Four Practical Attacks for "Optimistic Mixing for Exit-Polls"

Douglas Wikström, douglas@sics.se
Swedish Institute of Computer Science (SICS)
Box 1263, S-164 29 Kista, Sweden

February 25, 2003

Abstract

Golle, Zhong, Boneh, Jakobsson, and Juels [10] recently presented a very efficient mix-net, that they claim to be both robust, and secure. We present four practical attacks for their mix-net, and break both its privacy and robustness. Each attack exploits and illustrates a separate weakness of the protocol.

The first attack breaks the privacy of any given sender *without corrupting any mix-server*. The second attack requires that the first mix-server is corrupted. Both attacks are adaptations of the "relation attack" introduced by Pfitzmann [26, 25].

The third attack is similar to the attack of Desmedt and Kurusawa [4] and breaks the privacy of *all* senders, and *robustness*. It requires that all senders are honest, and that the last mix-server is corrupted.

The fourth attack is novel and breaks the privacy of any given sender. It requires that the first and last mix-servers are corrupted. This attack breaks also "Flash Mix" by Jakobsson [15], including the fixed version given by Mitomo and Kurosawa [19].

Keywords: cryptanalysis, mix-net, anonymous channel, electronic voting.

SICS Technical Report: T2003:04
ISSN: 1100-3154
ISRN: SICS-T-2003/04-SE

Four Practical Attacks for "Optimistic Mixing for Exit-Polls"

Douglas Wikström

Swedish Institute of Computer Science (SICS)
Box 1263, S-164 29 Kista, Sweden
douglas@sics.se

Abstract. Golle, Zhong, Boneh, Jakobsson, and Juels [10] recently presented a very efficient mix-net, that they claim to be both robust, and secure. We present four practical attacks for their mix-net, and break both its privacy and robustness. Each attack exploits and illustrates a separate weakness of the protocol.

The first attack breaks the privacy of any given sender *without corrupting any mix-server*. The second attack requires that the first mix-server is corrupted. Both attacks are adaptations of the "relation attack" introduced by Pfitzmann [26, 25].

The third attack is similar to the attack of Desmedt and Kurusawa [4] and breaks the privacy of *all* senders, and *robustness*. It requires that all senders are honest, and that the last mix-server is corrupted.

The fourth attack is novel and breaks the privacy of any given sender. It requires that the first and last mix-servers are corrupted. This attack breaks also "Flash Mix" by Jakobsson [15], including the fixed version given by Mitomo and Kurosawa [19].

1 Introduction

The notion of a mix-net was invented by Chaum [2], and further developed by a number of people. Properly constructed a mix-net enables a set of senders to send messages anonymously. A mix-net can be viewed as an electronic analog of a tombola; messages are put into envelopes, the envelopes are mixed, and finally opened. It is impossible to tell who sent any given message.

Informally the requirements on a mix-net are: correctness, privacy, robustness, availability, and efficiency. Correctness implies that the result is correct given that all mix-servers are honest. Privacy implies that if a fixed minimum number of mix-servers are honest privacy of the sender of a message is ensured. Robustness implies that if a fixed number of mix-servers are honest, then any attempt to cheat is detected, and defeated. Availability and efficiency are the general requirements on any system run on an open network.

A mix-net consists of a number of mix-servers, i.e. servers, that collectively execute a protocol. The basic idea of a mix-net is that each mix-server receives a list of encrypted messages, transforms them, using partial decryption or random re-encryption, reorders them, and then outputs the transformed and reordered list. It should be difficult to find an element in the input list, and an element in the output list that encrypts the same message. The use of several mix-servers allows a sender to trust a subset of the mix-servers to ensure privacy. Later constructions have mostly dealt with robustness, availability and efficiency, which are aspects ignored by Chaum.

1.1 Previous Work and Applications of Mix-Nets

The mixing paradigm has been used to accomplish privacy in many different scenarios. Chaum's original "anonymous channel" [2, 22] enables a sender to securely send mail to a receiver anonymously, and also to securely receive mail from this receiver without revealing the sender's identity. When constructing election schemes [2, 7, 24, 27, 21] the mix-net is used to ensure that the vote of a given voter can not be revealed. Also in the construction of electronic cash systems [14] mix-nets have been used to ensure privacy. Thus a mix-net is a useful primitive in constructing cryptographic protocols.

Abe gives an efficient construction of a general mix-net [1], and argues about its properties. Jakobsson has written (partly with Juels) a number of more general papers on the topic of mixing [13, 15, 16] also focusing on efficiency, of which the first appeared at the same time as Abe's construction.

There has been a breakthrough in the construction of zero-knowledge proofs of a correct shuffle recently. Furukawa and Sako [8], and Neff [20] respectively have both found efficient ways to design such proofs.

A promising approach to practical mixing is given by Golle et al. [10]. They combine a robustness test based partly on work by Jakobsson and Juels, with the notion of "double enveloping". The latter notion is introduced independently by Wikström [32], except that he uses different keys for the two layers, and a proof of knowledge of the inner most cleartext.

Desmedt and Kurosawa [4] describe an attack on a protocol by Jakobsson [13]. Similarly Mitomo and Kurosawa [19] exhibit a weakness in another protocol by Jakobsson [15]. Pfitzmann has given some general attacks on mix-nets [26, 25], and Michels and Horster give additional attacks in [18].

2 Review of "Optimistic Mixing for Exit-Polls"

We present a short review of the relevant parts of the protocol of Golle et al. [10]. The description given here is as close as possible to the original, but we avoid details irrelevant to our attacks, and change some notation to simplify the exposition of the attacks. For details we refer the reader to [10].

2.1 Participants and Setup

The protocol assumes the existence of a bulletin board on which each participant has a dedicated area on which only she can write. No participant can erase anything on the bulletin board, but all participants can read read everything.

