended in [8] (since each participant would have to send at least one separate message to each mix). Hence the whole approach makes more sense if one does not use the public board for these first ciphertexts. We mention in some places what happens in this case.

2.1 The Common Beginning

Both channels are based on the ElGamal cryptosystem [7].

- *General information*: The following information is agreed upon and published once and for all: a large prime p, the factorization of $p - 1$, and a primitive element g of $\mathbb{Z}_p^*$ (i.e., an element that generates the whole multiplicative group modulo p).
- *Keys of the mixes*: Let there be k mixes, $M_1, \ldots, M_k$. Each mix M_i has an ElGamal key pair, consisting of a secret key $x_i \in \{1, \ldots, p-1\}$ and a public key
$$y_i := g^{x_i} \bmod p.$$
- *Start of a protocol execution*: Now consider a particular execution of the anonymous channel protocol. Let the secret message of a participant P_j be $m_j \in \{1, \ldots, p-1\}$. Then P_j prepares the ciphertext C_j that it will send to the first mix as follows: It chooses a random number R from $\{1, \ldots, p-1\}$, encrypts m_j as
$$C_j := (g^R, m_j (y_1 \cdots y_k)^R), \qquad (*)$$
and publishes C_j on the public board.

2.2 Ending of the Type 1 Channel

In the second part of the first protocol, each mix in turn processes all the ciphertexts. Basically, it strips the parts with its own y_i off. Thus the last mix outputs the list of messages m_j. Before making any output, each mix reshuffles the messages, e.g., in alphabetical order (otherwise it would be trivial to know which message is which). More precisely, each mix M_i for $i := 1, \ldots, k-1$ acts as follows:

- It reads a list of intermediate ciphertexts of the form (t, u) that its predecessor has put on the public board.
- For each of these pairs, it chooses a random number r and computes
$$(t\, g^r, u\, (y_{i+1} \cdots y_k)^r / t^{x_i}).$$
- It outputs the new pairs in alphabetical order.

One can easily show by induction that if all the preceding mixes worked correctly, the ciphertext from $(*)$ has become
$$(t, u) = (g^{R'}, m\, (y_i \cdots y_k)^{R'}) \qquad (**)$$
for some R' when it is input to M_i, and M_i transforms it into
$$(g^{R'+r}, m\, (y_{i+1} \cdots y_k)^{R'+r}).$$
The final mix, M_k, only strips the remaining y_k off, without reencoding the first component of the pair. Thus it only transforms (t, u) into u / t^{x_k}. If all the mixes worked correctly, this is m. Note that this protocols is unchanged if each mix sends its output to the next mix in private, instead of using a broadcast channel.

2.3 Ending of the Type 2 Channel

The Type 2 channel consists of two subprotocols: In the first subprotocol, each mix M_i only reencodes the ciphertexts, without removing the part with its y_i. In the second subprotocol,

each mix publishes what the remaining part with its y_i is, so that everybody can remove these parts locally.

Subprotocol 1. For $i := 1, \ldots, k$, mix M_i acts as follows:

- It reads a list of intermediate ciphertexts of the form (t, u) that its predecessor has put on the public board.
- For each of these pairs, it chooses a random number r and computes
$$(t\, g^r, u\,(y_1 \cdots y_k)^r).$$
- It outputs the new pairs in alphabetical order.

Subprotocol 2.

- Each mix M_i reads the list of intermediate ciphertexts of the form (t, u) that M_k wrote at the end of Subprotocol 1, and for each of these pairs, it outputs $z_i := t^{x_i}$.
- Now each participant, for each entry (t, u), can compute
$$u \,/\, (z_1 \cdots z_k).$$

As above, one can easily show that the output resulting from C_j in this way is m_j if all mixes are honest. Note that this protocol really makes use of the given broadcast primitive.

3 The Simple Passive Attack

Note that the attack in this section is not our main attack, but it makes no sense to make more complicated attacks before one has countered the easier ones.

3.1 Basic Form

The idea of the attack is simply that the encryption scheme used does not hide all partial information, and thus one can restrict the number of participants who may have sent a particular message. More precisely, the group $\mathbb{Z}_p^*$ has subgroups. At least the group order $p - 1$ is even, and thus there is a subgroup U_2 of order 2, generated by g^2. Anybody can easily test if a group element a is in U_2 by testing if $a^{(p-1)/2} = 1 \bmod p$. Similarly, if $p - 1$ has another prime factor f, there is a subgroup U_f, and the criterion for membership in U_f is $a^{(p-1)/f} = 1 \bmod p$.

