Elements in $\mathbb{Z}_p^* \backslash G_q$ are Dangerous

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

February 27, 2003

Abstract

A subgroup G_q, of prime order q, for which the discrete logarithm problem is assumed hard is mostly a subgroup of a larger group, e.g. $\mathbb{Z}_p^*$ for some prime $p = \kappa q + 1$. Arithmetic for G_q is not implemented directly. Instead, arithmetic for the larger group, e.g. $\mathbb{Z}_p^*$, is implemented, which gives arithmetic also for G_q.

Furthermore, in most protocols based on arithmetic in G_q the participants do not verify their input properly. They only verify that inputs are in $\mathbb{Z}_p^*$, even when they should verify that the inputs are in G_q.

This practice sometimes allows the adversary to hand inputs from $\mathbb{Z}_p^* \backslash G_q$ to honest participants without detection. Surprisingly, this seems to be a novel observation.

In this paper we discuss the general problem further, introduce a novel class of attacks based on the above observations, and outline an generic recipe for how to counter the attacks efficiently.

To illustrate both the attacks and the countermeasures we provide examples of protocols that are vulnerable to the novel attacks, and show how they may be modified to counter the attacks.

In particular we present attacks on the robustness for the majority of the El Gamal based mix-nets in the literature, and an attack on the privacy for the generic mix-net based on the proof of knowledge of correct shuffle of Furukawa and Sako [9].

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

SICS Technical Report: T2003:05
ISSN: 1100-3154
ISRN: SICS-T-2003/05-SE

Elements in $\mathbb{Z}_p^* \backslash G_q$ are Dangerous

Douglas Wikström

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

February 27, 2003

Abstract. A subgroup G_q, of prime order q, for which the discrete logarithm problem is assumed hard is mostly a subgroup of a larger group, e.g. $\mathbb{Z}_p^*$ for some prime $p = \kappa q + 1$. Arithmetic for G_q is not implemented directly. Instead, arithmetic for the larger group, e.g. $\mathbb{Z}_p^*$, is implemented, which gives arithmetic also for G_q.
Furthermore, in most protocols based on arithmetic in G_q the participants do not verify their input properly. They only verify that inputs are in $\mathbb{Z}_p^*$, even when they should verify that the inputs are in G_q.
This practice sometimes allows the adversary to hand inputs from $\mathbb{Z}_p^* \backslash G_q$ to honest participants without detection. Surprisingly, this seems to be a novel observation.
In this paper we discuss the general problem further, introduce a novel class of attacks based on the above observations, and outline an generic recipe for how to counter the attacks efficiently.
To illustrate both the attacks and the countermeasures we provide examples of protocols that are vulnerable to the novel attacks, and show how they may be modified to counter the attacks.
In particular we present attacks on the robustness for the majority of the El Gamal based mix-nets in the literature, and an attack on the privacy for the generic mix-net based on the proof of knowledge of correct shuffle of Furukawa and Sako [9].

1 Introduction

The two most commonly used complexity assumptions in cryptography are the difficulty of factoring and the difficulty of solving discrete logarithms. Relative to these assumptions a large number of cryptographic problems have been solved. In this paper we consider protocols based on the latter assumption. The discrete logarithm problem is believed to be hard in large subgroups of prime order of multiplicative groups modulo a prime, or elliptic curve groups. Our results are applicable to both types of groups, but we restrict our exposition to the former type of group for concreteness.

Consider $\mathbb{Z}_p^*$ for some primes $p = \kappa q + 1$ and q, and let G_q be the subgroup of $\mathbb{Z}_p^*$ of order q in which the discrete logarithm problem is assumed to be hard. Arithmetic for G_q is normally not implemented directly. Instead, arithmetic for the group $\mathbb{Z}_p^*$, and the field $\mathbb{Z}_q$ is implemented, which gives arithmetic also for G_q. Furthermore, in most protocols based on arithmetic in G_q, the participants do not verify that inputs expected to be in G_q indeed do so, but only verify that their inputs belong to $\mathbb{Z}_p^*$.

The thesis of this paper is that this practice is dangerous, since it sometimes allows the adversary to hand corrupt inputs to honest participants without detection.

We discuss the general problem and outline a generic method for avoiding explicit verifications. We then illustrate the problem by giving novel attacks for some well known protocols in the literature. We also show how to apply our generic method and solve the problems efficiently in the protocols under attack.

In particular, we first introduce a novel adversarial strategy for proofs of knowledge of discrete logarithms. We use this to break the robustness of most El Gamal based mix-nets in the literature, e.g. [23, 13, 18, 22, 16, 30, 25, 1, 10]. Then we adapt the adversarial strategy to the proof of correct shuffle presented by Furukawa and Sako [9], and use this to give an attack on the privacy of the generic mix-net based on their proof.

The attack for the generic mix-net based on the Furukawa-Sako proof of knowledge is an extension of the attacks given by Wikström [35] for the mix-nets by Golle et al. [13], Jakobsson [18], and Mitomo and Kurusawa [22].

2 Notation

In this paper we let $p = \kappa q + 1$, and q be prime numbers, such that κ is small. Formally we require that κ is bounded by a constant. We write $p_1, \ldots, p_s$ for the prime factors of κ. Then we let G_q and G_κ be the unique subgroups of $\mathbb{Z}_p^*$, of order q and κ respectively, i.e. $\mathbb{Z}_p^* = G_q \times G_\kappa$. We also identify the elements of the field $\mathbb{Z}_q$ with the elements of the set $[q] = \{0, \ldots, q-1\}$, i.e. we use a fixed representative $a \in [q]$ for each coset $a + (q)$ of the ideal (q) in $\mathbb{Z}$. This is mostly the case in practice and important since it gives a well defined interpretation of a^x for $a \in \mathbb{Z}_p^* \backslash G_q$ and $x \in \mathbb{Z}_q$. Throughout the paper H denotes a hash function modeled as a random oracle, but the range of H differs depending on the section. We denote by Σ_N the set of permutations on N elements.

The mix-net we consider in this paper is based on the El Gamal [7] cryptosystem employed in G_q. The El Gamal cryptosystem is defined as follows. A private key x is generated by choosing $x \in \mathbb{Z}_q$ uniformly and independently at random. The corresponding public key is $y = g^x$, where g is some system wide generator of G_q. Encryption of a message $m \in G_q$ using the public key y is given by $E_y(m, r) = (g^r, y^r m)$, 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 $(g^{r'} u, y^{r'} v) = (g^{r+r'}, y^{r+r'} m) = E_y(m, r + r')$.

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

3 The Idea

The idea of this paper is based on two observations. Although the observations are fairly obvious, it seems that the important consequences of these observations have, somewhat surprisingly, been overlooked in the literature.

The first observation is that when G_q is used for cryptographic purposes the arithmetic of this group is not implemented directly. Instead a library for arithmetic in the group $\mathbb{Z}_p^*$ is implemented, and mostly also a library for arithmetic in the field $\mathbb{Z}_q$. Together two such libraries provide what is needed for efficient arithmetic in G_q. In fact, it seems very hard to implement arithmetic in G_q efficiently in some essentially other way. A side effect of this design is that implementations normally are able to perform arithmetic for *all* of $\mathbb{Z}_p^*$.

The second observation is that in most protocols in the literature the participants do not verify their input properly. Even if elements in the input are expected to belong to G_q, it is only verified that they belong to $\mathbb{Z}_p^*$. The reason is probably that it is easy to verify membership in $\mathbb{Z}_p^*$, but costly to verify membership in G_q.

