A Practical Mix

Markus Jakobsson

Information Sciences Research Center, Bell Labs
Murray Hill, New Jersey 07974
www.bell-labs.com/user/markusj

Abstract. We introduce a robust and efficient mix-network for exponentiation, and use it to obtain a threshold decryption mix-network for ElGamal encrypted messages, in which mix servers do not need to trust each other for the correctness of the result. If a subset of mix servers cheat, they will be caught with an overwhelming probability, and the decryption can restart after replacing them, in a fashion that is transparent to the participants providing the input to be decrypted. As long as a quorum is not controlled by an adversary, the privacy of the mix is guaranteed. Our solution is proved to be secure if a commonly used assumption, the Decision Diffie-Hellman assumption, holds.
Of possible independent interest are two new methods that we introduce: *blinded destructive robustness*, a type of destructive robustness with protection against leaks of secret information; and *repetition robustness*, a method for obtaining robustness for some distributed vector computations. Here, two or more calculations of the same equation are performed, where the different computations are made independent by the use of blinding and permutation. The resulting vectors are then unblinded, sorted and compared to each other. This allows us to detect cheating (resulting in inequality of the vectors).
Also of possible independent interest is a modular extension to the ElGamal encryption scheme, making the resulting scheme non-malleable in the random oracle model. This is done by interpreting part of the ciphertext as a public key, and sign the ciphertext using the corresponding secret key.

Keywords: mix-network, decryption, privacy, robustness, error detection

1 Introduction

A mix-network takes a list of values as input, and outputs a permuted list of function evaluations (typically decryptions) of the input items, without revealing the relationship between input and output elements. Mix-networks were introduced by Chaum in 1981 [4] as a primitive for privacy. Although alternative primitives for privacy (e.g., [27]) can be used where users trust each other to some extent, for most applications, mix-networks still today remain the only realistic method to ensure connection privacy in settings like the Internet. Consequently, mix-networks have seen many applications since they were introduced, spanning the areas of pure communication (e.g., [31, 25]), the special case of web-browsing

[12], and election schemes [4, 11, 28]. They also have other potential uses wherever privacy is important, as in payment schemes. Mix-networks are also often used *implicitly*, by the assumption of an anonymous channel. However, in spite of its usefulness and wide use, this important primitive has almost not evolved at all since it was conceived.

The implementations that have been suggested have accepted a model requiring a lot of trust, or sacrificed either robustness, practicality or efficiency in order to provide a solution. For example, the majority of currently used remailers use only *one* mix server, which has to be fully trusted by the users for both privacy and correctness. If several such servers are pipelined for improved privacy (such as in [14, 23, 31]), this raises concerns in terms of both correctness and availability of service, especially in situations where mix servers are welcome targets of attackers. (It is interesting to note that if an attacker corrupting some subset of mix servers can verify a claimed mix decryption, the degree of privacy might paradoxically *decrease* with increased pipelining/distribution. This was first noticed in [17], in the context of an adversary desiring to verify that purchased votes were cast "correctly".) Furthermore, in order to allow an adversarial trust model of the type often assumed in commerce and election settings, and to obtain robustness, pipelined servers must be able to perform some form of correctness proof of their computation (without sacrificing privacy). There has been no suggestion of how to do this efficiently. So far, the most efficient proposals are those of Ogata, Kurosawa, Sako and Takatani [22], and Abe [1]. Apart from these, no robust *threshold* mix-network has been proposed. A threshold solution is important, or users would be forced to re-encrypt and re-send the encrypted messages in situations where an error or attack would force one or more servers to be replaced. This is not only a limitation in terms of efficiency and the adversarial model, but also excludes certain types of uses of the mix-network.

We demonstrate an elegant and efficient solution to this problem. More precisely, we suggest a mix-network for decryption of a list of ElGamal [10] encrypted messages, with the properties for privacy, availability, robustness, and efficiency listed in the next section.

Using novel methods for catching cheaters, we propose a solution to the same problem as studied by Ogata, Kurosawa, Sako and Takatani [22], and by Abe [1]. They propose methods in which each mix server re-encrypts each encrypted message twice, and permutes the corresponding two lists. Then, the other mix servers asks to have either the first block or the second block of permutations and encryptions revealed, which if done correctly, gives them 50% confidence that the server performing the transaction was honest. This is then repeated a large number of times for each server. Our method avoids cut-and-choose, and minimizes the amount of costly zero-knowledge proofs needed, and as a result, is significantly more efficient. In our particular solution, the cost of privacy and robustness increases the price of the computation by a factor $3\kappa + 7k$, where $\kappa = 1 - \frac{log\epsilon}{2logN}$. Here, ϵ is the maximum acceptable success probability of an attacker, N the number of elements to be decrypted, and k is the size of the quorum required. For a million elements, a maximum success probability for

an attacker of 2^{-80}, and a quorum size of 5, this gives us a cost factor for the robustness and privacy of approximately 44. The corresponding overheads of [1, 22] are approximately a factor of between 1500 and 4000 (in a setting with no external verifiers.) We note that all of the above costs can be somewhat reduced by applying methods for batch verification (e.g., [3]), but it appears that our algorithm would be more efficient for *all sizes* of inputs and quorums even after the algorithms are optimized.

