Breaking and Mending Resilient Mix-nets

Lan Nguyen and Rei Safavi-Naini

School of IT and CS
University of Wollongong
Wollongong 2522
Australia
email: [ldn01,rei]@uow.edu.au

Outline

- Mix-net description and its requirements
- Cryptographic tools for discussed mix-nets
- Furukawa-Sako mix-net[1] and Millimix[2]
- Attacking Furukawa-Sako scheme and Millimix
- Countermeasures and their efficiency and security analysis.

Mix-net

Mix-net protects privacy of messages in network communication. A mix-net consists of a set of mix servers, each receiving as input a list of ciphertexts and outputting either a permuted list of the re-encrypted ciphertexts, or a permuted list of the corresponding plaintexts.
Mix-net participants:

- *Users* send messages to mix-net.
- *Mix servers* perform mixing of the input messages and produce an output, which is used as input to other mix-servers.

- *Verifier* verifies correctness of the mix-net operation.
- *Bulletin board* is a shared memory where all participants have read access to and can append messages after being authenticated. It simulates an authenticated broadcast channel.
- *Adversary* tries to compromise resiliency of the mix-net. We assume *static adversary*.

Mix-net Requirements

A mix-net is *resilient* if it satisfies *privacy*, *robustness* and *verifiability*.

- *privacy:* the adversary cannot output a pair of input and the corresponding output with probability non-negligibly greater than random guess.
- *verifiability:* the verification can detect and reveal the identities of the cheating servers with overwhelming probability. If only publicly available information is used, the mix-net is called *universally verifiable.*
- *robustness:* ensures that the probability of producing incorrect output is negligibly less than 1.

Cryptographic tools

El Gamal encryption p and q are primes, $p = 2kq + 1$, g is a generator of subgroup G_q of order q in Z_p^*. Private key is $x \in Z_q$, public key is (y, g) where $y = g^x$.
A ciphertext of message $m \in G_q$ is (α, β) where $\alpha = my^s$, $\beta = g^s$, $s \in_R Z_q$. The plaintext is computed as $m := \alpha/\beta^x$. A re-encryption of ciphertext (α, β) is $(\alpha \times y^r, \beta \times g^r)$, where $r \in_R Z_q$.
Schnorr identification $\mathcal{P}$ shows knowledge of private key x to $\mathcal{V}$

1. $\mathcal{P} \longrightarrow \mathcal{V}$: a commitment $w = g^e$, where $e \in_R Z_q$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge $c \in_R Z_q$
3. $\mathcal{P} \longrightarrow \mathcal{V}$: a response $s = e + cx \bmod q$

$\mathcal{V}$ then verifies that $g^s = wy^c$.

Disjunctive Schnorr identification $\mathcal{P}$ shows he knows one of private keys x_1 or x_2 to $\mathcal{V}$. Assume $\mathcal{P}$ possesses x_1.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: two commitments $w_1 = g_1^{e_1}$, $w_2 = g_2^{s_2} y_2^{-c_2}$, where $e_1, e_2, c_2, s_2 \in_R Z_q$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge $c \in_R Z_q$
3. $\mathcal{P} \longrightarrow \mathcal{V}$: responses $s_1 = e_1 + c_1 x_1 \bmod q$, s_2, $c_1 = c \oplus c_2$, c_2

$\mathcal{V}$ then checks if $g_i^{s_i} = w_i y_i^{c_i}$ for $i \in \{1, 2\}$.

Pairwise permutation network A *pairwise permutation network* is a permutation that is constructed from switching gates and requires $n \log_2 n - n + 1$ switching gates. A *switching gate* is a permutation for two input items.