Our observations indicate that the adversary may be able to hand the participants corrupt inputs from $\mathbb{Z}_p^* \backslash G_q$ without detection. Depending on the particular protocol under attack, the implications of this seemingly innocent additional power of the adversary are more or less severe. The adversary may use the additional power in several ways, e.g. it may introduce elements from $\mathbb{Z}_p^*$ to corrupt the output of an execution, but it may also try to extract information by using corrupt input in the interaction with honest participants. The particular protocol under attack decides what is appropriate.

It is easy[2] to verify that a string of bits s represent an element $a \in \mathbb{Z}_p^*$. In the most common representation this amounts to checking that if s is interpreted as an integer a in base two we have $0 < a < p$. Similarly it is easy to verify membership in $\mathbb{Z}_q$. Throughout this paper we assume that all participants explicitly verify that their inputs belong to $\mathbb{Z}_p^*$ or $\mathbb{Z}_q$ as appropriate. To verify that $a \in G_q$ given that $a \in \mathbb{Z}_p^*$ is more costly. This requires verifying that $a^q = 1$, i.e. a full exponentiation in $\mathbb{Z}_p^*$.

The naive method for removing the additional power of the adversary is that each subprotocol independently verifies that its inputs belong to G_q, but since this is costly it is natural to look for other solutions.

The following simple observations are useful. Note that if $a \in \mathbb{Z}_p^*$, we may write $a = a'\zeta$ for some $a' \in G_q$ and $\zeta \in G_\kappa$. This implies that $a^\kappa = (a')^\kappa \zeta^\kappa$, and since the order of ζ is divided by κ by definition, we have $\zeta^\kappa = 1$, and $a^\kappa = (a')^\kappa \in G_q$. It is also very easy to compute a^κ, since this corresponds to at most $\frac{3}{2}\log_2 \kappa$ multiplications in G_q. We call such exponentiations κ-exponentiations. Similarly, multiplication by κ in $\mathbb{Z}_q$ is not a full multiplication, and we call this a κ-multiplication. Suppose now that $a = a'\zeta \in \mathbb{Z}_p^* \backslash G_q$, and that $\gcd(\theta, \kappa) = 1$ (when θ and κ are viewed as integers). Then we have $a^\theta = (a')^\theta \zeta^\theta$. Clearly

[2] The corresponding verification for an elliptic curve group is slightly more complicated, but still easy.

$(a')^\theta \in G_q$, but $\zeta^\theta \notin G_q$, since the order of ζ divides κ and $\gcd(\theta, \kappa) = 1$. Thus if $\gcd(\theta, \kappa) = 1$, then $a \in \mathbb{Z}_p^* \backslash G_q$ implies that $a^\theta \in \mathbb{Z}_p^* \backslash G_q$.

It seems impossible to give single detailed description of how to fix the problem efficiently in all protocols. However, it is possible to give a generic recipe for how to solve the problem.

> RECIPE: Consider the elements of a protocol that are required to belong to G_q. Find a small subset of these such that if membership in G_q is explicitly verified for these, the protocol may be modified to ensure that the rest of the elements belong to G_q to.

Depending on the protocol, slightly different methods may be required, but the basic idea is the same. In the following we illustrate the above ideas by describing how the observations can be applied to find adversarial strategies, and how the recipe can be used to find efficient counter measures.

4 Application to Proofs of Knowledge of Logarithms

In this section we apply the general idea of Section 3 to proofs of knowledge of discrete logarithms. For concreteness we consider a specific proof of knowledge, but the ideas is easily adapted to the wide variety of similar proofs of knowledge in the literature. In this section we let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$.

4.1 A Proof of Knowledge of γ Such That $\log_a A = \log_b B = \gamma$

Let $a, A, b, B \in G_q$, where $A = a^\gamma$, and $B = b^\gamma$ for some $\gamma \in \mathbb{Z}_q$ known by the prover. We review the standard non-interactive zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$.

1. The prover chooses $\delta \in \mathbb{Z}_q$ randomly and computes $\alpha = a^\delta$, $\beta = b^\delta$, $\theta = H(\alpha, \beta, A, B, a, b)$, and $d = \gamma\theta + \delta$. The proof consists of the tuple (α, β, d).
2. The verifier computes $\theta = H(\alpha, \beta, A, B, a, b)$ and verifies that $A^\theta \alpha = a^d$, and $B^\theta \beta = b^d$.

An equivalent variant of this that gives a shorter proof is to let (θ, d) be the proof and let the verifier check that $\theta = H(a^d/A^\theta, b^d/B^\theta, A, B, a, b)$, but for increased readability we use the variant described above. A proof of the following proposition can be found in the literature, e.g. [4, 31].

Proposition 1. *Let $a, b, A, B \in G_q$. Then the above is a zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$.*

4.2 Adversarial Strategy

Remember that $\mathbb{Z}_p^* = G_q \times G_\kappa$, and let $1 \neq \zeta \in G_\kappa$. We consider three scenarios:

1. A is replaced by ζA,

2. a is replaced by ζa, and
3. a is replaced by ζa, A is replaced by $\zeta^\gamma A$, and γ is chosen randomly in $\mathbb{Z}_q$.

The prover is malicious and claims that it knows a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$. The adversarial strategy is very simple. Repeatedly, i.e. at most a polynomial h number of times, simulate the honest prover on the corrupted joint input a, b, A, B, until it outputs a proof that is deemed valid by the verifier. It is clear that if the prover succeeds, the verifier is convinced.

It may seem to some readers that the adversary always succeeds in Scenario 3, since we do have $\log_a A = \log_b B = \gamma$, i.e. if we identify $\mathbb{Z}_{p-1}$ with $[p-1]$ we may view γ as an element in both $\mathbb{Z}_q$ and $\mathbb{Z}_{p-1}$. Unfortunately, if we look closer at the protocol this is not true. The problem is that the prover computes $d = \gamma\theta + \delta \bmod q$, and the verifier explicitly verifies that $d \in \mathbb{Z}_q$.

What the adversary hopes for in the three scenarios is that $\zeta^\theta = 1$, $\zeta^\delta = \zeta^d$, and $\zeta^\theta \zeta^{\gamma\delta} = \zeta^d$ respectively. If this happens it is easy to see that the proof produced by the simulated prover is deemed valid. We prove the following in Appendix A.

Lemma 1. *Each iteration succeeds with probability at least $\frac{1}{\xi} + O(\log^2 q/q)$, and the adversarial strategy fails with negligible probability.*

We stress that we do not claim that Proposition 1 is false. So how is it possible that the prover can fool the verifier with non-negligible probability? The answer is very simple. Proposition 1 requires that $a, b, A, B \in G_q$, and when this is not the case the proposition is no longer guaranteed to hold. Unfortunately, in most applications of the protocol of Section 4.1 it is *not* ensured that $a, b, A, B \in G_q$, which makes many applications of the protocol vulnerable our adversarial strategy.

We note that the probability of success in a single iteration in Scenario 3 equals the probability that an honest prover, unaware that $a \notin G_q$, happens to output a proof deemed valid.

To verify that $a, b, A, B, \in G_q$ before each invocation of the protocol of Section 4.1 would be very costly, and many applications do ensure, or can be modified to ensure, that $a, b \in G_q$ prior to the invocation of the protocol. Thus an efficient proof of knowledge under the weaker assumption that $a, b \in G_q$ and $A, B \in \mathbb{Z}_p^*$ is useful. In the next section we describe such a protocol.