Novel Methods:
In order to balance robustness and integrity of secret information, a variety of methods have been introduced and used. One of the most common methods is the zero-knowledge proof, but proofs with very limited degrees of knowledge leaks may also be used in the litterature. Since general zero-knowledge proofs normally are very costly, we want to limit the use of these, and shift towards other methods for robustness where possible, and otherwise to use only few and specialized zero-knowledge proofs. A proof that we will make use of in this work is that of proving valid undeniable signatures, introduced by Chaum and van Antwerpen [5]. This will be used to prove valid exponentiations without revealing the secret exponents.

Traditionally, a protocol has been made robust by having each participant prove to each other participant that he perfomed his part of the computation according to the protocol. The concept of *destructive robustness* was introduced in [19, 20] to improve the efficiency of robustness verifications by making the common case inexpensive to perform. The method of destructive robustness involves two steps: the *detection* of an error, and the *tracing* of cheaters (in case of an error). This separation of tasks improves efficiency and allows for simpler protocols (and simpler proofs of the same). This is so since the tracing, which is only performed when necessary, is allowed to destruct important protocol properties (hence the name), such as privacy, which allows for more straightforward protocols. The destruction of protocol properties may not be a problem: In the situation where the concept was introduced, the transcript would become meaningless as a result of the cheating, and could therefore be opened up to pinpoint cheaters, after which the protocol would be started anew. However, in the setting of this paper, this is not possible, as this would potentially reveal the messages that in an encrypted form constitute the input. In order to avoid this, we introduce the idea of *blinded destructive robustness*. Here, the input is blinded to begin with; if no cheater is detected, the result is unblinded; on the other hand, if the result is found to be incorrect, then the computation on the blinded input is verified. (This requires special attention to ensure that *the blinding* was not incorrectly performed.)

Detecting that cheating has taken place is not always easy. More specifically, in the situation we consider, where vectors are manipulated and permuted, and the permutations are secret due to privacy requirements, the conflict between robustness and privacy is particularly noticeable. We introduce the method of *repetition robustness*. This is a method for detecting cheating behavior in settings like these, to be used to obtain robustness. By blinding copies of the input vector

twice or more, and performing the desired operation on the blinded vectors, we can, after unblinding the resulting vectors, sort and compare the results (which should be equal to each other). If some of the participants of the computation cheated, the resulting vectors will be different with an overwhelming probability.

Another new method we introduce is that for making ElGamal encryptions non-malleable, by adding one field correspodning to a signature, whose public key is a portion of the non-modified ciphertext (and therefore does not depend on the identity of the encryptor, but only to the ciphertext.)

Outline: We begin by presenting the model and the requirements on the mix-scheme for ElGamal decryption in section 2. In section 3, we review ElGamal encryption and decryption, present our extension for non-malleability, and show how to perform a mix-decryption of ElGamal encrypted messages given a primitive for mix-exponentiation. We demonstrate such a primitive in section 4. The properties of the resulting schemes are stated in section 5, and proven in the appendix.

2 Model and Requirements

We assume three types of (polynomial-time limited) participants: *users*, a *bulletin board*, and the *mix servers*. The *users* post encrypted messages to the *bulletin board*. The bulletin board is simply an area to which users have appendive write access, and to which all participants have read access. After the bulletin board fills up, or after some other triggering event occurs, the *mix servers* compute a permuted list of decryptions of all valid encryptions posted on the bulletin board. All honest users should be granted *privacy*, i.e., that the relationship between their encrypted message and the corresponding decrypted message is secret. It may in some settings be required that dishonest users can be deprived of their privacy. (Here, a user may be labeled *dishonest* by posting the encryption of an invalid message according to some standard, or generally just behaving in a way that is not allowed in the system on the whole.) Finally, we have the *adversary*, who can control other participants, and who may be mobile (see, e.g., [15]). The adversary may wish to break the privacy of a user he does not control, or make participants he does not control accept as valid an invalid decryption of the posted messages.