The participants of the protocol are N *senders*, and a relatively small number of *mix-servers*, $M_1, \ldots, M_k$. Each sender encrypts its message, and writes it on the bulletin board. The mix-servers then execute the mix-net protocol.

The protocol employs an El Gamal [6] cryptosystem in a subgroup G_Q of prime order Q of the multiplicative group modulo a prime P, i.e. $\mathbb{Z}_P^*$. A private key x is generated by choosing $x \in \mathbb{Z}_Q$ uniformly and independently at random. The corresponding public key is (g, y), where g is a generator of G_Q, and $y = g^x$. Encryption of a message $m \in G_Q$ using the public key (g, y) is given by $E_{(g,y)}(m, r) = (g^r, y^r m)$, where r is chosen uniformly at random from $\mathbb{Z}_Q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by $D_x(u, v) = u^{-x} v = m$. The El Gamal system also has the re-encryptability[1] property, i.e. given (u, v) and the public key (g, y), anybody can "update" the randomness used in the encryption of m, by forming $(g^{r'} u, y^{r'} v) = (g^{r+r'}, y^{r+r'} m) = E_{(g,y)}(m, r + r')$.

In the setup stage each mix-server M_j is somehow given a random $x_j \in \mathbb{Z}_Q$, and $y_l = g^{x_l}$ for $l \neq j$. The value x_j is also shared with the other mix-servers using a threshold verifiable secret sharing (VSS) scheme[2]. Thus, if any mix-server M_j is deemed to be cheating the other mix-servers can verifiable reconstruct its private key x_j. The mix-servers can also compute $y = \prod_{j=1}^{k} y_j$, which gives a joint public key (g, y), with secret corresponding private key $x = \sum_{j=1}^{k} x_j$.

A practical advantage of the mix-net is that it can be used to execute several mix-sessions using the same set of keys, i.e. the El Gamal keys are not changed between mix-sessions. To be able to do this the proofs of knowledge below are bound to a mix-session identifier id that is unique to the current mix-session.

2.2 Sending a Message to the Mix-Net

A typical honest sender, Alice, computes the following to send a message m to the mix-net:

$$(u, v) = E_{(g,y)}(m), \quad w = h(u, v), \text{ and}$$
$$\alpha = [E_{(g,y)}(u), E_{(g,y)}(v), E_{(g,y)}(w)] = [(\mu_1, \mu_2), (\nu_1, \nu_2), (\omega_1, \omega_2)] \ ,$$

where $h : \{0,1\}^* \rightarrow G_Q$ is a hash function modeled by a random oracle. Then Alice computes a zero-knowledge proof of knowledge $\pi_{\mathrm{id}}(u, v, w)$, in the random oracle model of u, v and w, that depends on the current mix-session identifier id. Finally Alice writes $(\alpha, \pi_{\mathrm{id}}(u, v, w))$ on the bulletin board.

2.3 Execution of the Mix-Net

First the mix-servers remove any duplicate inputs to the mix-net, and discard input tuples that contain components not in the subgroup G_Q. Then the mix-servers discard all input tuples where the proof of knowledge is not valid for

[1] Related terms are "homomorphic property", or "malleability property".

[2] Golle et al. [10] discuss different variants for sharing keys, but we choose to present a simple variant, since it has no impact on our attacks.

the current mix-session. Let $L_0 = \{[(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]\}_{i=1}^N$ be the resulting list of triples of El Gamal pairs. The mixing then proceeds in the following stages.

First Stage: Re-Randomization and Mixing. This step proceeds as in all re-randomization mix-nets based on El Gamal. One by one, the mix-servers $M_1, \ldots, M_k$ randomize all the inputs and their order. (Note that the components of triples are not separated from each other during the re-randomization.) In addition, each mix-net must give a proof that the product of the plaintexts of all its inputs equals the product of the plaintexts of all its outputs. The protocol proceeds as follows.

1. Each mix-server M_j reads from the bulletin board the list L_{j-1} output by the previous mix-server.
2. The mix-server then chooses $r_{ji}, s_{ji}, t_{ji} \in \mathbb{Z}_Q$, for $i = 1, \ldots, N$, randomly and computes the re-randomized list:
$$\{[(g^{r_{ji}} a_{j-1,i}, y^{r_{ji}} b_{j-1,i}), (g^{s_{ji}} c_{j-1,i}, y^{s_{ji}} d_{j-1,i}), (g^{t_{ji}} e_{j-1,i}, y^{t_{ji}} f_{j-1,i})]\}_{i=1}^N \ .$$
The above list of triples is then randomly permuted, and the resulting list: $L_j = \{[(a_{j,i}, b_{j,i}), (c_{j,i}, d_{j,i}), (e_{j,i}, f_{j,i})]\}_{i=1}^N$ is written on the bulletin board.
3. Define $a_j = \prod_{i=1}^N a_{j,i}$, and define b_j, c_j, d_j, e_j, and f_j correspondingly. The mix-server proves in zero-knowledge that $\log_g a_j/a_{j-1} = \log_y b_j/b_{j-1}$, $\log_g c_j/c_{j-1} = \log_y d_j/d_{j-1}$, and $\log_g e_j/e_{j-1} = \log_y f_j/f_{j-1}$. This implies that $D_x(a_j, b_j) = D_x(a_{j-1}, b_{j-1})$, and similarly for the pairs (c_j, d_j) and (e_j, f_j), i.e. the component-wise product of the inner triples remain unchanged by the mix-server.

Remark 1. Since $\log_y b_j/b_{j-1} = \log_g a_j/a_{j-1} = \sum_{i=1}^N r_{ji}$, and M_j knows the latter sum, the proof in step 3) can be implemented by a zero-knowledge proof of knowledge in the random oracle model, and similarly for the pairs (c_j, d_j), and (e_j, f_j). Such proofs are similar to Schnorr signatures [28], and presented in several papers, e.g. Jakobsson [13].