Permutation Matrix A matrix $(A_{ij})_{n \times n}$ is a permutation matrix $\Leftrightarrow \exists\ \phi$ so that $\forall i, j \in \{1, ..., n\}$

$$A_{ij} = \begin{cases} 1 \bmod q & \text{if } \phi(i) = j \\ 0 \bmod q & \text{otherwise} \end{cases}$$

Theorem 1 $(A_{ij})_{n \times n}$ is a permutation matrix $\Leftrightarrow \forall i, j, k \in \{1, ..., n\}$

$$\sum_{h=1}^{n} A_{hi} A_{hj} = \begin{cases} 1 \bmod q & \text{if } i = j \\ 0 \bmod q & \text{otherwise} \end{cases} \tag{1}$$

$$\sum_{h=1}^{n} A_{hi} A_{hj} A_{hk} = \begin{cases} 1 \bmod q & \text{if } i = j = k \\ 0 \bmod q & \text{otherwise} \end{cases} \tag{2}$$

Furukawa-Sako01 Mix-net

Input to a mix-server is El Gamal ciphertexts $\{(g_i, m_i)|i = 1, ..., n\}$ encrypted by (y, g). Output is $\{(g'_i, m'_i)|i = 1, ..., n\}$
The mix-server proves knowledge of a permutation matrix $(A_{ij})_{n\times n}$ and $\{r_i|i = 1, ..., n\}$

$$g'_i = g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}} \tag{3}$$

$$m'_i = y^{r_i} \prod_{j=1}^{n} m_j^{A_{ji}} \tag{4}$$

Based on Theorem 1, this can be done by proving:

- $\{g'_i\}$ can be expressed as (3) using a matrix satisfying (1).
- $\{g'_i\}$ can be expressed as (3) using a matrix satisfying (2).
- The matrix and $\{r_i\}$ in these statements are the same.
- For each (g'_i, m'_i), the same r_i and $\{A_{ij}\}$ is used.

Furukawa-Sako01 Verification Protocol

Suppose $\{\tilde{g}, \tilde{g_1}, ..., \tilde{g_n}\}$ so that under discrete logarithm assumption, infeasible to obtain $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^n \tilde{g_i}^{a_i} = 1$.

1. $\mathcal{P}$ generates: $\delta, \rho, \tau, \alpha, \alpha_i, \lambda, \lambda_i \in_R Z_q, i = 1, ..., n$

2. $\mathcal{P}$ computes:

$$t = g^\tau, v = g^\rho, w = g^\delta, u = g^\lambda, u_i = g^{\lambda_i}, i = 1, ..., n$$

$$\tilde{g}_i{}' = \tilde{g}^{r_i} \prod_{j=1}^{n} \tilde{g}_j{}^{A_{ji}}, i = 1, ..., n \tag{5}$$

$$\tilde{g}' = \tilde{g}^{\alpha} \prod_{j=1}^{n} \tilde{g}_j{}^{\alpha_j} \tag{6}$$

$$g' = g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j} \tag{7}$$

$$m' = y^{\alpha} \prod_{j=1}^{n} m_j^{\alpha_j} \tag{8}$$

$$\dot{t}_i = g^{\sum_{j=1}^{n} 3\alpha_j A_{ji} + \tau\lambda_i}, i = 1, ..., n \tag{9}$$

$$\dot{v}_i = g^{\sum_{j=1}^{n} 3\alpha_j^2 A_{ji} + \rho r_i}, i = 1, ..., n \tag{10}$$

$$\dot{v} = g^{\sum_{j=1}^{n} \alpha_j^3 + \tau\lambda + \rho\alpha} \tag{11}$$

$$\dot{w}_i = g^{\sum_{j=1}^{n} 2\alpha_j A_{ji} + \delta r_i}, i = 1, ..., n \tag{12}$$

$$\dot{w} = g^{\sum_{j=1}^{n} \alpha_j^2 + \delta\alpha} \tag{13}$$

3. $\mathcal{P} \longrightarrow \mathcal{V}$:
   $t, v, w, u, \{u_i\}, \{\tilde{g}_i{}'\}, \tilde{g}', g', m', \{\dot{t}_i\}, \{\dot{v}_i\}, \dot{v}, \{\dot{w}_i\}, \dot{w}, i = 1, ..., n$

4. $\mathcal{P} \longleftarrow \mathcal{V}$: challenges $\{c_i | i = 1, ..., n\}$, $c_i \in_U Z_q$

5. $\mathcal{P} \longrightarrow \mathcal{V}$:

$$\begin{aligned} s &= \sum_{j=1}^{n} r_j c_j + \alpha \\ s_i &= \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \bmod q, i = 1, ..., n \\ \lambda' &= \sum_{j=1}^{n} \lambda_j c_j^2 + \delta \bmod q \end{aligned}$$