More rigorously, the requirements on the mix-decryption scheme are as follows:

- **Correctness:** If all mix servers are honest, the correct output will be generated.
- **Privacy:** If there is not a quorum of corrupt and collaborating mix servers, it will not be possible for any set of participants to establish the relationship between input- and output items with a success probability non-negligibly better than what could be achieved by guessing a permutation of the unknown items uniformly at random.

- **Robustness:** If any subset of cheating participating mix servers produce an output different from that prescribed by the protocol, then the honest participating mix servers will be able to establish this fact. Furthermore, the honest servers will efficiently be able to establish the identities of the cheating servers.
- **Availability:** The system implements a high degree of availability, meaning that it allows a large number of mix servers not to be available, to refuse to participate, or to cheat actively, without affecting the availability of the service. The exact number that has to be available is governed by the choice of the threshold scheme.
- **Efficiency:** The scheme does not require any interaction between different users posting encrypted messages, nor between these users and the mix servers. The users need only a minimum amount of interaction (and only one round) with the bulletin board, and the amount of data transferred is low (with a maximum overhead per posted message of one signature). The amount of computation required by the users is low, allowing for smart card implementations. Furthermore, the amount of computation, communication, and storage required by the mix servers is low, both asymptotically and practically.

We provide a scheme that satisfies all of the above requirements, and prove it to be secure if the following assumption holds:

The Decision Diffie-Hellman Assumption: Let $p = 2q + 1$, for primes p and q, and let m, g be generators of a subgroup of order q. Then, the pairs (m, m^x, g, g^x) and (m, m^r, g, g^x) are indistinguishable, for random and unknown values $r, x \in Z_q$, $m, g \in G_p$.

3 ElGamal Decryption

3.1 Review

Public and Secret Information: Let p, q be primes such that $p = 2q + 1$, and g be a generator of G_p. The mix servers share a secret key x using a (k, n) threshold scheme (see [30, 24]); their corresponding public key is $y = g^x \bmod p$. Server j's secret share is x_j, its public share $y_j = g^{x_j} \bmod p$. (Onwards, we assume all arithmetic to be modulo p where applicable, unless otherwise stated.)

ElGamal: Our protocol uses ElGamal encryption [10]: To encrypt a value[1] m using the public key y, a value $\gamma \in_u Z_q$ is picked uniformly at random, and the pair $(a, b) = (my^\gamma, g^\gamma)$ calculated. Thus, (a, b) is the encryption of m. In order to decrypt this and obtain m, $m = a/b^x$ is calculated. It is known that the ElGamal encryption scheme is semantically secure if the Decision Diffie-Hallman assumption holds.

[1] Here, $m = (\frac{M}{p})M$ for an original message $M \in [1 \ldots \frac{p-1}{2}]$, where $(\frac{M}{p})$ is the Jacobi symbol of M.

Proof of valid exponentiation: One tool that will be repeatedly used to prove that a valid exponentiation was performed is a proof protocol for proving valid undeniable signatures (introduced by Chaum and van Antwerpen [5]). This is a proof that $log_m s = log_g y$ for a given quadruple (m, s, g, y). There are several protocol versions developed; in order to obtain a high degree of efficiency, we will use a non-interactive proof version, such as a Schnorr signature [29]. This requires one exponentiation per proof for the prover, and two per verifier.

3.2 Non-malleable ElGamal

Even if we have a primitive that takes a list of encrypted messages and decrypts these in a fashion that does not reveal any information about the relationship between items in the input list and items in the output list, we do not necessarily have a system that implements privacy. The reason is the following: an attacker may repeat (potentially disguised) copies of some items of the decrypted input list (potentially submitted by other parties than himself,) and then determine (with some probability) what the decryption of the attacked message was, by counting repeats or correlations in the output list. Therefore, it is necessary to use a non-malleable [9] scheme, such as our proposed extension to the ElGamal encryption scheme. Before the mix-decryption process starts, the mix servers remove any invalid encryption entry, and any duplicates of the same entry.

To implement non-malleability, the following approach will be taken: Each user proves knowledge of the value γ_i used for the encryption when submitting the pair (a_i, b_i). This can be done by treating b_i as the public key (for which γ_i is the corresponding secret key) and signing (a_i, b_i, aux) using this secret key, where aux is some auxiliary information – in our case, a number indicating how many previous batches have been decrypted, which is public information. We suggest using Schnorr [29] signatures for this. An encryption pair is said to be *valid* if the above signature is valid and correpsonds to the ciphertext pair. Encryptions that are not valid are discarded. The resulting encryption scheme is non-malleable in the random oracle model [26], as shown in the appendix.

Remark: Other methods of providing non-malleability can be employed. For example, we may use the recently proposed scheme by Cramer and Shoup [7], which is shown to be secure against an adaptive chosen message attack (which is shown to imply non-malleability in [2].) A list of such encryptions can be mix-decrypted using almost identical methods to those described in this paper.