We show that the residue class of a message with respect to such a subgroup is not hidden. (This must have been noticed before, but we do not know a reference.) For example, consider just one mix and two participants P_1, P_2 with messages m_1 and m_2; all three are honest. In this case, an outsider should not be able to trace which of the two participants sent which of the two messages. The two ciphertexts on the public board are of the form

$$C_1 = (t_1, u_1) = (g^R, m_1 {y_1}^R),$$
$$C_2 = (t_2, u_2) = (g^{R'}, m_2 {y_1}^{R'}).$$

Note that it is public that C_1 was sent by P_1 and C_2 by P_2. The mix outputs m_1 and m_2 in alphabetical order. Now the attacker tests if t_1 is in the subgroup U_2. This is true in half the cases. If yes, he knows that ${y_1}^R$ is in this subgroup, too, since

$$y_1{}^R = (g^{x_i})^R = (g^R)^{x_i}.$$

Hence he knows that u_1 and the message hidden in it are in the same residue class with respect to this subgroup. (In simpler words, in the case of U_2, he knows that $u_1 \in U_2 \Leftrightarrow m_1 \in U_2$.) Thus if exactly one of m_1 and m_2 is in the subgroup, the attacker knows which of them is P_1's message.

3.2 Extensions

If there are more than two participants, the attacker can partition them into possible senders of certain messages. Note that even a rough partition, such as in two subsets, is quite dangerous in practice, since an attacker may know or guess from context information that a series of messages to the same recipient have the same sender. He can then construct the intersection of the sets of possible senders of each of these messages to identify the sender.

If there is more than one mix, the attacker can relate the initial C_j and the final m_j in the same way. Hence it is obvious that the attack is successful against both protocols.

3.3 A Countermeasure

The countermeasure against the simple passive attack is to use a multiplicative group of prime order. Usually one does this by choosing two primes, p and $q \mid p - 1$, and using the subgroup U_q of $\mathbb{Z}_p{}^*$. In our case, one has to encode the given messages into this subgroup in an easily recoverable way. The easiest way is to use $p = 2q + 1$ and the factor group $F_q := \mathbb{Z}_p{}^* / \{1, -1\}$. It is represented by the numbers $\{1, \ldots, (p - 1)/2\}$ (as noted for a similar purpose in [4]).

Now g has to be a generator of the group of prime order, i.e., F_q, and random exponents are from the interval $\{0, \ldots, q - 1\}$. The rest of the protocol can be described as before. Then the conjecture would be that no information is known (in efficiently computable form) by g^R about $y_1{}^R = (g^R)^{x_i}$.

4 Active Attack on the Type 1 Channel

Our main attack is active, i.e., some dishonest mixes or participants perform transformations on the messages different from those prescribed in the protocol. The basic idea is that the attackers take the ciphertext C_j of an honest participant P_j, prepare a somehow related ciphertext C_j', and input both these ciphertexts to an honest mix. The relation between the inputs is chosen so that the outputs are also related. The attackers search through the outputs until they find a pair of related ones; then with probability very close to 1 they have identified the message from P_j.

Before going into details of the real attack, note that a very simple form of this attack was generally countered in the original protocols in [3], but no longer in [8]: the mixes never process the same input twice. In [8], it is obvious that at least the last mix would need the same measure: If it gets the same input twice, it also gives the same output twice, and hence one can see which output corresponds to this input.

4.1 Basic Form

For the real attack, assume that participant P_j and mix M_i for some $i \in \{2, \ldots, k–1\}$ are honest (and at least one more participant — otherwise there is nothing to hide). Then P_j should be anonymous even if the remaining mixes collude. Let P_j's ciphertext be $C_j = (g^R, m_j (y_1 \cdots y_k)^R)$ as in Equation $(*)$. The first dishonest mixes know how they transform this ciphertext into a form $C_j' = (t, u) = (g^{R'}, m_j (y_i \cdots y_k)^{R'})$ as in $(**)$, i.e., they know that C_j' contains the secret message of P_j. To find out what M_i does, they prepare the second, related ciphertext as follows:

$$C_j'' := (t^x, u^x) = (g^{R'x}, m_j^x (y_i \cdots y_k)^{R'x}),$$

where x is a random number. For now, just assume that M_i indeed processes both C_j' and C_j'' — we consider in Section 5.2 how it might try to prevent this attack and how the attackers deal with that. According to its protocol (see Section 3.2), M_i transforms C_j' and C_j'' into

$$(g^{R'+r}, m_j (y_{i+1} \cdots y_k)^{R'+r})$$

and

$$(g^{R'x+r'}, m_j^x (y_{i+1} \cdots y_k)^{R'x+r'}),$$

respectively, and outputs these two pairs somewhere in its alphabetical list. With their legal transformations, the remaining attacking mixes, $M_{i+1}, \ldots, M_k$, obtain

$$m_j \text{ and } m_j^x,$$

respectively, from these two entries, somewhere in the final list of messages. Now they exponentiate each resulting message m with x and check if the result occurs among the other messages. If yes, this m was almost certainly m_j, since it is highly unlikely that two honest participants should have input messages that are in this relation.

4.2 Some Countermeasures and Why They Don't Work

The honest mixes might try to take measures in addition to the original protocol to prevent the attack described above. We discuss a few possibilities and how the attackers counter them.

- If each mix checks that the number of messages has not changed compared to the first list on the public board, the attackers omit another message when they add C_j''. This should be a message from a colluding participant, so that no message from an honest participant is missing in the final output.
- One might introduce redundancy into the messages m_j (such as a string of zeros at the end of the message, as in the election protocol) in the hope that m_j^x is not of this form, so that the attack is detected. However, this is of no use, since the attacking mix M_k need not output m_j^x: The attackers can do their computations in private, find out which message is m_j^x (this is now even easier to see by its wrong form), and replace it by a message m' of the correct form.
- Finally, the honest participant might try to discover that two inputs to M_i are of the form (t, u) and (t^x, u^x) for an arbitrary, unknown x, or that one mix has not transformed a message correctly. However, both these tasks seem equally difficult as passive attacks on the original scheme (assuming that the attack from Section 3 has been countered), which is the problem of recognizing a Diffie-Hellman key.

4.3 Countermeasures that Yield the Original Mix-Net

Of course, one can counter such active attacks (at least if one does not bother about proofs): As usual with active attacks on encryption schemes, one introduces redundancy into the encrypted message (such as a string of zeros at the end, although this is a rather weak form). After deciphering, the recipient only outputs the message if it is correct; moreover, one has to check for duplicates. For special measures against active attacks on the ElGamal cryptosystem, see [6, 11].

In the given scenario, the recipient under attack can be any mix M_i. Hence there must be redundancy in the result any M_i obtains after its operation. With all known scheme, this makes the total length of the ciphertext proportional to the number of mixes again, which [8] tried to avoid.

4.4 Attack on Several Honest Mixes

The attack can be extended to any number of honest mixes. This is fairly clear since the transformation that several honest mixes perform is just like that of one mix, but with the sum of the individual secret keys x_i as the public (group) key. First take the case where neither the first nor the last mix are honest. Then M_1 can produce a ciphertext C_j'' related to the C_j' they want to trace as before. All other mixes process it honestly, so that M_k obtains m_j and m_j^x. Again it can output a better-looking message instead of m_j^x to the public board.

If the first mix is honest, the possibility of the attack depends on synchronism: If the participants can write their inputs on the public board in any order, the attackers can write theirs last and already choose them as variants of C_j. Otherwise, M_1 can prevent the attack. (If the same key is used in successive protocol executions, one needs secret key encryption between the participants and the first mix.) If the last mix is honest and one adds redundancy to the messages, the attack is obviously prevented. Anyway, this is no consolation, since there would be no need to use several mixes if one knew one could trust the first or the last one.

5 Active Attack on the Type 2 Channel

5.1 Basic Form

The attack on the Type 2 channel is similar to that in Section 4. We just describe it for the case where the first and last mix are attacking, but any number of the others may be honest.

- As before, M_1 transforms the interesting ciphertext $C_j' = (t, u)$ into a related version $C_j'' := (t^x, u^x)$.
- All the other mixes, honest or not, perform their correct transformations. This yields two entries of the form $(g^{r'}, m_j(y_1 \cdots y_k)^{r'})$ and $(g^{r''}, m_j^x(y_1 \cdots y_k)^{r''})$ in the list at the end of Subprotocol 1.
- All mixes perform Subprotocol 2 correctly. This yields outputs of the form m_j and m_j^x, which can be recognized as before.