4.3 How to Avoid Explicitly Verifying That $A, B \in G_q$

In this section we assume that the verifier is somehow convinced that $a, b \in G_q$, i.e. the calling context ensures this. This assumption is natural for two reasons: many protocols do ensure this by construction, and many protocols that do not can be modified such that they do.

We apply the generic method of Section 3 to modify the proof of knowledge of Section 4.1, such that it stays a proof of knowledge under the weaker assumption that $a, b \in G_q$ and $A, B \in \mathbb{Z}_p^*$.

Since we identified $[q] = \{0, \ldots, q-1\}$ and $\mathbb{Z}_q$, we may interpret $\theta \in \mathbb{Z}_q$ as an integer, denote by θ^* the unique integer such that $\theta = \theta^* \prod_{i=1}^{s} p_i$, where $p_i \nmid \theta^*$ for $i = 1, \ldots, s$, and then view θ^* as an element in $\mathbb{Z}_q$. The modified proof of knowledge is defined as follows:

1. The prover chooses $\delta \in \mathbb{Z}_q$ randomly and computes $\alpha = a^\delta$, $\beta = b^\delta$, $\theta = H(\alpha, \beta, A, B, a, b)$, and $d = \gamma\theta^* + \kappa\delta$. The proof consists of the tuple (α, β, d).
2. The verifier first computes $\theta = H(\alpha, \beta, A, B, a, b)$ Then it verifies that that $A^{\theta^*}\alpha^\kappa = a^d$, and $B^{\theta^*}\beta^\kappa = b^d$.

Recall that κ is bounded by a constant. This implies that the prover and the verifier may compute θ^* from θ by repeated trial division with the prime factors $p_1, \ldots, p_s$ of κ. It also implies that the exponentiations α^κ and β^κ can be performed quickly. We prove the following proposition.

Proposition 2. *Let $a, b \in G_q$, and $A, B \in \mathbb{Z}_p^*$. Then the above is a zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$.*

We emphasize that Proposition 2 still requires the calling context to ensure that $a, b \in G_q$, and $A, B \in \mathbb{Z}_p^*$. There are obvious variations of the above, for proofs of knowledge involving more complex relations between logarithms.

5 Application to El Gamal Based Mix-Nets

In this section we describe how our ideas can be applied to break the robustness of most El Gamal based mix-nets.

5.1 The Generic Structure of El Gamal Based Mix-Nets

Suppose a set of senders $S_1, \ldots, S_N$ each have an input $m_i \in G_q$, and want to compute the *unordered* set $\{m_1, \ldots, m_N\}$, but they do not want to disclose which message m_i they sent. Furthermore, suppose that there is no trusted participant that can perform this service.

Given a bulletin board, a set of mix-servers $M_1, \ldots, M_k$ can provide the service by forming a mix-net. The bulletin board is defined as follows. Each participant may write on a dedicated area on the bulletin board, nobody can erase anything from the bulletin board, and everybody can read everything. The idea is that it should suffice if a subset of the mix-servers $M_1, \ldots, M_k$ are honest for the service to be provided securely, and robustly. The notion of a mix-net was introduced by Chaum [2], and further developed by a large number of people.

It is assumed that $M_1, \ldots, M_k$ have set up a distributed El Gamal cryptosystem, where M_j is somehow given a random $x_j \in \mathbb{Z}_q$, and $y_l = g^{x_l}$ is made public for $l = 1, \ldots, k$. We define a joint public key $y = \prod_{j=1}^{k} y_j$, with corresponding private key $x = \sum_{j=1}^{k} x_j$. It is also assumed that x_j is shared verifiably, secretly and robustly to all M_l for $l \neq j$. How this is done is of no importance to us in this paper, but it allows the robust elimination of any mix-server identified as a cheater.

The following is a generic description of an El Gamal based mix-net. First each sender S_i computes $(u_i, v_i) = E_y(m_i)$, and a proof of knowledge π_i of m_i (as described in Tsiounis and Yung [33] or Schnorr and Jakobsson [32]), and writes the pair $((u_i, v_i), \pi_i)$ on the bulletin board. Thus $((u_i, v_i), \pi_i)$ is a non-malleable [6] cryptotext in the random oracle model. Let $L_0 = \{(u_{0,i}, v_{0,i})\}$ be the set of cryptotexts for which π_i is valid. For $j = 1, \ldots, k$, M_j does the following:

1. It chooses $r_{j,i} \in \mathbb{Z}_q$ and $\phi_j \in \Sigma_N$ randomly, and computes the list $\{(g^{r_{j,i}} u_{j-1,i}, y^{r_{j,i}} v_{j-1,i})\}$. Then it uses ϕ_j to permute the pairs in this list and forms the list $L_j = \{(u_{j,i}, v_{j,i})\}$ that it writes on the bulletin board.
2. It proves in zero-knowledge that it performed step 1 correctly.

Then each M_j computes the list $\{B_{j,i}\} = \{u_{k,i}^{-x_j}\}$, and gives for each i a zero-knowledge proof that $\log_{u_{j,i}} B_{j,i} = \log_g y_j^{-1}$. Finally each M_j computes the list $\{v_{k,i} \prod_{j=1}^{k} B_{j,i}\}$, which is a permutation of the list of inputs $\{m_i\}$. The proof of knowledge produced by M_j is verified by all other mix-servers. If M_j produces a corrupt proof of knowledge its private key x_j is reconstructed using the secret sharing scheme and its computations are performed openly.

All mix-nets in the literature do not fit the above description, but the mix-net we consider in this paper, Furukawa and Sako [9] (and several others) do.

5.2 An Attack on the Robustness for El Gamal Based Mix-Nets

To break the robustness of most El Gamal based Mix-Nets it suffices to note that they almost without exception end with a standard joint decryption of the re-encrypted and randomly permuted inputs as outlined in the previous section. This applies not only to mix-nets on the form described above.

In most such mix-nets the mix-servers neglect to verify that $B_{j,i} \in G_q$. This implies that the adversary using the strategy of Section 4.2 can publish $B_{j,i} = \zeta u_{k,i}^{-x_j}$, where $1 \neq \zeta \in G_\kappa$, instead of $B_{j,i} = u_i^{-x_j}$ and produce a proof of knowledge deemed valid by all other mix-servers. Thus m_i is replaced by ζm_i in the output without detection and the robustness of the mix-net is broken.

This attack is applicable to the vast majority of El Gamal based mix-nets in the literature, e.g. [23, 13, 18, 22, 16, 30, 25, 1, 10], but there are also mix-nets that are not vulnerable to this attack, e.g. [5].

It may be argued that the attack is not particularly strong, since the fact that some output ζm_i of the mix-net does not belong to G_q is likely to be discovered, e.g. if some encoding of arbitrary strings into G_q is used.

6 Application to "An Efficient Scheme for Proving a Shuffle"

Consider the generic structure of an El Gamal based mix-net as presented in Section 5.1. The second step of the re-encryption and permutation stage states that the active mix-server M_j should prove in zero-knowledge that it performed the first step correctly.