3.3 Mix-decryption of ElGamal

Assume that we have a primitive $MIXEXP$ that takes a list of items $(c_1, \ldots, c_N)$ and robustly computes a permutation of the list, $({c_1}^{\delta}, \ldots, {c_N}^{\delta})$. Here, $\delta = \prod_{j \in Q} \delta_j$, where δ_j is server j's share, $\Delta_j = g^{\delta_j}$ is public, and Q denotes a quorum.

Consider now the list $((a_1, b_1), \ldots, (a_N, b_N))$ of ElGamal encrypted messages. We want to produce a permutation of the list $(m_1, \ldots, m_N)$, for $m_i = a_i / {b_i}^x$. This can be obtained in the following manner:

1. Each participating mix server, $j \in Q$, selects a secret value $\delta_j \in_u Z_q$, and publishes $\Delta_j = g^{\delta_j}$.
2. Each item, (a_i, b_i), in the input list is blinded using sequential exponentiation, resulting in a list $((\alpha_1, \beta_1), \ldots, (\alpha_N, \beta_N))$, s.t. $(\alpha_i, \beta_i) = ({a_i}^{1/\delta}, {b_i}^{1/\delta})$. $VALIDEXP(\alpha_{ij}, \alpha_{ij-1}, g, \Delta_j)$ is then run by each active server $j \in Q$ to prove that his output $\alpha_{ij} = {\alpha_{i(j-1)}}^{1/\delta_j}$. (Notice how the order of the input and output is switched, compared to the normal expression: thus, instead of proving correct exponentiation using δ_j, we prove the correct exponentiation using $1/\delta_j$.)
3. For each item β_i in the above list, the mix-servers distributively compute $\hat{\beta}_i = {\beta_i}^x$ (using a parallel threshold exponentiation, as demonstrated in among others, [24].) Server $j \in Q$ runs $VALIDEXP(\hat{\beta}_{ij}, \beta_i, g, y_j)$, for the share $\hat{\beta}_{ij}$ of $\hat{\beta}_i$ to prove correct computation.
4. For each new pair $(\alpha_i, \hat{\beta}_i)$, the corresponding "blinded decryption" $\mu_i = \alpha_i / \hat{\beta}_i$ is calculated. The result is a list $(\mu_1, \ldots, \mu_N)$.
5. Using the assumed primitive $MIXEXP$, the active mix-servers robustly compute a permutation of $(m_1, \ldots, m_N)$. Here, $m_i = {\mu_i}^{\delta}$; therefore, this list is the desired list of decrypted messages. Note that this is the only step where the relationship between items in the input and output is hidden.

4 A Mix-Exponentiation Scheme

Before we introduce how to design the protocol $MIXEXP$, we will look at a non-robust version of the same. This protocol will later be used as a building block. We then discuss the principles of the "global" solution, after which we present this solution itself.

4.1 A Building Block

Let the public input be the vector $\overline{\mu} = (\mu_1, \ldots, \mu_N)$, the secret keys $(\delta_1, \ldots, \delta_k)$, such that $\delta = \sum_{j=1}^{k} \delta_j$, and the desired output a permutation of $\overline{\sigma} = (\sigma_1, \ldots, \sigma_N)$, where $\sigma_i = {\mu_i}^{\delta}$. Let $\overline{\pi} = (\pi_1, \ldots, \pi_k)$ be a secret distributed vector of permutations, for which $\pi_j \in N \rightarrow N$. The servers in the quorum sequentially raise all the elements to their secret key, and apply their secret permutation to this output, producing the input to the next server, or, for the last server, the output of the protocol. We denote the application of the above protocol by $\pi(\overline{\mu}^{\delta})$, where π indicates the final permutation, δ is the accumulated exponent, and the exponentiation is element-wise w.r.t. the input vector. This building block has two uses: (1) it is used for robustness by disassociating its input from its output w.r.t. the order of the items. This is done in a manner that if performed several times produces several independent instances of the same input, preventing a cheater to modify a certain item (since he has to correctly guess the appearance of this item in each instance). (2) It is used to achieve privacy, again by disassociating the input order from the output order.

4.2 Principles of Our Robust Mix-Decryptor

The following ideas are used to ensure correctness of our solution:

- **Blinding I:** Multiple, independent blindings and permutations are performed on copies of the input vector. Later, the desired operations will be performed on these blinded vectors; after these are unblinded, the resulting vectors are sorted and compared. If they are different, this indicates that at least one participant cheated, and the result is declared invalid.
- **Blinding II:** At the same time as the desired operation is performed on the blinded vectors, a second blinding is performed. This affects all the vector elements in the same manner (and if it does not, this will constitute cheating, and will be detected as described in the previous step.) This blinding will be removed only for valid vectors, after blinding I has been verified.
- **Tracing:** If a result is declared invalid, then only blinding I is removed. This is the only blinding that is different for the multiple vectors, and the problem must therefore reside here. Each mix server then proves that it performed the correct computation on these unblinded vectors.