Second Stage: Decryption of the Inputs.

1. A quorum of mix-servers jointly decrypt each triple of ciphertexts in L_k to produce a list L on the form $L = \{(u_i, v_i, w_i)\}_{i=1}^N$. Since the method used to do this is irrelevant to our attacks, we do not present it here.
2. All triples for which $w_i = h(u_i, v_i)$ are called *valid*.
3. Invalid triples are investigated according to the procedure described below. If the investigation proves that all invalid triples are *benign* (only senders cheated), we proceed to Step 4. Otherwise, the decryption is aborted, and we continue with the back-up mixing.
4. A quorum of mix-servers jointly decrypt the ciphertexts (u_i, v_i) for all valid triples. This successfully concludes the mixing. The final output is defined as the set of plaintexts corresponding to valid triples.

Special Step: Investigation of Invalid Triples. The mix-servers must reveal the path of each invalid triple through the various permutations. For each invalid triple, starting from the last server, each server reveals which of its inputs corresponds to this triple, and how it re-randomized this triple. One of two things may happen:

- **Benign case (only senders cheated):** if the mix-servers successfully produce all such paths, the invalid triples are known to have been submitted by users. The decryption resumes after the invalid triples have been discarded.
- **Serious case (one or more servers cheated):** if one or more servers fail to recreate the paths of invalid triples, these mix-servers are accused of cheating and replaced, and the mix-net terminates producing no output. In this case, the inputs are handed over to the back-up mixing procedure below.

Back-Up Mixing. The *outer-layer* encryption of the inputs posted to the mix-net is decrypted by a quorum of mix-servers. The resulting list of *inner-layer* ciphertexts becomes the input to a standard re-encryption mix-net based on El Gamal (using, for example Neff's scheme described in [20]). Then the output of the standard mix-net is given as output by the mix-net.

Remark 2. It is impossible to find two lists $\{(u_i, v_i)\}_{i=1}^N \neq \{(u_i', v_i')\}_{i=1}^N$ such that $\prod_{i=1}^N h(u_i, v_i) = \prod_{i=1}^N h(u_i', v_i')$, if the product is interpreted in a group where the discrete logarithm problem is hard. This is stated as a theorem by Wagner[3] [31], and appears as a lemma in Golle et al. [10].

Golle et al. [10] apply the lemma in a very clever way to provide robustness of their protocol. During the re-encryption and mixing stage, each mix-server proves in zero-knowledge that it leaves the component wise product $(\prod u_i, \prod v_i, \prod w_i)$, of the inner triples (u_i, v_i, w_i) unchanged, but individual triples may still be corrupted. Then invalid triples are traced back. This leaves only valid inner triples in the output and the proofs of knowledge of each server is used to conclude that the component wise product of these valid inner triples was left unchanged by the mix-net. The lemma is then applied to conclude that the set of valid triples in the output is identical to the set of valid triples hidden in the double encrypted input to the mix-net.

Unfortunately, our third attack in Section 3.3, shows that it is possible to cheat without changing the set of inner triples.

3 The Attacks

The goal of the adversary, who we in this paper call Eve, is to break the privacy of our typical honest sender Alice, and cheat without detection. Each of our attacks illustrates a separate weakness of the protocol.

The first two attacks are adaptations of the "relation attack", introduced by Pfitzmann [26, 25], to the setting with double enveloping. The idea of the

[3] Wagner credits Wei Dai with this observation.

"relation attack" is that to break the privacy of Alice, Eve computes a cryptotext of a message related to Alice's message. Then the mix-net is run as usual. The output of the mix-net contains two messages related in a way chosen by Eve. Some relations let Eve determine the message sent by Alice. The third attack is similar to the attack of Desmedt and Kurosawa [4] in that it exploits intermediate results of the protocol and fools a "product test". The fourth attack is novel.

3.1 First Attack: Honest Mix-Servers

We show that the adversary Eve can break the privacy of the typical sender Alice. All that is required is that Eve can send two messages to the mix-net, which is a natural assumption in most scenarios. In the following we use the notation for the cryptotext of Alice introduced in Section 2.2. Eve does the following:

1. Eve chooses δ and γ randomly in $\mathbb{Z}_Q$, and computes:

$$w_\delta = h(\mu_1^\delta, \mu_2^\delta),\ \alpha_\delta = (E_{(g,y)}(\mu_1^\delta), E_{(g,y)}(\mu_2^\delta), E_{(g,y)}(w_\delta)) \ , \text{ and}$$
$$w_\gamma = h(\mu_1^\gamma, \mu_2^\gamma),\ \alpha_\gamma = (E_{(g,y)}(\mu_1^\gamma), E_{(g,y)}(\mu_2^\gamma), E_{(g,y)}(w_\gamma)) \ .$$

   Then Eve computes corresponding proofs of knowledge $\pi_{\mathrm{id}}(\mu_1^\delta, \mu_2^\delta, w_\delta)$ and $\pi_{\mathrm{id}}(\mu_1^\gamma, \mu_2^\gamma, w_\gamma)$. This gives Eve two perfectly valid pairs $(\alpha_\delta, \pi_{\mathrm{id}}(\mu_1^\delta, \mu_2^\delta, w_\delta))$, $(\alpha_\gamma, \pi_{\mathrm{id}}(\mu_1^\gamma, \mu_2^\gamma, w_\gamma))$, that she sends to the bulletin board (possibly by corrupting two senders).
2. Eve waits until the mix-net has successfully completed its execution. During the execution of the mix-net, the mix-servers first jointly decrypt the "outer layer" of the double encrypted messages. After benign tuples have been removed, the result is a list of valid triples

$$((u_1, v_1, w_1), \ldots, (u_N, v_N, w_N)) \ . \tag{1}$$

   The final output of the mix-net is the result of decrypting each inner El Gamal pair (u_i, v_i) and results in a list of cleartext messages $(m_1, \ldots, m_N)$.
3. Eve computes the list $(m_1', \ldots, m_N') = (m_1^{\delta/\gamma}, \ldots, m_N^{\delta/\gamma})$, and then finds a pair (i, j) such that $m_i = m_j'$. From this she concludes that with very high probability $m_j = u^\gamma$. Then she computes $z = m_j^{1/\gamma}$, and finds a triple (u_l, v_l, w_l) in the list (1) such that $z = u_l$. Finally she concludes that with very high probability m_l was the message sent by Alice to the mix-net.