Furukawa and Sako [9] provide exactly this. Given $\{(u_{j-1,i}, v_{j-1,i})\}$ and $\{(u_{j,i}, v_{j,i})\}$ they provide an efficient zero-knowledge proof of knowledge of $r_{j,i}$ and ϕ_j such that $\{(u_{j,i}, v_{j,i})\} = \{(g^{r_{j,i}} u_{j-1,\phi_j^{-1}(i)}, g^{r_{j,i}} v_{j-1,\phi_j^{-1}(i)})\}$. We have a closer look at their proof. To simplify we write $\{(u_i, v_i)\} = \{(u_{j-1,i}, v_{j-1,i})\}$, $\{(u'_i, v'_i)\} = \{(u_{j,i}, v_{j,i})\}$, $r_i = r_{j,i}$, and $\phi = \phi_j$. We also denote the prover M_j by P and any verifier, e.g. M_l for $l \neq j$, by V. In this section we denote by H a random oracle $H : \{0,1\}^* \to \mathbb{Z}_q^N$. Following [9] we define the permutation matrix $A = (A_{ij})$ by:

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

and let $\tilde{g}, \tilde{g}_1, \ldots, \tilde{g}_N \in G_q$ be a set of system wide uniformly and independently generated random elements, i.e. no participant knows any non-trivial relation among these elements. The $\tilde{g}, \tilde{g}_i$ elements are an integral part of the Furukawa-Sako protocol. Then the protocol proceeds as follows.

6.1 The Furukawa-Sako Proof of Knowledge

The list $g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}$ is the joint input to the prover P and the verifier V, and $\{r_i\}, \phi$ is the additional input to P. $\phi \in \Sigma_N$.

1. P chooses $\sigma, \rho, \tau, \alpha, \alpha_i, \lambda, \lambda_i \in \mathbb{Z}_q$ uniformly and independently at random.
2. P computes:

$$t = g^{\tau}, \quad \nu = g^{\rho}, \quad \omega = g^{\sigma}, \quad \mu = g^{\lambda}, \quad \mu_i = g^{\lambda_i}$$
$$\tilde{g}'_i = \tilde{g}^{r_i} \prod_{j=1}^{N} \tilde{g}_j^{A_{ji}}, \quad \tilde{g}' = \tilde{g}^{\alpha} \prod_{j=1}^{N} \tilde{g}_j^{\alpha_j}, \quad u' = g^{\alpha} \prod_{j=1}^{N} u_j^{\alpha_j}, \quad v' = y^{\alpha} \prod_{j=1}^{N} v_j^{\alpha_j}$$
$$\dot{\nu}_i = g^{\sum_{j=1}^{N} 3\alpha_j^2 A_{ji} + \rho r_i}, \quad \dot{\nu} = g^{\sum_{j=1}^{N} 3\alpha_j^3 + \tau\lambda + \rho\alpha},$$
$$\dot{\omega}_i = g^{\sum_{j=1}^{N} 2\alpha_j A_{ji} + \sigma r_i}, \quad \dot{\omega} = g^{\sum_{j=1}^{N} \alpha_j^2 + \sigma\alpha}, \quad t_i = g^{\sum_{j=1}^{N} 3\alpha_j A_{ji} + \tau\lambda_i}$$

3. P computes a challenge:

$$(c_1, \ldots, c_N) = H\big(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}\big) .$$

4. P computes $s = \alpha + \sum_{j=1}^{N} r_j c_j$, $s_i = \alpha_i + \sum_{j=1}^{N} A_{ij} c_j$, and $\lambda' = \lambda + \sum_{j=1}^{N} \lambda_j c_j^2$. The proof consists of the following tuple:

$$(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda') .$$

5. V computes

$$(c_1, \ldots, c_N) = H\big(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}\big) ,$$

and verifies that:

$$g^s \prod_{j=1}^{N} u_j^{s_j} = u' \prod_{j=1}^{N} u_j'^{c_j}, \quad y^s \prod_{j=1}^{N} v_j^{s_j} = v' \prod_{j=1}^{N} v_j'^{c_j}, \tag{1}$$

$$\tilde{g}^s \prod_{j=1}^{N} \tilde{g}_j^{s_j} = \tilde{g}' \prod_{j=1}^{N} \tilde{g}_j'^{c_j}, \quad g^{\lambda'} = \mu \prod_{j=1}^{N} \mu_j^{c_j^2},$$

$$\omega^s g^{\sum_{j=1}^{N}(s_j^2 - c_j^2)} = \dot{\omega} \prod_{j=1}^{N} \dot{\omega}_j^{c_j}, \quad \text{and} \quad t^{\lambda'} v^s g^{\sum_{j=1}^{N}(s_j^3 - c_j^3)} = \dot{\nu} \prod_{j=1}^{N} \dot{\nu}_j^{c_j} t_j^{c_j^2} \ .$$

Furukawa and Sako [9], present an interactive protocol corresponding to the above, but their intention [11] is that the protocol is made non-interactive using the Fiat-Shamir heuristic. Since honest verifier zero-knowledge with uniform challenges corresponds to zero-knowledge in the random oracle model their results imply the following.

Proposition 3. *Let $g, y, \tilde{g}_i, u_i, v_i, u_i', v_i' \in G_q$. If the verifier explicitly verifies that $t, \nu, \omega, \mu, \mu_i, \tilde{g}_i', \tilde{g}', u', v', t_i, \dot{\nu}_i, \dot{\nu}, \dot{\omega}_i, \dot{\omega} \in G_q$, then the Furukawa-Sako protocol is a proof of knowledge of $r_i \in \mathbb{Z}_q$ and $\phi \in \Sigma_N$ such that $\{(u_i', v_i')\} = \{(g^{r_i} u_{\phi_j^{-1}(i)}, g^{r_i} v_{\phi_j^{-1}(i)})\}$.*

6.2 Adversarial Strategy

In this section we describe an adversarial strategy for the Furukawa-Sako protocol. We stress that we have found no weakness in the proof of knowledge itself. Similarly as in Section 4 we instead exploit how this proof of knowledge is applied.

Let ζ be an element in G_τ of order ξ, and let $\{(u_i, v_i)\}$, and $\{(u_i', v_i')\}$ be as defined in the previous section. We consider three scenarios, where $l \in [N]$.

1. $v'_{\phi^{-1}(l)}$ is replaced by $\zeta^\iota v'_{\phi^{-1}(l)}$,
2. v_l is replaced by ζv_l, and
3. a combination of 1 and 2.

In each scenario the strategy convinces an honest verifier that the prover knows $r_i \in \mathbb{Z}_q$ and a permutation ϕ such that $\{(u_i', v_i')\} = \{(g^{r_i} u_{\phi^{-1}(i)}, g^{r_i} v_{\phi^{-1}(i)})\}$, despite that this is not even possible in Scenario 1 and 2.

The strategy is very simple. Repeatedly, i.e. at most a polynomial h number of times, run the first 4 steps of the Furukawa-Sako protocol of Section 6.1 on the corrupted input $g, y, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u_i', v_i')\}, \{r_i\}, \phi$ until it gives a proof that is deemed valid by the verifier. Then output the proof. It may seem that the strategy always succeeds in Scenario 3, but for similar reasons as in Section 4.2 this is not true. A proof of the following lemma is given in Appendix A.

Lemma 2. *Each iteration in Scenario 1 and Scenario 2 succeeds with probability at least $\frac{1}{\xi} + O(1/q)$. Each iteration in Scenario 3 succeeds with probability at least $\frac{1}{2}$. In all scenarios the adversarial strategy fails with negligible probability.*

Numerous variants of above three scenarios may be considered, e.g. several elements may be multiplied by ζ, or different $\zeta_i \in G_\kappa$ may be used for different elements. For most such variants the probability of failure in each iteration grows rapidly, but we present one interesting exception. Consider the following three scenarios, where $l_1, \ldots, l_t \in [N]$ are all unique:

4. $v'_{\phi^{-1}(l_\iota)}$ is replaced by $\zeta v'_{\phi^{-1}(l_\iota)}$,
5. v_{l_ι} is replaced by ζv_{l_ι}, and
6. a combination of 4 and 5.