4.3 Our Solution

In order to obtain the desired degree of security, the blinded computation is repeated $\kappa \geq 2$ times. Here, the probability of an attack going unnoticed is bounded above by $(N(N-1))^{1-\kappa}$, where N is the number of items of the input list. (Thus, for an input vector with a million items, $\kappa = 3$ gives 2^{-80} as the maximum success probability of an attack.)

In the robust version, the public input is $\overline{\mu}$ as before, and the public keys of the participants $\{\Delta\}_{j \in Q}$, where $\Delta_j = g^{\delta_j}$. The corresponding secret keys are $\{\delta\}_{j \in Q}$, and the accumulated exponent is $\delta = \prod_{j \in Q} \delta_j$. The permutations used are generated internally, and are therefore not part of the input.

We are now ready to introduce the protocol for $MIXEXP$:

1. **Setup:** Each participating mix server j selects secret exponents, $\rho_{I\lambda j}$, for $1 \leq \lambda \leq \kappa$, and $\rho_{II\,j}$, each number chosen uniformly at random from the even numbers of Z_q; and secret permutations, $\pi_{I\lambda j}$, and $\pi_{II\,\lambda j}$, for $1 \leq \lambda \leq \kappa$, all chosen uniformly at random from $N \rightarrow N$.
2. **Blinding I:** The mix servers compute $\overline{\mu}_{I\lambda} = \pi_{I\lambda}(\overline{\mu}^{\rho_{I\lambda}})$, $1 \leq \lambda \leq \kappa$.
3. **Exponentiation and Blinding II:** The mix servers compute $\overline{\sigma}_{II\,\lambda} = \pi_{II\,\lambda}(\overline{\mu}_{I\lambda}{}^{\delta\,\rho_{II}})$, for $1 \leq \lambda \leq \kappa$.
4. **Unblinding I:**
   (a) The mix servers publish $\{\rho_{I\lambda j}\}_{j \in Q}$, for $1 \leq \lambda \leq \kappa$.
   (b) Each mix server computes $\overline{\sigma}_{I\lambda} = \overline{\sigma}_{II\,\lambda}{}^{1/\rho_{I\lambda}}$, for $1 \leq \lambda \leq \kappa$, where $\rho_{I\lambda} = \prod_{j \in Q} \rho_{I\lambda j}$.
   (c) The lists $\{\overline{\sigma}_{I\lambda}\}_{\lambda=1}^{\kappa}$ are sorted and compared. If they are all equal, and no element is zero, then the result is labeled valid, otherwise invalid.

5. **Unblinding II:** *(for valid results only)*
   (a) The mix servers publish $\{\pi_{I1j}\}_{j\in Q}$.
   (b) The computation of $\overline{\mu}_{I1}$ in "Blinding I" is verified.
   (c) The mix servers publish $\{\rho_{IIj}\}_{j\in Q}$.
   (d) Let M_j denote the product (modulo p) of all the elements constituting the input to mix server j, for $\lambda = 1$. Similarly, S_j denotes the product of all the output elements for mix server j, and $\lambda = 1$. Mix server $j \in Q$ runs $VALIDEXP(M_j, S_j, g, \Delta_j{}^{\rho_{IIj}})$.
   (e) The mix servers compute $\overline{\sigma}_1 = \overline{\sigma}_{I1}{}^{1/(\rho_{I1}\rho_{II})}$, and output $\overline{\sigma}_1$.
6. **Tracing:** *(for invalid results only)*
   (a) The mix servers publish $\{\pi_{I\lambda j}\}_{j\in Q}, \{\pi_{II\lambda j}\}_{j\in Q}, \{(\delta_j \rho_{IIj})\}_{j\in Q}$.
   (b) The computation of $\overline{\mu}_{I\lambda}$, $1 \leq \lambda \leq \kappa$, in "Blinding I" is verified.
   (c) The computation of $\overline{\sigma}_{II\lambda}$, $1 \leq \lambda \leq \kappa$, in "Exponentiation and Blinding II" is verified.

In the above, if any mix server is found to have cheated, then the tracing protocol halts after the current step has been completed, the cheater(s) replaced, and the protocol restarts.