6. $\mathcal{V}$ verifies:

$$\tilde{g}^s \prod_{j=1}^{n} \tilde{g_j}^{s_j} = \tilde{g}' \prod_{j=1}^{n} \tilde{g_j}'^{c_j} \tag{14}$$

$$g^s \prod_{j=1}^{n} g_j^{s_j} = g' \prod_{j=1}^{n} g_j'^{c_j} \tag{15}$$

$$y^s \prod_{j=1}^{n} m_j^{s_j} = m' \prod_{j=1}^{n} m_j'^{c_j} \tag{16}$$

$$g^{\lambda'} = u \prod_{j=1}^{n} u_j^{c_j^2} \tag{17}$$

$$t^{\lambda'} v^s g^{\sum_{j=1}^{n}(s_j^3 - c_j^3)} = \dot{v} \prod_{j=1}^{n} \dot{v_j}^{c_j} \dot{t_j}^{c_j^2} \tag{18}$$

$$w^s g^{\sum_{j=1}^{n}(s_j^2 - c_j^2)} = \dot{w} \prod_{j=1}^{n} \dot{w_j}^{c_j} \tag{19}$$

Intuition

- (5),(6),(7),(8),(14), (15) and (16) show prover's knowledge of matrix (A_{ij}) and $\{r_i\}$ satisfying (3) and (4)
- (9),(10),(11),(17) and (18) show (A_{ij}) satisfying (2)
- (12),(13),(19) show (A_{ij}) satisfying (1)
- based on Theorem 1, (A_{ij}) is a permutation matrix

Millimix

It is efficient for small input batches because each mix server needs $O(nlogn)$ exponentiations with low constant coefficient.
Each mix server simulates a pairwise permutation network. The mix server proves the correctness of each of its switching gate using the following verification protocol.

Verification Protocol for Switching Gate

Input is El Gamal ciphertexts (α_1, β_1), (α_2, β_2) of plaintexts m_1, m_2 respectively. Output is El Gamal ciphertexts (α'_1, β'_1), (α'_2, β'_2) of plaintexts m'_1, m'_2 respectively. The server proves statements:

- Statement 1: $m_1 m_2 = m'_1 m'_2$ using Plaintext Equivalent Proof (PEP) for $(\alpha_1\alpha_2, \beta_1\beta_2)$ and $(\alpha'_1\alpha'_2, \beta'_1\beta'_2)$.
- Statement 2: $m_1 = m'_1$ OR $m_1 = m'_2$ using DISjunctive Plaintext Equivalent Proof ($DISPEP$)

$\underline{PEP}$ proves (α', β') is a re-encryption of (α, β) by using Schnorr identification protocol

- Compute $(y_s, g_s) = ((\alpha/\alpha')^z(\beta/\beta'), y^z g)$ as Schnorr public key

- (α', β') re-encrypts $(\alpha, \beta) \Leftrightarrow \exists \gamma \in Z_q$: $(y_s, g_s) = ((y^z g)^\gamma, y^z g)$
- Prover uses Schnorr identification protocol to show that it knows γ

DISPEP proves (α_1, β_1) is a re-encryption of one of (α'_1, β'_1) and (α'_2, β'_2) by using Disjunctive Schnorr identification protocol. Proof in [2]:

- Compute $(y_{s1}, g_{s1}) = (\alpha_1/\alpha'_1, \beta_1/\beta'_1)$ and $(y_{s2}, g_{s2}) = (\alpha_1/\alpha'_2, \beta_1/\beta'_2)$ as Schnorr public keys
- Use Disjunctive Schnorr identification protocol to show knowledge of one of the Schnorr private keys, which is also the El Gamal private key x of the ciphertexts
- This requires the mix-server to know the El Gamal private key x, which is not acceptable

- We will show a revised version of this protocol which uses the approach in PEP and removes this problem

Modified $DISPEP$:

Compute

$$\begin{aligned}(y_{s1}, g_{s1}) &= ((\alpha_1/\alpha'_1)^{z_1}(\beta_1/\beta'_1), y^{z_1}g) \\ (y_{s2}, g_{s2}) &= ((\alpha_1/\alpha'_2)^{z_2}(\beta_1/\beta'_2), y^{z_2}g)\end{aligned}$$

as Schnorr public keys.