A proof of the following Corollary is given in Appendix A.

Corollary 1. *If $\xi = 2$ each iteration in Scenario 4,5, and 6 fails with probability at most $\frac{1}{2} + O(1/q)$, and the adversarial strategy fails with negligible probability.*

6.3 Attack for the Generic Mix-Net Using Furukawa-Sako Proofs

In this section we describe an attack for the mix-net defined in Section 5.1, where the the Furukawa-Sako proof of knowledge is used in Step 2 by each M_j during the re-encryption and permutation stage. The goal of the adversary is to break the privacy of any particular sender S_t.

Let $1 \neq \zeta \in G_\tau$. The adversary corrupts the first and last mix-servers M_1 and M_k, and as many other arbitrarily chosen mix-servers as possible. The execution then proceeds as follows.

1. M_1 simulates the first step of an honest mix-server, but it does not write L_1 on the bulletin board. Instead it forms the list:
$$L'_1 = \{(u_{1,1}, v_{1,1}), \ldots, (u_{1,\phi_1(z)}, \zeta v_{1,\phi_1(z)}), \ldots, (u_{1,N}, v_{1,N})\}$$
and writes L'_1 on the bulletin board, i.e. it multiplies the second component of the El Gamal pair of S_1 by ζ. Then it uses the adversarial strategy of Section 6.2 on the corrupt input $g, y, \{\tilde{g}_i\}, L_0, L'_1, \{r_{1,i}\}, \phi_1$ to construct a proof deemed valid.
2. Each honest mix-server executes its re-encryption and permutation step honestly.
3. Each corrupt mix-server except M_1 and M_k first simulates the first step of an honest mix-server. Then it uses the adversarial strategy of Section 6.2 to construct a proof that is deemed valid.
4. M_k simulates the first step of an honest mix-server, but it does not write L_k on the bulletin board. Instead it finds an l such that $v_{k,l}^q \neq 1$. Finally it forms the corrupted list
$$L'_k = \{(u_{k,1}, v_{k,1}), \ldots, (u_{k,l}, \zeta^{-1} v_{k,l}), \ldots, (u_{k,N}, v_{k,N})\}$$
and writes L'_k on the bulletin board. Then it uses the adversarial strategy of Section 6.2 on the corrupt input $g, y, \{\tilde{g}_i\}, L_{k-1}, L'_k, \{r_{k,i}\}, \phi_k$ to construct a proof deemed valid by any verifier.

5. All mix-servers, including M_1 and M_k, jointly decrypt the El Gamal pairs of the list L'_k, and publish the result $\{m'_1, \dots, m'_N\}$, i.e. a permutation of $\{m_1, \dots, m_N\}$.
6. The adversary concludes that S_z was the sender of m'_l.

We prove the following in Appendix A.

Proposition 4. *Let k' be the number of honest mix-servers. Then the attack succeeds with probability at least $2^{-k'}$.*

Corollary 2. *Proposition 4 corresponds to scenarios 1,2, and 3 in Section 6.2. For the attack similar to the above corresponding to scenarios 4,5, and 6 the probability of success is at least $2^{-k'}$.*

The proposition says that we can break the privacy of S_z with probability $2^{-k'}$. The corollary states that the adversary may establish a correspondence between a subset $\{S_{z_1}, \dots, S_{z_\iota}\}$ of the senders and a subset $\{m_{l_1}, \dots, m_{l_\iota}\}$ of the messages output from the mix-net, such that there is a bijection between the two sets. It does not say that we can break the privacy of t individual senders completely.

In most applications k, and thereby k', is relatively small, e.g. $k = 5$ and $k' = 3$. Thus the attack is quite practical in terms of success probability. On the other hand, if the adversarial strategy fails some honest mix-server produces a corrupt proof, and is accused of cheating. It is likely that the honest mix-server accused of cheating performs an investigation to find out how it was "framed", and discover the fact that its inputs are not in G_q. If this is done the corrupted mix-servers are eventually identified.

Apparently [11] Furukawa et al. do perform some verifications of inputs in later work [10]. The protocol described in [10] is not vulnerable to the above attack, but it is still vulnerable to the attack of Section 5.2.

In the next two sections we show how the Furukawa-Sako proof of knowledge can be strengthened, and the generic mix-net based on this proof modified to efficiently counter our attack.

6.4 A Strengthened Furukawa-Sako Proof of Knowledge

Consider the adversarial strategy of Section 6.2. The problem is that formally the verifier is required to verify that all components expected to reside in G_q indeed do so. Since this is costly we want to avoid this. One way to solve the problem is to go through the details of the proofs of Furukawa and Sako [9], and verify that the protocol is still a proof of knowledge under the weaker assumption that some elements are only contained in $\mathbb{Z}_p^*$. Another way of solving the problem that illustrates our generic recipe is the following.

Let $g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\} \in G_q$ be the joint input to the prover P and the verifier V, and let $\{r_i\}, \phi$ be the additional input to the prover. The strengthened Furukawa-Sako protocol proceeds as follows.

1. P computes $(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda')$ so the following is deemed a valid proof of knowledge by the original verifier: $(t^\kappa, \nu^\kappa, \omega^\kappa, \mu^\kappa, \{\mu_i^\kappa\}, \{\tilde{g}_i'^\kappa\}, \tilde{g}'^\kappa, u'^\kappa, v'^\kappa, \{t_i^\kappa\}, \{\dot{\nu}_i^\kappa\}, \dot{\nu}^\kappa, \{\dot{\omega}_i^\kappa\}, \dot{\omega}^\kappa, s, \{s_i\}, \lambda')$.

2. V computes the list $(t^\kappa, \nu^\kappa, \omega^\kappa, \mu^\kappa, \{\mu_i^\kappa\}, \{\tilde{g}_i'^\kappa\}, \tilde{g}'^\kappa, u'^\kappa, v'^\kappa, \{t_i^\kappa\}, \{\dot{\nu}_i^\kappa\}, \dot{\nu}^\kappa, \{\dot{\omega}_i^\kappa\}, \dot{\omega}^\kappa, s, \{s_i\}, \lambda')$ and verifies that the original verifier accepts this as a proof.

Intuitively this may be viewed as if the verifier takes part of the construction of the proof such that it is ensured that elements are in G_q. In Appendix A we give details for how the prover using some simple tricks can compute a list with the above property efficiently, and also prove the following.

Proposition 5. *Let $g, y, \tilde{g}, \tilde{g}_i, u_i, v_i, u_i', v_i' \in G_q$. Then the strengthened protocol above is a proof of knowledge of $r_i \in \mathbb{Z}_q$ and $\phi \in \Sigma_N$ such that $\{(u_i', v_i')\} = \{(g^{r_i} u_{\phi_j^{-1}(i)}, g^{r_i} v_{\phi_j^{-1}(i)})\}$.*

The prover computes $2N+6$ additional κ-multiplications in $\mathbb{Z}_q$, and the prover and verifier each computes $5N+9$ additional κ-exponentiations respectively, compared to the original protocol.