Remark 3. At additional computational cost it suffices for Eve to send 2 messages to break the privacy of K senders. Suppose Eve wants to break the privacy also of Bob who sent m' encrypted as $(u', v') = E_{(g,y)}(m')$, $w' = h(u', v')$, and $\alpha' = [E_{(g,y)}(u'), E_{(g,y)}(v'), E_{(g,y)}(w')] = [(\mu_1', \mu_2'), (\nu_1', \nu_2'), (\omega_1', \omega_2')]$. Then Eve performs the attack above with the change that she starts with a single pair $(\mu_1^\zeta {\mu'}_1, \mu_2^\zeta {\mu'}_2)$ for some randomly chosen ζ instead of the two distinct pairs (μ_1, μ_2), and (μ_1', μ_2') that would have given two "unrelated" attacks. The original third step of the attack first gives Eve $z = u^\zeta u'$. To finish the attack she finds a pair (l, l') such that $u_l^\zeta u_{l'} = z$, and concludes that with high probability Alice sent m_l, and Bob sent $m_{l'}$. The approach is generalized to higher dimensions in the natural way, to break the privacy of several senders (K must clearly be reasonably sized).

Why the Attack is Possible. The attack exploits two different flaws of the protocol. The first is that the sender of a message, e.g. Alice, proves only knowledge of the inner El Gamal pair (u, v) and the hash value $w = h(u, v)$, and not knowledge of the message m. The second flaw is that identical El Gamal keys are used for both the inner and outer El Gamal system.

Anybody can compute a single encrypted message $(\mu_1^\delta, \mu_2^\delta) = (g^{r\delta}, y^{r\delta} u^\delta) = E_{(g,y)}(u^\delta, r\delta)$ of a power u^δ of the first component u of the *inner* El Gamal pair (u, v) of the triple α sent by Alice. Anybody can also compute a proof of knowledge of $(\mu_1^\delta, \mu_2^\delta)$ and $w_\delta = h(\mu_1^\delta, \mu_2^\delta)$ (and similarly for $(\mu_1^\gamma, \mu_2^\gamma)$ and ω_γ).

The first flaw allows Eve to input triples of El Gamal pairs with such proofs of knowledge as input to the mix-net. The second flaw allows Eve to use the mix-net to decrypt $(\mu_1^\delta, \mu_2^\delta)$, and thus get her hands on u^δ (and similarly for u^γ). Eve can then identify (u, v) as the inner El Gamal pair of Alice, and break her privacy.

3.2 Second Attack: Different Keys and Corrupt Mix-Server

Suppose we change the protocol slightly by requiring that the mix-servers generate separate keys for the outer and inner El Gamal systems, to avoid the first attack of Section 3.1. That is, we assume that there are two different key pairs $((g, y_{\mathrm{in}}), x_{\mathrm{in}})$ and $((g, y_{\mathrm{out}}), x_{\mathrm{out}})$, for the inner and outer El Gamal layers respectively, and also assume that these keys has been shared similarly as the original key pair $((g, y), x)$. This is the type of double enveloping proposed by Wikström [32]. For the second attack to succeed we need some additional assumptions.

Unclear Details and Additional Assumptions. We start by quoting Section 5, under "Setup." point 4 of Golle et al. [10], that presents the proof of knowledge $\pi_{\mathrm{id}}(u, v, w)$ of the sender Alice:

> 4. This proof of knowledge should be bound to a unique mix-session identifier to achieve security over multiple invocations of the mix. Any user who fails to give the proof is disqualified, and the corresponding input is discarded.

If different keys are used for each mix-session then the above makes no sense, since the proof of knowledge of u, v and w already depends on the public key of the outer El Gamal system. There is clearly practical value in not changing keys between mix-sessions.

We assume that the keys are not changed between mix-sessions, even if a mix-server is found to be cheating. If a mix-server is found to be cheating its shared keys are instead reconstructed by the remaining mix-servers using the VSS-scheme, and in later mix-sessions the actions of the cheating mix-server are performed either in the open (the details of this does not matter to our attack). Under these assumptions we can give an attack on the protocol.

The original paper of Golle et al. [10] does not explicitly say if the discovery of the corrupted mix-server results in a new execution of the key generation protocol. Apparently the intention of the authors is to let the remaining mix-servers generate a new set of keys if any cheating is discovered [11].

The attack is interesting even though this interpretation is not the one intended by the authors, since it shows the importance of explicitly defining all details of protocols, and highlights some issues with running several concurrent mix-sessions using the same set of keys.

The Attack. Apart from the above assumptions, the attack only requires that the first mix-server in the mix-chain is corrupted. The attack is employed during two mix-sessions using the same keys, and the corrupted mix-server is identified as a cheater in the first mix-session. In the following we describe the actions of Eve during the first, and second mix-sessions, respectively.