5.2 Countermeasures and How Far They Help

As in Section 4.2, the mixes can omit the ciphertext of an attacking participant in the place of C_j", so that the other participants do not notice that this ciphertext was added, nor that another message is missing at the end.

If there is redundancy in the messages, it is now harder for the mixes to get $m_j{}^x$ replaced by a correct-looking message m' again. During Subprotocol 1, they do not know yet which message to replace; hence they have to do it during Subprotocol 2. If, as can be assumed, mix M_k is the last one to perform the first step of Subprotocol 2, i.e., to output his factors z_k, he can already perform the second step in private and thus see which of the messages are m_j and $m_j{}^x$. Then he replaces $m_j{}^x$ by m' by choosing the corresponding z_k so that $u / (z_1 \cdots z_k) = m'$.

If the protocol is now changed so as to enforce simultaneous broadcast (i.e., no mix can choose its value z_i after having seen the others), it seems that the attack will be detected. If no physical simultaneous broadcast is available (this is an even more unusual primitive than reliable broadcast), one can, as usual, simulate it with a commitment scheme.

Note, however, that these countermeasures only help to detect the attack after the fact, i.e., the attackers have successfully traced one or more messages once. If one does not tolerate this, one might split messages up, as it is done in the election protocol for a different purpose. If one tries to achieve high probabilities of detection, however, one has to reconsider the efficiency. Furthermore, one will have to use secret key encryption, as mentioned in Section 4.4, so that normal participants cannot mount the attack without the help of a mix.

Another problem with redundancy checks is that any participant can now disrupt the anonymous channel (or the election protocol) — in the original mix-net, only mixes can do this, which seems far less likely. Since it was not claimed that the protocols provide correctness under attacks, we did not consider this issue further. However, if the channel were to be used in practice, one would at least have to reconsider identification of disrupters (with broadcast or additional signatures in the messages) and then to compare the efficiency again.

6 Outlook

We have shown several attacks on the anonymous channels from [8] and countermeasures for those on the so-called Type 2 channel. We do not guarantee that the channel or the corresponding election protocol is secure after these modifications. Generally, one can only plead that at least as much care is taken with respect to partial information and active attacks with large protocols as with cryptosystems and signature schemes. Even with schemes like those in [8] where one cannot expect to be able to prove that they are as hard as, say, computing discrete logarithms (because of the exploitation of multiplicativity), an outline of a security analysis would be helpful.

Acknowledgements

It is a pleasure to thank *Josh Benaloh* and *David Chaum* for an interesting discussion that made me thinnk about this subject, and *Andreas Pfitzmann* and *Michael Waidner* for helpful comments on this paper. The passive attack was noticed independently by *Joe Kilian* and *Kazue Sako*.

References

[1] Josh Cohen Benaloh: *Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret*; CRYPTO '86, LNCS 263, Springer-Verlag, Berlin 1987, 251-260.

[2] Josh Benaloh: Questions after the presentation of [8], Lofthus, May 1993.

[3] David Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*; CACM 24/2 (1981) 84-88.

[4] David Chaum, Hans van Antwerpen: *Undeniable signatures*; CRYPTO '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 212-216.

[5] Josh Cohen, Michael Fischer: *A robust and verifiable cryptographically secure election scheme*; FOCS 1985, 372-382.

[6] Ivan Damgård: *Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks*; CRYPTO '91, Springer-Verlag, Berlin 1992, 445-456.

[7] Taher ElGamal: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*; IEEE Transactions on Information Theory 31/4 (1985) 469-472.

[8] Choonsik Park, Kazutomo Itoh, Kaoru Kurosawa: *All/Nothing Election Scheme and Anonymous Channel*; EUROCRYPT '93, Pre-proceedings, Lofthus, May 1993, T97-T112.

[9] Birgit Pfitzmann, Andreas Pfitzmann: *How to Break the Direct RSA-Implementation of mixes*; EUROCRYPT '89, Springer-Verlag, Berlin 1990, 373-381.

[10] Charles Rackoff, Daniel Simon: *Cryptographic Defense Against Traffic Analysis*; STOC 1993.

[11] Yuliang Zheng, Jennifer Seberry: *Immunizing Public Key Cryptosystems Against Chosen Ciphertext Attacks*; IEEE J. Selected Areas in Communications 11/5 (1993) 715-723.