6.5 A Strengthened Generic Mix-Net Using Strengthened Furukawa-Sako Proofs of Knowledge

The attack of Section 6.3 exploits that the mix-servers do not verify that all components of their input are members of G_q. Since it is costly to verify this explicitly we would like to avoid it. Below we present a generic mix-net based on the strengthened Furukawa-Sako proof of knowledge that does this.

We assume that each mix-server explicitly verifies that $g, y \in G_q$. Then let $h, h_1, \ldots, h_N \in \mathbb{Z}_p^*$ be randomly chosen, and define $\tilde{h} = h^\kappa$, $\tilde{h}_i = h_i^\kappa$. The purpose of introducing $\tilde{h}$ and $\tilde{h}_i$ is to improve efficiency of the strengthened Furukawa-Sako proof (see the proof of Proposition 5 in Appendix A for details).

Let $L_0 = \{(u_{0,i}, v_{0,i})\}$ be the set of cryptotexts for which π_i is valid. For $j = 1, \ldots, k$, M_j does the following:

1. It computes the list $L_{j-1}^\kappa = \{(u_{j-1,i}^\kappa, v_{j-1,i}^\kappa)\}$. Then it chooses $r_{j,i} \in \mathbb{Z}_q$ and $\phi_j \in \Sigma_N$ randomly, and computes the list $\{(g^{r_{j,i}} u_{j-1,i}^\kappa, y^{r_{j,i}} v_{j-1,i}^\kappa)\}$. Finally it uses ϕ_j to permute the pairs in this list and forms the list $L_j = \{(u_{j,i}, v_{j,i})\}$ that it writes on the bulletin board.
2. It proves in zero-knowledge that it performed step 1 correctly by executing the strengthened Furukawa-Sako proof of knowledge of Section 6.4 with input $g, y, \tilde{h}^\kappa, \{\tilde{h}_i^\kappa\}, L_{j-1}^{\kappa^2}, L_j^\kappa, \{\kappa r_{j,i}\}, \phi_j$.

To verify the proof of M_l the verifier first computes $L_{l-1}^{\kappa^2}$ and L_l^κ from L_{l-1} and L_l respectively, and then verify the strengthened Furukawa-Sako proof. From Section 3 follows that all components of $L_{l-1}^{\kappa^2}$ and L_l^κ belong to G_q, which means that Proposition 6.4 can be applied.

At the end of the re-encryption and permutation phase each M_j computes the list $L_k^\kappa = \{(u_{k,i}^\kappa, v_{k,i}^\kappa)\}$. Then each M_j computes the list $\{B_{j,i}\} = \{(u_{k,i}^\kappa)^{-x_j}\}$, and gives for each i a zero-knowledge proof that $\log_{u_{j,i}^\kappa} B_{j,i} = \log_g y_j^{-1}$ using the strengthened protocol of Section 4.3. Finally each M_j computes the output by forming the list $\{(v_{k,i}^\kappa \prod_{j=1}^k B_{j,i})^{1/\kappa^{k+1}}\}$, which is a permutation of $\{m_i\}$.

Each mix-server computes $2(N+1)$ additional κ-exponentiations to construct the elements $\tilde{h}^{\kappa}$ and $\tilde{h}_i^{\kappa}$. Each mix-server computes $6N$ additional κ-exponentiations to ensure that its inputs belongs to G_q, and to prepare for computing the proof. By Proposition 5 the strengthened proof requires $5N+9$ additional κ-exponentiations. To prepare for the verification of all proofs each mix-server computes $2kN$ additional κ-exponentiations and $2(k-1)N$ additional κ^2-exponentiations. By Proposition 5 the verifier computes $5kN+9k$ additional κ-exponentiation to verify the proofs. Each mix-server also computes N additional exponentiations during the joint decryption. In total the complexity of each mix-server increases by approximately N full exponentiations, $2kN$ κ^2-exponentiations, and $(7k+13)N$ κ-exponentiations.

7 Further Applications

We expect that natural adaptations of the adversarial strategy for the proofs of knowledge of logarithms can be used to break other protocols.

The attack for the mix-net based on the Furukawa-Sako proof of knowledge is an extension of the attacks for Golle et al. [13], Jakobsson [18], and Mitomo and Kurosawa [22] presented by Wikström [35]. It may be possible to apply this attack to further mix-nets. In particular it seems possible to give a purely theoretical attack for the mix-net of Neff [23].

An interesting open question is in what other ways and for which protocols the adversary may exploit elements in $\mathbb{Z}_p^* \backslash G_q$.

8 Conclusion

We have identified a previously overlooked weakness in several cryptographic protocols, and introduced a novel class of attacks. The basic idea of the attack is general, and applicable to a broad range of protocols based on the discrete logarithm problem.

The general problem of elements in $\mathbb{Z}_p^* \backslash G_q$ has been described and discussed, and we have also given a recipe for how explicit verifications of membership in G_q can be avoided efficiently.

To illustrate both the problem and the solution we have provided some examples of protocols that are vulnerable to the novel attack, and shown how they can be modified to efficiently counter the attack. In particular we have broken the robustness of most of the El Gamal based mix-nets in the literature, and broken the privacy of the generic mix-net based on the proof of correct shuffle of Furukawa and Sako [9].

Our results indicate that many protocols in the literature need a review. The complexity estimates must be revised to acknowledge the need for costly explicit verifications, or the proofs of security must be verified to hold under the weaker assumption that elements are in $\mathbb{Z}_p^*$ instead of G_q, or the protocols must be modified such that it is ensured that elements are in G_q and explicit verifications are avoided.

9 Acknowledgements

Johan Håstad helped with the original attack in Wikström [35] and inspired me to investigate the idea of elements in G_κ further. Torsten Ekedahl provided a technical lemma. I had discussions with Gunnar Sjödin. Jun Furukawa answered all my questions about their protocol. Markus Jakobsson answered some of my initial questions. I am grateful to all of them.

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. R. Cramer, R. Gennaro, and B. Schoenmakers, *A secure and optimally efficient multi-authority election scheme*, Eurocrypt '97, pp. 103-118, LNCS 1233, 1997.
5. Y. Desmedt, K. Kurosawa, *How to break a practical MIX and design a new one*, Eurocrypt 2000, pp. 557-572, LNCS 1807, 2000.
6. 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.
7. 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.
8. A. Fujioka, T. Okamoto and K. Ohta, *A practical secret voting scheme for large scale elections*, Auscrypt '92, LNCS 718, pp. 244-251, 1992.
9. J. Furukawa, K. Sako, *An efficient scheme for proving a shuffle*, Crypto 2001, LNCS 2139, pp. 368-387, 2001.
10. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, K. Sako, *An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling*, Financial Crypto 2002.
11. J. Furukawa, *Personal communication*, email, 13-25 february, 2003.
12. S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences (JCSS), Vol. 28, No. 2, pp. 270-299, 1984.
13. Golle, Zhong, Boneh, Jakobsson, Juels, *Optimistic Mixing for Exit-Polls*, Asiacrypt 2002, LNCS, 2002.
14. Golle, Zhong, Boneh, Jakobsson, Juels, *Private Communication*, 16 October 2002.
15. M. Hirt, K. Sako, *Efficient Reciept-Free Voting Based on Homomorphic Encryption*, Eurocrypt 2000, LNCS 1807, pp. 539-556, 2000.
16. M. Jakobsson, *A Practical Mix*, Eurocrypt '98, LNCS 1403, pp. 448-461, 1998.
17. 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.
18. M. Jakobsson, *Flash Mixing*, In Proceedings of the 18:th ACM Symposium on Principles of Distributed Computing - PODC '98, pp. 83-89, 1998.
19. M. Jakobsson, A. Juels, *Millimix: Mixing in small batches*, DIMACS Techical report 99-33, June 1999.