The First Mix-Session. We assume that Alice, and some other arbitrary sender Bob, have sent inputs to the mix-net (and use the notation of Remark 3 for the input of Bob). Eve corrupts M_1. It then replaces α and α' with: $[E_{y_{\mathrm{out}}}(u), E_{y_{\mathrm{out}}}(v), E_{y_{\mathrm{out}}}(w')]$, and $[E_{y_{\mathrm{out}}}(u'), E_{y_{\mathrm{out}}}(v'), E_{y_{\mathrm{out}}}(w)]$ respectively, in its input list, i.e. the third components of the two triples are shifted. Then Eve forces M_1 to simulate a completely honest mix-server on the resulting altered list $L_0' = \{[(a_{0,i}', b_{0,i}'), (c_{0,i}', d_{0,i}'), (e_{0,i}', f_{0,i}')]\}_{i=1}^N$. Note that $\prod_{i=1}^N a_{0,i}' = a_0$, and similarly for b_0, c_0, d_0, e_0, and f_0. Thus, the simulated honest mix-server outputs perfectly valid zero-knowledge proofs that the product of the inner triples are unchanged.

At the end of the mixing the mix-servers verify the tuples, and discover the invalid tuples (u, v, w') and (u', v', w). These tuples are traced back all the way to the first mix-server, which is revealed as a cheater. In this process Eve is able to link Alice to (u, v) (and Bob to (u', v')). Finally the honest mix-servers finish the protocol by using the general constructions based on the work by Neff [20] as in Golle et al. [10].

The Second Mix-Session. To allow the mix-net to execute a second mix-session using the same set of keys, the cheater's key is reconstructed, and revealed by a quorum of the mix-servers.

To determine the contents of the El Gamal pair (u, v), Eve waits for the second mix-session using the same set of keys. Then she uses a "relation attack" [26, 25, 13] in the second mix-session to decrypt (u, v), i.e. Eve does the following:

1. Eve chooses δ and γ randomly in $\mathbb{Z}_Q$, and computes:
   $w_\delta = h(u^\delta, v^\delta)$, $\alpha_\delta = (E_{y_{\mathrm{out}}}(u^\delta), E_{y_{\mathrm{out}}}(v^\delta), E_{y_{\mathrm{out}}}(w_\delta))$, and
   $w_\gamma = h(u^\gamma, v^\gamma)$, $\alpha_\gamma = (E_{y_{\mathrm{out}}}(u^\gamma), E_{y_{\mathrm{out}}}(v^\gamma), E_{y_{\mathrm{out}}}(w_\gamma))$.

Then Eve computes corresponding proofs of knowledge $\pi_{\mathrm{id}}(u^\delta, v^\delta, w_\delta)$ and $\pi_{\mathrm{id}}(u^\gamma, v^\gamma, w_\gamma)$. This gives Eve two perfectly valid pairs $(\alpha_\delta, \pi_{\mathrm{id}}(u^\delta, v^\delta, w_\delta))$, $(\alpha_\gamma, \pi_{\mathrm{id}}(u^\gamma, v^\gamma, w_\gamma))$, that she sends to the bulletin board (possibly by corrupting two senders).

2. Eve waits until the mix-net has successfully completed its execution. The final output of the mix-net is a list of cleartext messages $(m_1, \ldots, m_N)$.
3. Note that $m_i = m^\delta$ and $m_j = m^\gamma$ for some i and j. Eve computes $\delta\gamma^{-1} \bmod Q$, computes the list $(m'_1, \ldots, m'_N) = (m_1^{\delta/\gamma}, \ldots, m_N^{\delta/\gamma})$, and finally finds a pair (i, j) such that $m_i = m'_j$. Then she concludes that with high probability $m_j^{1/\gamma}$ is the message sent by Alice to the mix-net in the *first* mix-session.

Remark 4. The attack is easily generalized to break the privacy of several senders by using a circular shift of the third components during the first mix-session. It is also easy to see that Remark 3 can be applied to reduce the number of messages sent by Eve during the second mix-session.

Why the Attack is Possible. The attack exploits that the sender of a message only proves knowledge of the inner triple (u, v, w). At the cost of detected cheating Eve finds a (u, v) corresponding to Alice, and then uses the second mix-session as a decryption oracle, and get her hands on m.

A Note on Concurrent Mix-Sessions. Ignoring the other attacks, a simple countermeasure to the second attack above, is to stipulate that if a cheating mix-server is identified, new keys must be generated for the next mix-session.

A disadvantage of this countermeasure is that the mix-net can not be allowed to execute several concurrent mix-sessions using the same keys. If one mix-session is still receiving messages while another mix-session discovers a cheating mix-server, the second attack of Section 3.2 can still be applied. The problem is not solved by running the back-up mix-net of Neff [20] on all mix-sessions using the same keys at this point.

This problem of concurrency may seem academic, since in most election scenarios it is not very cumbersome to have different keys for each mix-session, but in future applications of mix-nets it may be useful to run several concurrent mix-sessions using the same keys.

3.3 Third Attack: Honest Senders and Two Corrupt Mix-Servers

In this section we assume that all senders, and all mix-servers, except the last mix-server M_k, are honest. The last mix-server M_k is corrupted by the adversary Eve, and performs the attack. The attack breaks both robustness and privacy.

To simplify notation we write $L_0 = \{\alpha_i\}_{i=1}^N$ for the input list, where we define $\alpha_i = [(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]$ to be the tuple sent by sender S_i. Instead of following the protocol, M_k proceeds as follows in the first stage.

1. It computes the 6-tuple: $(a', b', \ldots, f') = (a_{k-1}/a_0, b_{k-1}/b_0, \ldots, f_{k-1}/f_0)$, and the tuple $\alpha'_1 = [(a'a_{0,1}, b'b_{0,1}), (c'c_{0,1}, d'd_{0,1}), (e'e_{0,1}, f'f_{0,1})]$.