5 Claims

The proposed scheme for Mix-ElGamal decryption, using the introduced primitive for $MIXEXP$, has the following properties, which are proven in the Appendix: correctness (theorem 1,) robustness (theorem 2,) and privacy (theorem 3.) As a result, the scheme for mix-decryption of ElGamal can easily be seen to have the same properties. Following the method in [16], the system can be shown to be proactivizable. Our proposed non-malleable ElGamal version is proved to indeed be non-malleable in theorem 4.

6 Efficiency Analysis

In this section, we specify the cost of the privacy and robustness in terms of the cost factor with which the computation increases. This depends on the size k of the quorum, and the maximum probability ϵ of success for an adversary. We have that κ, the number of repeats required, equals $\kappa = 1 - \frac{log\epsilon}{2logN}$, where N is the number of items of the input.

The non-robust building block for exponentiation and permutation requires kN modular exponentiations. Generating a proof in $VALIDEXP$ requires one exponentiation, verifying its correctness requires 2 exponentiations per verifier. $MIXEXP$ therefore requires $3\kappa kN + k^2N + k(2k-1)$ exponentiations given honest participants (κkN each from step 2, 3 and 4; $(k^2N + k(1 + 2(k-1)))$ from step 5.) Therefore, the cost of the entire protocol for mix-decryption is

approximately $3\kappa kN + 7k^2N$ exponentiations given honest[2] participants ($2kN + 2kN(2k-1)$ for step 2; $kN + kN(2k-1)$ for step 3; and $3\kappa kN + k^2N + k(2k-1)$ for step 4.)

Given the same quorum size but no privacy protection or robustness, the corresponding number of exponentiations would have been kN, giving us a cost increase factor of $3\kappa + 7k$. For $\kappa = 3$, $k = 5$, which is reasonable in many settings, this means an overhead of a factor of 44.

We note that the verification of valid exponentiations can be done using batch verification, bringing down the cost substantially. We can use the methods for modular exponentiation with common exponent suggested in [3] for this. Also, we note that if joint proofs (see [1]) are performed for $VALIDEXP$, then it appears that the overhead can be made linear instead of quadratic.

7 Acknowledgements

Many thanks to Daniel Bleichenbacher, Russell Impagliazzo, Ari Juels, Markus Michels and Markus Stadler for valuable discussions and suggestions. Thanks to David Pointcheval for contributing the proof that the suggested extension of the ElGamal encryption scheme is non-malleable in the random oracle model. Last but not least, I wish to thank Masayuki Abe for pointing out unclarities in a previous version.

References

1. M. Abe, "Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-centers," Eurocrypt '98.
2. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, "Plaintext Awareness, Non-Malleability, and Chosen Ciphertext Security: Implications and Separations," manuscript.
3. M. Bellare, J. Garay, T. Rabin, "Batch Verification with Applications to Program Checking and Cryptography," Eurocrypt '98
4. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, ACM 1981, pp. 84-88 "Undeniable Signatures,"
5. D. Chaum, H. Van Antwerpen, Crypto '89, pp. 212-216
6. D. Chaum, "Zero-Knowledge Undeniable Signatures," Eurocrypt '90, pp. 458-464
7. R. Cramer, V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," available at `www.cs.wisc.edu/ shoup`.
8. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to Share a Function Securely," STOC '94, pp. 522-533
9. D. Dolev, C. Dwork, M. Naor, "Non-malleable cryptography," STOC'91, pp. 542 – 552

[2] Recall that we do not assume that the participants are honest, but only that the common case is that they are. When they are not, an extra cost is imposed for finding the identities of the cheaters.