Assume w.l.o.g. that (α_1, β_1) is a re-encryption of (α'_1, β'_1), then $\exists\gamma_1 \in Z_q$ such that $(y_{s1}, g_{s1}) = ((y^{z_1}g)^{\gamma_1}, y^{z_1}g)$.

Mix-server uses Disjunctive Schnorr identification protocol with $(y_{s1}, g_{s1}), (y_{s2}, g_{s2})$ to show that it knows γ_1.

Attacking Furukawa-Sako01 Scheme

Break correctness with a success chance of at least 50%

Let a be a generator of Z_p, then $a^{kq} \neq 1$ and $a^{2kq} = 1$. The mix server modifies one of the output ciphertexts as

$$\begin{aligned} g'_{i_0} &= g^{r_{i_0}} g_{\phi^{-1}(i_0)} \\ m'_{i_0} &= y^{r_{i_0}} m_{\phi^{-1}(i_0)} a^{kq} \end{aligned}$$

Modifying m'_{i_0} only affects equation (16) in verification protocol

If c_{i_0} is even, $a^{c_{i_0}kq} = 1$. So

$$m'^{c_{i_0}}_{i_0} = (y^{r_{i_0}} m_{\phi^{-1}(i_0)} a^{kq})^{c_{i_0}} = (y^{r_{i_0}} m_{\phi^{-1}(i_0)})^{c_{i_0}}$$

Therefore, equation (16) remains correct and the verification protocol still accepts

In a similar way, the mix server can modify g'_{i_0}

Countermeasure

$m'_{i_0} \notin G_q$. So the attack can be detected by checking whether
$g'_i, m'_i \in G_q$, $i = 1, ..., n$
If $k = 1$, it requires one extra modular multiplication. If $k \neq 1$, two extra modular exponentiations are required

Security

The attack only affects Lemma 1 in [1]. We show the short-coming of the original proof and how the fix completes the proof.

Lemma 1 *Assume* $\mathcal{P}$ *knows* $\{A_{ij}\}, \{r_i\}, \{\alpha_i\}$ *and* α *satisfying (5) and (6), and* $\{s_i\}$ *and* s *satisfying (14). If (15) and (16) hold with non-negligible probability, then either the relationships*

$$\begin{cases} g' &= g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j} \\ g'_i &= g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}}, i = 1, ..., n \\ m' &= y^{\alpha} \prod_{j=1}^{n} m_j^{\alpha_j} \\ m'_i &= y^{r_i} \prod_{j=1}^{n} m_j^{A_{ji}}, i = 1, ..., n \end{cases}$$

hold or $\mathcal{P}$ *can generate nontrivial integers* $\{a_i\}$ *and* a *satisfying* $\tilde{g}^{a} \prod_{i=1}^{n} \tilde{g}_i{}^{a_i} = 1$ *with overwhelming probability.*

Proof Replace $\tilde{g}'$ and $\{\tilde{g}'_i\}$ in (14) by those in (5) and (6):

$$\tilde{g}^{\sum_{j=1}^{n} r_j c_j + \alpha - s} \prod_{i=1}^{n} \tilde{g_i}^{\sum_{j=1}^{n} A_{ij} c_j + \alpha_i - s_i} = 1$$

Therefore, either

$$\begin{cases} s = \sum_{j=1}^{n} r_j c_j + \alpha \\ s_i = \sum_{j=1}^{n} A_{ij} c_j + \alpha_i \end{cases}$$

hold or $\mathcal{P}$ can generate nontrivial integers $\{a_i\}$ and a satisfying $\tilde{g}^a \prod_{i=1}^{n} \tilde{g_i}^{a_i} = 1$

Replace s and $\{s_i\}$ in (15):

$$1 = b_0 \prod_{i=1}^{n} b_i^{c_i} \tag{20}$$

where

$$b_0 = \frac{g^{\alpha} \prod_{j=1}^{n} g_j^{\alpha_j}}{g'}$$
$$b_i = \frac{g^{r_i} \prod_{j=1}^{n} g_j^{A_{ji}}}{g'_i}, i = 1, ..., n$$