20. 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.
21. 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.
22. M. Mitomo, K. Kurosawa, *Attack for Flash MIX*, Asiacrypt 2000, pp. 192-204, LNCS 1976, 2000.
23. 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.
24. V. Niemi, A. Renvall, *How to prevent buying of votes in computer elections*, Asiacrypt'94, LNCS 917, pp. 164-170, 1994.
25. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, *Fault Tolerant Anonymous Channel*, Information and Communications Security - ICICS '97, pp. 440-444, LNCS 1334, 1997.
26. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Eurocrypt '99, LNCS 1592, pp. 223-238, 1999.
27. C. Park, K. Itoh, K. Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Eurocrypt '93, LNCS 765, pp. 248-259, 1994.
28. B. Pfitzmann, *Breaking an Efficient Anonymous Channel*, Eurocrypt '94, LNCS 950, pp. 332-340, 1995.
29. B. Pfitzmann, A. Pfitzmann, *How to break the direct RSA-implementation of mixes*, Eurocrypt '89, LNCS 434, pp. 373-381, 1990.
30. K. Sako, J. Killian, *Reciept-free Mix-Type Voting Scheme*, Eurocrypt '95, LNCS 921, pp. 393-403, 1995.
31. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, No. 4, pp. 161-174, 1991.
32. C. Schnorr, M. Jakobsson, *Security of Signed El Gamal Encryption*, Asiacrypt 2000, LNCS 1976, pp. 73-89, 2000.
33. Yiannis Tsiounis, Moti Yung, *On the Security of El Gamal based Encryption*, International workshop on Public Key Cryptography, LNCS 1431, pp. 117–134, 1998.
34. D. Wagner *A Generalized Birthday Problem*, Crypto 2002, LNCS 2442, pp. 288-304, 2002.
35. D. Wikström, *Four Attacks for "Optimistic Mixing For Exit-Polls"*, submitted to Crypto.

A Proofs

To prove Lemma 1 we need the following technical lemma that was kindly provided by Torsten Ekedahl. Any errors are due to us.

Lemma 3. *Let a, b and c be uniformly and independently distributed in $[q]$, where q is prime. Let ξ be a constant prime. Define $d = ab + c$, and write d' and m_d for the unique integers such that $d = d' + m_d q$ and $0 \le d' < q$.*

Then $\Pr[d' = d \pmod{\xi}] = \frac{1}{\xi} + O(\log^2 q/q)$.

Proof. Let $a, b \in \mathbb{Z}$, and write $a = a' + m_a q$, $b = b' + m_b q$ where $0 \le a', b' < q$. If $a = b \pmod{q\xi}$, then $a' = a \pmod{\xi}$ implies that $b' = b \pmod{\xi}$.

Thus we consider the set $S = \{a \in \mathbb{Z}_{q\xi} \mid a = a' + m_a q \text{ and } a' = a \pmod{\xi}\}$. Let $a \in \mathbb{Z}_{q\xi}$, and write $a = a' + m_a q$. Then $a' = a \pmod{\xi}$ is equivalent to $m_a q = 0 \pmod{\xi}$, but since q is prime and $m_a < \xi$ this implies that $m_a = 0$. We conclude that $S = \{0, \ldots, q-1\}$.

Let χ_S be the characteristic function of S. We compute in the ring $\mathbb{Z}_{q\xi}$, and the probability we are investigating may be written:

$$\Pr[d' = d \pmod{\xi}] = \frac{1}{q^3} \sum_{a,b,c \in [q]} \chi_S(ab + c)$$

where $ab+c$ in $\chi_S(ab+c)$ is computed in $\mathbb{Z}_{q\xi}$. Set $\Delta = e^{2\pi i/q\xi}$. Then the functions $\alpha \mapsto \Delta^{\alpha\beta}$ are orthogonal under the inner product $\langle f, g\rangle = \frac{1}{q\xi}\sum_{\alpha \in \mathbb{Z}_{q\xi}} f(\alpha)\overline{g(\alpha)}$, and we may apply the discrete Fourier transform to χ_S, and write:

$$\chi_S(\alpha) = \sum_{\beta \in \mathbb{Z}_{q\xi}} f_\beta \Delta^{\alpha\beta}, \text{ where } f_\beta = \frac{1}{qt} \sum_{\gamma \in \mathbb{Z}_{qt}} \chi_S(\gamma)\Delta^{-\beta\gamma} \quad .$$

Note that $f_0 = \frac{|S|}{qt} = 1/t$. Thus we may bound the error by:

$$\frac{1}{q^3} \sum_{a,b,c \in [q]} (\chi_S(ab + c) - 1/t) = \frac{1}{q^3} \sum_{a,b,c \in \mathbb{Z}_{q\xi}} (\chi_S(ab + c) - 1/t)\chi_S(a)\chi_S(b)\chi_S(c)$$

$$= \frac{1}{q^3} \sum_{\alpha \neq 0, \beta,\gamma,\delta \in \mathbb{Z}_{q\xi}} f_\alpha f_\beta f_\gamma f_\delta \sum_{a,b,c \in \mathbb{Z}_{q\xi}} \Delta^{(ab+c)\alpha + a\beta + b\gamma + c\delta}$$

Since $\sum_{c \in \mathbb{Z}_{q\xi}} \Delta^{c(\alpha+\delta)} = q\xi$ if $\alpha + \delta = 0$ and 0 otherwise the above equals:

$$\frac{\xi}{q^2} \sum_{\alpha \neq 0, \beta, \gamma \in \mathbb{Z}_{q\xi}} |f_\alpha|^2 f_\beta f_\gamma \sum_{a,b \in \mathbb{Z}_{q\xi}} \Delta^{\alpha ab + \beta a + \gamma b} \quad ,$$

where we use $f_{-\alpha} = \overline{f_\alpha}$. We have that $\mathbb{Z}_{q\xi}$ and $\mathbb{Z}_q \times \mathbb{Z}_\xi$ are isomorphic as rings, i.e. we may write $1 = m_\xi \xi + m_q q$ for some m_ξ and m_q. If we set $\Delta_q = e^{2\pi i m_q/q}$, and $\Delta_\xi = e^{2\pi i m_\xi/\xi}$, we have $\Delta = \Delta_q \Delta_\xi$. Thus we may write:

$$\sum_{a,b \in \mathbb{Z}_{q\xi}} \Delta^{\alpha ab + \beta a + \gamma b} = \sum_{a',b' \in \mathbb{Z}_q} \Delta_q^{\alpha a' b' + \beta a' + \gamma b'} \sum_{a'',b'' \in \mathbb{Z}_\xi} \Delta_\xi^{\alpha a'' b'' + \beta a'' + \gamma b''} \quad .$$

The norm of the right factor is clearly bounded by ξ^2, and we focus on the left factor. If $\alpha = \beta = \gamma = 0$ (in $\mathbb{Z}_q$ that is) the left factor equals q^2. Otherwise, if $\alpha = 0$, $\beta = 0$, or $\gamma = 0$ it equals 0. The remaining case $\alpha, \beta, \gamma \neq 0$ is considered next. Summing over λb instead of over b corresponds to changing (α, β, γ) to $(\lambda\alpha, \beta, \lambda\gamma)$, and summing over λa instead of over a corresponds to changing (α, β, γ) to $(\lambda\alpha, \lambda\beta, \gamma)$. Thus we may replace (α, β, γ) by $(\beta, \beta, \beta\gamma/\alpha)$, and then by $(\beta\gamma/\alpha, \beta\gamma/\alpha, \beta\gamma/\alpha)$ without changing the sum. Replacing $\varDelta_q$ with $\varDelta_q^{\beta\gamma/\alpha}$ corresponds to replacing $(\beta\gamma/\alpha, \beta\gamma/\alpha, \beta\gamma/\alpha)$ with $(1, 1, 1)$, and does not change the sum, since $\beta\gamma/\alpha$ is a generator in $\mathbb{Z}_q$. Thus the left sum equals $(\varDelta_q)^{-1} \sum_{a,b \in \mathbb{Z}_q} (\varDelta_q)^{ab} = q$.