10. T. ElGamal "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," Crypto '84, pp. 10-18
11. A. Fujioka, T. Okamoto, K. Ohta, "A practical secret voting scheme for large scale elections," Auscrypt '92, pp. 244-251
12. E. Gabber, P. Gibbons, Y. Matias, A. Mayer, "How to make personalized web browsing simple, secure, and anonymous," Financial Cryptography '97
13. S. Goldwasser and S. Micali, "Probabilistic Encryption," J. Comp. Sys. Sci. 28, pp 270-299, 1984.
14. C. Gulcu, G. Tsudik, "Mixing email with babel," ISOC Symposium on Network and Distributed System Security, 1996.
15. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Secret Sharing, or How to Cope with Perpetual Leakage," Crypto '95, pp. 339-352
16. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Public Key and Signature Systems," Proceedings of the 4th ACM Conference on Computer and Communications Security, 1997, pp. 100-110
17. M. Michels, P. Horster, "Some remarks on a receipt free and universally verifiable Mix-Type Voting scheme," Asiacrypt'96, pp. 125-132
18. M. Jakobsson, K. Sako, R. Impagliazzo, "Designated Verifier Proofs and Their Applications," Eurocrypt '96, pp. 143-154
19. M. Jakobsson, M. Yung, "Distributed 'Magic Ink' Signatures," Eurocrypt '97, pp. 450-464
20. M. Jakobsson, "Privacy vs. Anonymity," Ph.D. Thesis, University of California, San Diego, 1997. Available at `www.bell-labs.com/markusj/`.
21. NIST FIPS PUB XX, "Digital Signature Standard," National Institute of Standards and Technology, U.S. Department of Commerce, Draft, 1 Feb. 1993
22. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, "Fault Tolerant Anonymous Channel," ICICS '97, pp. 440-444
23. C. Park, K. Itoh, K. Kurosawa, "All/nothing election scheme and anonymous channel," Eurocrypt '93, pp. 248-259
24. T. P. Pedersen. "A threshold cryptosystem without a trusted party," Eurocrypt '91, pp. 522 - 526.
25. A. Pfitzmann, B. Pfitzmann, M. Waidner, "ISDN-MIXes: Untraceable Communication with Very Small Bandwidth Overhead," Information Security, Proc. IFIP/Sec'91, Mai 1991, Brighton, D. T. Lindsay, W. L. Price (eds.), North-Holland, Amsterdam 1991, 245-258.
26. D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes," Advances in Cryptology - Proceedings of Eurocrypt '96, pp. 387–398.
27. M. Reiter, A. Rubin, "Crowds: Anonymous Web Transactions," Manuscript at `www.research.att.com/projects/crowds/`
28. K. Sako, J. Kilian, "Receipt-Free Mix-Type Voting Scheme," Eurocrypt '95, pp. 393 - 403
29. C. P. Schnorr, "Efficient Signature Generation for Smart Cards," Advances in Cryptology - Proceedings of Crypto '89, pp. 239-252
30. A. Shamir, "How to Share a Secret," Communications of the ACM, Vol. 22, 1979, pp. 612-613
31. P. Syverson, D. Goldschlag, M. Reed, "Anonymous connections and onion routing," IEEE Symposium on Security and Privacy, 1997.

Appendix: Proofs of Claims

Instead of basing the proofs on the Decision Diffie-Hellman assumption, we will use a (more relevant) assumption, implied by the DDH assumption:

The Ordering Assumption: Consider the sets $M = \{m_j\}_{i=1}^{N}$ and $S = \{s_j\}_{i=1}^{N}$, where $s_i = m_i{}^x$ for secret, independent and uniformly distributed m_i, x. We say that (M, S) can be *ordered* if there exists a polynomial-time algorithm that matches at least one of the items s_i to m_i with a probability non-negligibly larger than that of choosing the pair uniformly at random. The ordering assumption states that no (M, S) can be ordered.

Lemma 1: The ordering assumption is implied by the Decision Diffie-Hellman assumption.

Proof of Lemma 1:
Let us assume that the Decision Diffie-Hellman assumption holds. Consider the sets $M = \{m_i\}_{i=1}^{N}$ and $S_0 = \{s_{0i}\}_{i=1}^{N}$, where $s_{0i} = m_i{}^{x_i}$ for secret, independent and uniformly distributed m_i, x_i. Clearly, (M, S_0) cannot be ordered, since for every possible order, there is a set $X = \{x_i\}_{i=1}^{N}$ of secret keys that makes the matching valid. Consider now $S_j = \{s_{ji}\}_{i=1}^{N}$, where $s_{ji} = m_i{}^{x_i}$, where j out of the N secret keys in $X = \{x_i\}_{i=1}^{N}$ are identical (with a value chosen uniformly at random), and the rest are independent and uniformly distributed. Assume that it is not possible to order (M, S_j), but that it is possible to order (M, S_{j+1}). This can be used as a black box to produce a polynomial-time algorithm that with a non-negligible probability can distinguish between the quadruples (g, g^x, m, m^x) and (g, g^x, m, m^r), thus contradicting the Decision Diffie-Hellman assumption. We produce the algorithm by substituting a value s_i of S_j, whose exponent x_i is not a duplicate, with a value s_i' which we want to know if it equals m_i^x, for the duplicated exponent x. If the resulting pair (M, S') can be ordered, then $s_i' = m_i^x$, otherwise not. In order not to get a contradiction, we must conclude that (M, S_N) cannot be ordered if the Decision Diffie-Hellman assumption holds, which concludes the proof. □

Theorem 1: *The scheme is correct:* If all participating mix servers are honest, then their output will be a permuted list of messages corresponding to the list of encrypted messages.