2. Then it forms the list $L'_{k-1} = \{\alpha'_1, \alpha_2, \ldots, \alpha_N\}$, i.e. a copy of L_0 with the first tuple α_1 replaced by α'_1.
3. When M_k is supposed to re-randomize and permute the output L_{k-1} of M_{k-1}, it instead simulates the actions of an honest mix-server on the corrupted input L'_{k-1}. The output list written to the bulletin board by the simulated mix-server is denoted L_k.
4. Eve then waits until the inner layer has been decrypted, and uses her knowledge of the permutation that relates L_k to L_0, to break privacy of all senders.

We show that the attack goes undetected, i.e. the mix-servers decrypt the inner layer. This implies that the attack succeeds.

Firstly, consider the proof of knowledge that M_k produces during the re-encryption and mixing stage. Define $a'_{k-1} = (a' a_{0,1}) \prod_{i=2}^{N} a_{0,i}$, and similarly for for b'_{k-1}, c'_{k-1}, d'_{k-1}, e'_{k-1}, and f'_{k-1}. In step 3 above the simulated honest mix-server outputs proofs of knowledge of the logarithms: $\log_g a_k / a'_{k-1} = \log_y b_k / b'_{k-1}$, $\log_g c_k / c'_{k-1} = \log_y d_k / d'_{k-1}$, and $\log_g e_k / e'_{k-1} = \log_y f_k / f'_{k-1}$. By construction we have that $a'_{k-1} = (a' a_{0,1}) \prod_{i=2}^{N} a_{0,i} = a' \prod_{i=1}^{N} a_{0,i} = \frac{a_{k-1}}{a_0} a_0 = a_{k-1}$, and similarly for b_{k-1}, c_{k-1}, d_{k-1}, e_{k-1}, and f_{k-1}. This implies that the proof of knowledge produced by M_k is deemed valid by the other mix-servers.

Secondly, consider the investigation of invalid inner triples. Tracing back invalid triples is difficult to M_k, since it does not know the re-encryption exponents and the permutation relating L_{k-1} and L_k. We show that there are no invalid inner triples. Suppose we define the sums $r = \sum_{j=1}^{k-1} \sum_{i=1}^{N} r_{ji}$, $s = \sum_{j=1}^{k-1} \sum_{i=1}^{N} s_{ji}$, and $t = \sum_{j=1}^{k-1} \sum_{i=1}^{N} t_{ji}$, i.e. we form the sum of all re-encryption exponents used by all mix-servers except the last, for the first second and third El Gamal pairs respectively. Since all mix-servers except M_k are honest, we have $(a', b', c', d', e', f') = (g^r, y^r, g^s, y^s, g^t, y^t)$, which implies that α'_1 is a valid re-encryption of α_1. Thus M_k does not corrupt any inner triple by simulating an honest mix-server on the input L'_{k-1}. Since all senders are honest and the set of inner triples hidden in L_0 and L'_{k-1} are identical, there are no invalid inner triples. Thus cheating is not detected, and robustness is broken.

We conclude that the mix-servers decrypt the inner triples, i.e. the privacy of *all* senders is broken.

Why the Attack is Possible. The third attack above exploits that the last mix-server M_k is not forced to take the output L_{k-1} of the next to last mix-server as input. This allows M_k to use a slightly modified version of L_0 instead, which breaks the privacy of all senders.

3.4 Fourth Attack: Two Corrupt Mix-Servers

In this section we assume that the first and last mix-servers, M_1 and M_k are corrupted. We give a novel attack that breaks the privacy of any given sender Alice. Let $L_0 = \{\alpha_i\}_{i=1}^{N}$, where $\alpha_i = [(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]$. Without loss we let α_1 and α_2 be the tuples sent by Alice and Bob respectively. Let $\xi \in \mathbb{Z}_P^* \backslash G_Q$. Eve corrupts M_1 and M_k, and they proceed as follows.

1. M_1 computes the elements $\alpha'_1 = [(\xi a_{0,1}, b_{0,1}), (c_{0,1}, d_{0,1}), (e_{0,1}, f_{0,1})]$ and $\alpha'_2 = [(\xi^{-1} a_{0,2}, b_{0,2}), (c_{0,2}, d_{0,2}), (e_{0,2}, f_{0,2})]$, and forms the modified list $L'_0 = \{\alpha'_1, \alpha'_2, \alpha_3, \ldots, \alpha_N\}$. Then M_1 simulates an honest mix-server on input L'_0.
2. M_k simulates an honest mix-server on input L_{k-1}, but it does not write the output L_k of this simulation on the bulletin board. Let $L_k = \{\beta_i\}_{i=1}^{N}$, where $\beta_i = [(a_{k,i}, b_{k,i}), (c_{k,i}, d_{k,i}), (e_{k,i}, f_{k,i})]$. M_k finds $l, l' \in \{1, \ldots, N\}$ such that $a_{k,l}^{Q} = \xi^{Q}$ and $a_{k,l'}^{Q} = \xi^{-Q}$.
   Then it computes the tuples $\beta'_l = [(\xi^{-1} a_{k,l}, b_{k,l}), (c_{k,l}, d_{k,l}), (e_{k,l}, f_{k,l})]$ and $\beta'_{l'} = [(\xi a_{k,l'}, b_{k,l'}), (c_{k,l'}, d_{k,l'}), (e_{k,l'}, f_{k,l'})]$, and writes L'_k, where L'_k defined by $L'_k = \{\beta_1, \ldots, \beta_{l-1}, \beta'_l, \beta_{l+1}, \ldots, \beta_{l'-1}, \beta'_{l'}, \beta_{l'+1}, \ldots, \beta_N\}$, on the bulletin board.
3. The mix-net outputs $(m_1, \ldots, m_N)$, and Eve concludes that Alice sent m_l.