At this point, proof in [1] concludes $b_i = 1, i = 0, ..., n$. However, it is only correct if $b_i \in G_q$

Millimix Attack

An attack similar to one against Furukawa-Sako01 mix-net can be applied to Millimix.
A second attack exploits the fact that the exponents z in PEP and z_1, z_2 in $DISPEP$ can be arbitrarily chosen. Let (α_1, β_1) and (α_2, β_2) be input to a switching gate of a malicious mix-server. The server computes output as follows.

$$\begin{aligned}(\alpha'_1, \beta'_1) &= (\alpha_1 y^{-r_1 - s_1 z_1} g^{-s_1}, \beta_1 g^{-r_1}) \\ (\alpha'_2, \beta'_2) &= (\alpha_2 y^{-r_2 + s_1 z_1 - sz} g^{s_1 - s}, \beta_2 g^{-r_2})\end{aligned}$$

Using PEP and $DISPEP$ the server can still show that: (i) $(\alpha'_1 \alpha'_2, \beta'_1 \beta'_2)$ is the re-encryption of $(\alpha_1 \alpha_2, \beta_1 \beta_2)$, and (ii) either (α'_1, β'_1) or (α'_2, β'_2) re-encrypts (α_1, β_1). To show (i), the server computes

$$\begin{aligned}(\alpha/\alpha', \beta/\beta') &= (\alpha_1\alpha_2/\alpha_1'\alpha_2', \beta_1\beta_2/\beta_1'\beta_2') \\ &= (y^{r_1+r_2+sz}g^s, g^{r_1+r_2}) \\ (y_s, g_s) &= ((\alpha/\alpha')^z(\beta/\beta'), y^z g) = ((y^z g)^{r_1+r_2+sz}, y^z g) \\ &= (g_s^{r_1+r_2+sz}, g_s)\end{aligned}$$

Now Schnorr identification protocol will be performed as follows.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: a commitment $w = g_s^e$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge c
3. $\mathcal{P} \longrightarrow \mathcal{V}$: a response $s = e + c(r_1 + r_2 + sz)$

$\mathcal{V}$ then check if $g_s^s = wy_s^c$. This equation is correct and PEP has been broken.

To show (ii), we note that

$$\begin{aligned}(y_{s1}, g_{s1}) &= ((\alpha_1/\alpha'_1)^{z_1}(\beta_1/\beta'_1), y^{z_1}g) = ((y^{z_1}g)^{r_1+s_1z_1}, y^{z_1}g) \\ &= (g_{s1}^{r_1+s_1z_1}, g_{s1})\end{aligned}$$

Disjunctive Schnorr identification protocol can be performed as follows.

1. $\mathcal{P} \longrightarrow \mathcal{V}$: two commitments $w_1 = g_{s1}^{e_1}$, $w_2 = g_{s2}^{s_2}y_{s2}^{-c_2}$
2. $\mathcal{P} \longleftarrow \mathcal{V}$: a challenge c
3. $\mathcal{P} \longrightarrow \mathcal{V}$: responses $s_1 = e_1 + c_1(r_1 + s_1z_1)$, s_2, $c_1 = c \oplus c_2$, c_2

$\mathcal{V}$ then check that $g_{si}^{s_i} = w_i y_{si}^{c_i}$, $i = 1, 2$ holds

Countermeasure

z must be either chosen by the verifier after the switching gate has produced output. Or in non-interactive version, prover provides (z, c, s). A verifier then verifies

$$
\begin{aligned}
z &\stackrel{?}{=} H(\alpha' \parallel \beta' \parallel \alpha \parallel \beta) \bmod q \\
c &\stackrel{?}{=} H(g' \parallel y' \parallel g'^s y'^c) \bmod q
\end{aligned}
$$

where $(y', g') = ((\alpha/\alpha')^z(\beta/\beta'), y^z g)$ and $H : \{0,1\}^* \rightarrow 2^{|q|}$ is a hash function

$DISPEP$ can be modified similarly. Both z_1 and z_2 must be either chosen by the verifier after the switching gate has produced the output, or computed as

$z_1 = z_2 = H(\alpha_1' \parallel \beta_1' \parallel \alpha_2' \parallel \beta_2' \parallel \alpha_1 \parallel \beta_1 \parallel \alpha_2 \parallel \beta_2)$.