Proof of Theorem 1: *(Sketch)*
The non-robust version of $MIXEXP$ produces as output a permutation of $\overline{\mu}^{\delta}$, where the exponentiation is per item, $\overline{\mu}$ is the public input, and δ is the product of all the distributed components $\{\delta_j\}_{j \in Q}$. Therefore, step 5e of MIXEXP cancels the effects of the blindings in steps 2 and 3 (apart from the permutations), resulting in an output which is a permutation of the list $\overline{\mu}^{\delta}$. Considering the mix-decryption now, we see that the exponentiations of steps 2 and 5 cancel each other; therefore, considering the operation in step 3, if the public input is a list of pairs (a_i, b_i), then the output is a permutation of a list with items $m_i = a_i / b_i{}^x$. Thus, the scheme is correct. □

Theorem 2: *The scheme is robust:* Assume that there are $k-1$ dishonest participating mix servers controlled by an adversary. It is the goal of the adversary to force the output of the scheme to be a vector that corresponds to an incorrect decryption of the input vector. The adversary has only a negligible probability of success, and the identity of the adversary will be learnt by the honest participants with an overwhelming probability.

Proof of Theorem 2: *(Sketch)*
We see that step 2 of $MIXEXP$ must be performed correctly by an adversary, or the cheating will be detected in steps 4a (Unblinding I) and 5a-b (unblinding II). Due to the use of $VALIDEXP$, the adversary must keep the product of all the items he is outputting correct, or the cheating will be detected in step 5d. As a result of the operation in step 2, and the ordering assumption, we have that if the adversary does not control a quorum (implying that at least one participating mix server is honest), then the adversary cannot correlate the items constituting the inputs and outputs in step 2 to each other with a probability non-negligibly better than a uniform at random guess. Therefore, the adversary cannot identify the same two items of the output lists of the different repetitions with a probability significantly exceeding $\frac{1}{N(N-1)}^{\kappa-1}$. Therefore, the adversary cannot alter two items in each repetition (and this is the minimum number to retain the same product of all the items), with a probability significantly exceeding $\frac{1}{N(N-1)}^{\kappa-1}$. Therefore, any cheating will be detected, and the tracing option invoked. Since all computation has to be revealed in step 6 of $MIXEXP$, a cheating adversary will be identified. □

Theorem 3: *The scheme implements privacy:* Assume that there are $k-1$ dishonest mix servers, i.e., there is at least one participating mix server that is honest. Let the dishonest mix servers be controlled and coordinated by an adversary. This adversary also controls up to $N-2$ out of the N users posting encrypted messages. It is the goal of the adversary to match each one of the two decrypted messages that he does not control to their corresponding encrypted messages. The adversary has only a negligible probability of success.

Proof of Theorem 3: *(Sketch)*
If some mix servers break the privacy of some individual, then they match this individual's input to the mix to the corresponding output with a probability significantly better than making a guess uniformly at random given all the known input-output relations. We note that the input items are independent, since a proof of knowledge is required for an input item to be accepted. Also, note that items to the mix cannot be distinguished from random elements of the group, due to the blinding in the pre-processing stage (see section 3.3.) Let us now consider two cases, one in which the final result is declared valid, and one in which it is declared invalid:
Valid result: We know from theorem 2 that as long as one of the participating servers is honest, then the computation is robust, and a result that is declared valid must be valid with an overwhelming probability. Therefore, in order to break the privacy, the cheating mix servers need to match input elements to

output elements of the honest servers computation in step 3 (Exponentiation and Blinding II) of $MIXEXP$. Since the permutation employed in this step is not revealed by the honest mix server, nor is his share of the exponent δ, if the cheating mix servers can match the input to the output, then this would contradict the ordering assumption, and therefore also the Decision Diffie-Hellman assumption.
Invalid result: If a result is declared invalid, then the honest server will perform step 6 (Tracing), in which the permutations used, and the shares of the product $\delta\rho_{II}$ are published. However, the honest mix server will not reveal its share of δ (or ρ_{II}). Therefore, even though the permutation is known, no information about δ is given to the cheaters, and the exponent the input is raised to is chosen uniformly at random. Therefore, this is a result that the cheaters can simulate, and therefore, does not give them any information that allows them to break privacy. □

Theorem 4: The proposed extension of the ElGamal encryption scheme is non-malleable in the random oracle model.

Proof of Theorem 4: *(Sketch)*
The scheme can be proven semantically secure, as the original ElGamal scheme, relative to the the DDH assumption. We assume an adversary can provide a ciphertext c' given a ciphertext c, relative to a message m_b, for an unknown $b \in \{0, 1\}$ and two known plaintext messages m_0 and m_1. Decrypting c' using an oracle replay technique (see [26]) we get m', the plaintext message corresponding to c'. Knowing the relation between c' and c, we can determine whether m' corresponds to m_0 or m_1, thereby breaking the semantic security, which gives a contradiction. □