Since all mix-servers except M_1 and M_k are honest, there exists $l, l' \in \{1, \ldots, N\}$ and $r, r' \in \mathbb{Z}_Q$ such that $a_{k,l} = g^{r} \xi a_{0,1}$ and $a_{k,l'} = g^{r'} \xi^{-1} a_{0,2}$. This implies that $a_{k,l}^{Q} = \xi^{Q} (g^{r} a_{0,1})^{Q} = \xi^{Q}$ and $a_{k,l'}^{-Q} = \xi^{-Q}$, since $\beta^{Q} = 1$ for any $\beta \in G_Q$. We have $\xi^{Q} \neq 1 \neq \xi^{-Q}$, since the order of ξ does not divide Q. On the other hand we have $a_{k,i}^{Q} = 1$ for $i \neq l, l'$, since $a_{k,i} \in G_Q$ for $i \neq l, l'$. Thus Eve successfully identifies Alice's cryptotext, if the cheating of M_1 and M_k is not detected.

Clearly, b_1, c_1, d_1, e_1, and f_1 are not changed by replacing L_0 with L'_0 in step 1. Neither is a_1, since $(\xi a_{0,1})(\xi^{-1} a_{0,2}) \prod_{i=3}^{N} a_{0,i} = \prod_{i=1}^{N} a_{0,i} = a_1$. Similarly, b_k, c_k, d_k, e_k, and f_k are not changed by replacing L_k with L'_k in step 2. Neither is a_k, since $(\xi^{-1} a_{k,l})(\xi a_{k,l'}) \prod_{i \neq l,l'}^{N} a_{k,i} = \prod_{i=1}^{N} a_{k,i}$. Similarly as in the second attack of Section 3.3, we conclude that the proofs of knowledge produced by M_1 and M_k are deemed valid by the other mix-servers. If we assume that Alice and Bob are honest, their inner triples are never traced back, and cheating is not detected.

If $\xi = \xi^{-1}$, Eve can only conclude that Alice sent a message from the set $\{m_l, m_{l'}\}$. This already breaks the privacy of Alice, but "Bob" may also be chosen to be some corrupted sender, and Alice's message identified uniquely.

Why the Attack is Possible. The third attack above exploits the fact that a mix-server M_j never verifies that all elements in its input L_{j-1} reside in G_Q. This is used by M_1 to "tag" elements in the input list, which let them be identified by the last mix-server M_k.

Note that it is very common in protocols to implicitly assume that elements are in G_Q. We believe that several other protocols are vulnerable to the malicious use of elements in $\mathbb{Z}_P^* \backslash G_Q$.

4 Further Applications and Future Work

The attacks given in this paper work also if $\mathbb{Z}_P^* \supset G_Q$ are replaced by similarly related elliptic curves.

The attacks of sections 3.1, 3.2, and 3.3 all exploit the particular structure of the protocol of Golle et al. [10]. We have found no other protocol vulnerable to these attacks. In particular the protocol by Jakobsson [15] with similar structure, is not vulnerable to the attack of Section 3.2.

The novel attack of Section 3.4 can be applied to break privacy of "Flash Mix" by Jakobsson [15], and the fixed protocol proposed[4] by Mitomo and Kurosawa [19]. We sketch how this is done in Appendix A. We are currently investigating what other mix-nets are vulnerable to variants of the novel attack.

As explained within this paper, our attacks exploit four different flaws of the protocol: the lack of an appropriate proof of knowledge, the use of identical keys for the inner and outer system, the possibility for the last mix-server to use corrupt input in the re-randomization and mixing stage without detection, and the lack of verification that elements in the input to a mix-server reside in G_Q.

It is not hard to come up with countermeasures for our particular attacks. However, it is not clear that countering our particular attacks suffices to make the protocol secure.

An interesting open question is what other protocols are vulnerable to attacks based on malicious use of elements in $\mathbb{Z}_P^* \backslash G_Q$.

5 Conclusion

We have presented several practical attacks for the mix-net recently proposed by Golle, Zhong, Boneh, Jakobsson, and Juels [10], claimed secure by the authors. In particular we have shown that the privacy of any given sender can be broken *without* corrupting any mix-server, and that the privacy of *all* senders can be broken if the last mix-server is corrupted. Each of our attacks illustrates a separate weakness of the protocol.

Furthermore, one of the attacks is novel and breaks privacy also of the mix-net by Jakobsson [15], including the fixed protocol given by Mitomo and Kurosawa [19].

The relatively large number of attacks for mix-nets, presented here and in other papers [25, 25, 18, 4, 19], suggest that we should be careful with unproven claims of security for mix-nets. Subtle mistakes on details can make a mix-net completely insecure.

6 Acknowledgements

I am grateful to Gunnar Sjödin, Torsten Ekedahl, and Björn Grönvall, since most of the results in this paper would have been far fetched without the many discussions we have had on a previous mix-net protocol by Wikström [32]. I am also very grateful to Johan Håstad for contributing a missing piece of the fourth attack, and in general for his support and advice.

[4] We note that their proposal is given without security claims.