Security

We show revised Lemma 2 in [2] and its proof, Lemma 3 in [2] can be revised similarly.

Lemma 2 *Let* (α, β) *and* (α', β') *be two ciphertexts for which* PEP *produces* accept *response.*

- *if* z *is chosen by the prover, then* (α', β') *is not necessarily a valid re-encryption of* (α, β)*.*
- *if* z *is chosen by the verifier or computed by hash function as shown above, then either* (α', β') *is a valid re-encryption of* (α, β) *or the prover can find the El Gamal private key* x*.*

Proof Let z be chosen by verifier. Suppose K is the set of $z \in Z_q$ such that prover knows $o \in Z_q$ satisfying $(\alpha/\alpha')^z(\beta/\beta') = (y^z g)^o$. The probability that PEP outputs *accept* is $|K|/q$. With sufficiently large q, we can assume $|K| \geq 3$.

Assume distinct elements $z_0, z_1, z_2 \in K$. Let $\alpha/\alpha' = g^u$ and $\beta/\beta' = g^v$. Prover knows $o_0, o_1, o_2 \in Z_q$ satisfying $(\alpha/\alpha')^{z_i}(\beta/\beta') = (y^{z_i} g)^{o_i}$, $i = 0, 1, 2$ and so has the following system of three linear equations with three unknowns u, v and x:

$$\begin{cases} z_0 u + v - o_0 z_0 x & = & o_0 \\ z_1 u + v - o_1 z_1 x & = & o_1 \\ z_2 u + v - o_2 z_2 x & = & o_2 \end{cases}$$

As $\alpha, \beta, \alpha', \beta' \in G_q$, then u, v, x must exist, and so the system must have a solution. If the solution is unique, the prover will be able to solve it and find the value of x and that demonstrates a knowledge extractor for x.

On the other hand, if the system has more than one solution, the following determinants are equal zero.

$$det = \begin{vmatrix} z_0 & 1 & -o_0 z_0 \\ z_1 & 1 & -o_1 z_1 \\ z_2 & 1 & -o_2 z_2 \end{vmatrix} = 0$$

$$det_x = \begin{vmatrix} z_0 & 1 & -o_0 \\ z_1 & 1 & -o_1 \\ z_2 & 1 & -o_2 \end{vmatrix} = 0$$

This implies that,

$$\begin{aligned} 0 &= det + z_0 det_x \\ &= (o_2 - o_1)(z_0 - z_1)(z_0 - z_2) \end{aligned}$$

and so $o_2 = o_1$. This leads to $u = vx$, which means that

$\alpha/\alpha' = (\beta/\beta')^x$ and so (α', β') is a valid re-encryption of (α, β).

Lemma 3 *Let* (α_1, β_1), (α'_1, β'_1) *and* (α'_2, β'_2) *be ciphertexts for which* $DISPEP$ *produces* accept *response.*

- *if* z_1 *and* z_2 *are chosen by the prover, then* (α_1, β_1) *is not necessarily a valid re-encryption of either* (α'_1, β'_1) *or* (α'_2, β'_2)*.*
- *if* z_1 *and* z_2 *are chosen by the verifier or computed by hash function as shown above, then either* (α_1, β_1) *is a valid re-encryption of either* (α'_1, β'_1) *or* (α'_2, β'_2) *or the prover can find the El Gamal private key* x*.*

Conclusion

Two attacks against resilient mix-nets

Countermeasures and security and efficiency analysis

First attack against Furukawa-Sako01 mix-net can also be used against a number of other mix-nets. It could have wider implications proofs that are based on discrete logarithm assumption

Second attack breaks the verification protocol of Millimix. It can be countered by carefully choosing the challenge.

References

[1] J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle, pages 368 ff. J. Kilian (Ed.), CRYPTO 2001. LNCS 2139

[2] M. Jakobsson and A. Juels. Millimix: Mixing in small batches, 1999. DIMACS Technical Report 99-33.