References

1. M. Abe, *Universally Verifiable mix-net with Verification Work Independent of the Number of Mix-centers*, Eurocrypt '98, pp. 437-447, LNCS 1403, 1998.
2. D. Chaum, *Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms*, Communications of the ACM - CACM '81, Vol. 24, No. 2, pp. 84-88, 1981.
3. R. Cramer, V. Shoup, *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Crypto '98, pp. 13-25, LNCS 1462, 1998.
4. Y. Desmedt, K. Kurosawa, *How to break a practical MIX and design a new one*, Eurocrypt 2000, pp. 557-572, LNCS 1807, 2000.
5. D. Dolev, C. Dwork, M. Naor, *Non-Malleable Cryptography*, In Proceedings of the 23:rd ACM Symposium on Theory of Computing - STOC '91, pp. 542–552, 1991.
6. T. El Gamal, *A Public Key Cryptosystem and a Signiture Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469-472, 1985.
7. A. Fujioka, T. Okamoto and K. Ohta, *A practical secret voting scheme for large scale elections*, Auscrypt '92, LNCS 718, pp. 244-251, 1992.
8. J. Furukawa, K. Sako, *An efficient scheme for proving a shuffle*, Crypto 2001, LNCS 2139, pp. 368-387, 2001.
9. S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences (JCSS), Vol. 28, No. 2, pp. 270-299, 1984.
10. Golle, Zhong, Boneh, Jakobsson, Juels, *Optimistic Mixing for Exit-Polls*, Asiacrypt 2002, LNCS, 2002.
11. Golle, Zhong, Boneh, Jakobsson, Juels, *Private Communication*, 16 October 2002.
12. M. Hirt, K. Sako, *Efficient Reciept-Free Voting Based on Homomorphic Encryption*, Eurocrypt 2000, LNCS 1807, pp. 539-556, 2000.
13. M. Jakobsson, *A Practical Mix*, Eurocrypt '98, LNCS 1403, pp. 448-461, 1998.
14. M. Jakobsson, D. M'Raihi, *Mix-based Electronic Payments*, In Proceedings of the 5:th Workshop on Selected Areas in Cryptography - SAC'98, LNCS 1556, pp. 157-173, 1998.
15. M. Jakobsson, *Flash Mixing*, In Proceedings of the 18:th ACM Symposium on Principles of Distributed Computing - PODC '98, pp. 83-89, 1998.
16. M. Jakobsson, A. Juels, *Millimix: Mixing in small batches*, DIMACS Techical report 99-33, June 1999.
17. S. Micali, C. Rackoff, B. Sloan, *The notion of security for probabilistic cryptosystems*, SIAM Journal of Computing, Vol. 17, No. 2, pp. 412-426, 1988.
18. M. Michels, P. Horster, *Some remarks on a reciept-free and universally verifiable Mix-type voting scheme*, Asiacrypt '96, pp. 125-132, LNCS 1163, 1996.
19. M. Mitomo, K. Kurosawa, *Attack for Flash MIX*, Asiacrypt 2000, pp. 192-204, LNCS 1976, 2000.
20. A. Neff, *A verifiable secret shuffle and its application to E-Voting*, In Proceedings of the 8:th ACM Conference on Computer and Communications Security - CCS 2001, pp. 116-125, 2001.
21. V. Niemi, A. Renvall, *How to prevent buying of votes in computer elections*, Asiacrypt'94, LNCS 917, pp. 164-170, 1994.
22. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, *Fault Tolerant Anonymous Channel*, Information and Communications Security - ICICS '97, pp. 440-444, LNCS 1334, 1997.

23. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Eurocrypt '99, LNCS 1592, pp. 223-238, 1999.
24. C. Park, K. Itoh, K. Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Eurocrypt '93, LNCS 765, pp. 248-259, 1994.
25. B. Pfitzmann, *Breaking an Efficient Anonymous Channel*, Eurocrypt '94, LNCS 950, pp. 332-340, 1995.
26. B. Pfitzmann, A. Pfitzmann, *How to break the direct RSA-implementation of mixes*, Eurocrypt '89, LNCS 434, pp. 373-381, 1990.
27. K. Sako, J. Killian, *Reciept-free Mix-Type Voting Scheme*, Eurocrypt '95, LNCS 921, pp. 393-403, 1995.
28. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, No. 4, pp. 161-174, 1991.
29. C. Schnorr, M. Jakobsson, *Security of Signed El Gamal Encryption*, Asiacrypt 2000, LNCS 1976, pp. 73-89, 2000.
30. Yiannis Tsiounis, Moti Yung, *On the Security of El Gamal based Encryption*, International workshop on Public Key Cryptography, LNCS 1431, pp. 117–134, 1998.
31. D. Wagner *A Generalized Birthday Problem*, Crypto 2002, LNCS 2442, pp. 288-304, 2002.
32. D. Wikström, *An Efficient Mix-Net*, Swedish Institute of Computer Science (SICS) Technical Report T2002:21, ISSN 1100-3154, SICS-T-2002/21-SE, 2002. An implementation of this protocol was demonstrated during the Conference of the Swedish Research Institute for Information Technology (SITI), februari 7, 2002.

A Attack for "Flash Mix"

In this section we assume that the reader is familiar with the protocol by Jakobsson [15], and sketch how this protocol can be broken by a natural adaptation of the novel attack of Section 3.3.

Computations in this protocol are performed in a similar group as in Golle et al. [10]. Thus let $\xi \in \mathbb{Z}_P^* \backslash G_Q$.

The attack is employed during the "second re-encryption". The adversary corrupts the first and the last mix-servers, M_1 and M_k, in the mix-chain. During the "second re-encryption", M_1 "tags" two arbitrary El Gamal pairs in its input by multiplying their first components with ξ and ξ^{-1} respectively (similarly as in Section 3.3). Then the honest mix-servers perform their re-encryption and mixing. When the last mix-server M_k is about to re-encrypt and mix the output of the previous mix-server M_{k-1}, it identifies the "tagged" El Gamal pairs, removes the "tags" by multiplying the first components by ξ^{-1} and ξ respectively, and then finally re-encrypts and mix the resulting list honestly.

After the verification of the "first re-encryption", this breaks the privacy of some randomly chosen sender, if the cheating goes undetected. Although the attack is weak, it does break privacy.

Cheating is detected if one of two things happen; the adversary by chance chooses a "dummy element" that is later traced back through the mix-chain, or if M_1 or M_k fails to play its part in the computation of the "relative permutations" correctly. The first event is very unlikely since by construction there are very few "dummy elements". Since the "tags" are removed by M_k, and both M_1 and M_k follow the protocol honestly except for the use of the tags, they can easily play their part of the computation of the "relative permutations" correctly. Thus the cheating is not detected and the privacy of the protocol is broken.

It is easy to see that the changes introduced by Mitomo and Kurosawa [19] do not counter the above attack.