Going back to the original sum we have $|f_\alpha| = \frac{1}{q\xi} | \sum_{\gamma \in \mathbb{Z}_{q\xi}} \chi_S(\gamma) \varDelta^{-\alpha\gamma}| = \frac{1}{q\xi} | \sum_{\gamma \in S} \varDelta^{-\alpha\gamma}| = \frac{1}{q\xi} |\frac{\varDelta^{-q\alpha}-1}{\varDelta^{-\alpha}-1}| \leq \frac{1}{q\xi} \frac{2}{|\varDelta^{-\alpha}-1|} \leq \frac{1}{|\alpha|}$, if $\alpha \neq 0$, since $|\varDelta^{-\alpha} - 1| = \frac{2\pi}{q\xi}\alpha + O(1/q^2)$. We also have $|f_\alpha^2 f_\beta f_\gamma| \leq 1$. This gives us:

$$
\begin{aligned}
&\frac{\xi}{q^2} \left| \sum_{\alpha \neq 0, \beta, \gamma \in \mathbb{Z}_{q\xi}} |f_\alpha|^2 f_\beta f_\gamma \sum_{a,b \in \mathbb{Z}_{q\xi}} \varDelta^{\alpha ab + \beta a + \gamma b} \right| \\
&\leq \frac{\xi}{q^2} \left| \sum_{\alpha \neq 0, q | \alpha, \beta, \gamma} |f_\alpha|^2 f_\beta f_\gamma q^2 \xi^2 + \sum_{q \nmid \alpha, \beta, \gamma} |f_\alpha|^2 f_\beta f_\gamma q \xi^2 \right| \\
&\leq \frac{\xi}{q^2} \left(\frac{1}{q} q^2 \xi^2 + q \log^2 q \xi^2 \right) = \xi^3 \frac{(1 + \log q)}{q} = O(\log q / q) \enspace .
\end{aligned}
$$

Proof (Lemma 1). Consider Scenario 1, where $A = za^\gamma$. The proof (α, β, d) is deemed valid if $A^\theta \alpha = \zeta^\theta a^{\gamma\theta} a^\delta = a^d$. This happens if $\zeta^\theta = 1$, i.e. with probability $\frac{1}{\xi} + O(1/q)$. Similarly, for Scenario 2, the proof is deemed valid if $\zeta^\delta = \zeta^d$. This also happens with probability $\frac{1}{\xi} + O(1/q)$, since δ and d are independently distributed.

In Scenario 3, the proof is deemed valid if $\zeta^\theta \zeta^{\gamma\delta} = \zeta^d$. If $\gamma = 0 \pmod{\xi}$ this reduces to Scenario 2. Otherwise, we have $\theta + \gamma\delta = d + m_d q$ for some m_d. The equation holds precisely when $d = \theta + \gamma\delta \pmod{\xi}$. Thus we may apply Lemma 3 to conclude that the probability that this happens is $\frac{1}{\xi} + O(\log^2 q/q)$.

From independence of the iterations follows that the strategy fails with negligible probability.

Proof (Proposition 2). First we prove that a proof is not valid if either A or B is in $\mathbb{Z}_p^* \backslash G_q$. We may by the structure of $\mathbb{Z}_p^*$ write $A = A'\zeta$ for some $A \in G_q$ and $1 \neq \zeta \in G_\kappa$. This implies that $A^{\theta^*} = (A')^{\theta^*} z^{\theta^*}$. We have $(A')^{\theta^*} \in G_q$, and since $\gcd(\theta^*, \kappa) = 1$ by construction we have $1 \neq \zeta^{\theta^*} \in G_\kappa$. Thus if $A \in \mathbb{Z}_p^* \backslash G_q$ then $A^{\theta^*} \in \mathbb{Z}_p^* \backslash G_q$. Similarly, since $\alpha \in \mathbb{Z}_p^*$ we may write $\alpha = \alpha'\zeta$ for some $\alpha' \in G_q$ and $\zeta \in G_\kappa$. This implies that $\alpha^\kappa = (\alpha')^\kappa \zeta^\kappa = (\alpha')^\kappa \in G_q$. Intuitively we may view this as if the prover and verifier jointly generated a uniformly distributed element $\alpha^\kappa \in G_q$. It is clear that $a^d \in G_q$. Thus if $A \in \mathbb{Z}_p^* \backslash G_q$, then we have that $A^{\theta^*} \alpha^\kappa \in \mathbb{Z}_p^* \backslash G_q$, which clearly can not equal $a^d \in G_q$ in $\mathbb{Z}_p^*$. The proof for B is similar.

The extractor of the above protocol is almost identical to the extractor of the original protocol. The only difference is that when $\alpha, \beta \in G_q$ and $\theta_0, \theta_1 \in \mathbb{Z}_q$ are found such that the prover outputs valid proofs (α, β, d_0) and (α, β, d_1) given θ_0 or θ_1 as answer to the question $(\alpha, \beta, A, B, a, b)$ the extractor solves the equation system $\{d_b = \gamma\theta_b^* + \kappa\delta\}_{b\in\{0,1\}}$ in the unknowns γ and δ.

The simulator is almost identical to the simulator of the original protocol. Since κ is a generator of the additive group $\mathbb{Z}_q$, and δ is uniformly distributed in $\mathbb{Z}_q = [q]$, d is uniformly distributed in $\mathbb{Z}_q$. Thus the simulator chooses $d, \theta \in \mathbb{Z}_q$ uniformly at random and defines $\alpha = (a^d/A^{\theta^*})^{1/\kappa}$, and $\beta = (b^d/B^{\theta^*})^{1/\kappa}$. It follows that the transcript (α, β, d) is correctly distributed.

Lemma 4. *Let a and b be uniformly distributed in $[q]$, and write $c = a + b$ and $c = c' + m_c q$, where $m_c \in \{0, 1\}$ and $0 \leq c' < q$. Then we have*
$\Pr[c' = c \pmod q] = \frac{1}{2} + \frac{1}{2q}$.

Proof. If $c < q$ we have $c' = c = a + b$, and there are $\frac{q^2+q}{2}$ such pairs (a, b). Otherwise we have $c = c' + q \neq c' \pmod \xi$, since $q \neq 0 \pmod \xi$. Thus, the probability is $\frac{1}{2} + \frac{1}{2q}$.

Proof (Lemma 2). Consider Scenario 1. The only equation involving $v_{\phi^{-1}(l)}$ in the verification procedure is Equation (1). Thus it suffices to consider this equation. By construction we may rewrite Equation (1) as

$$y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{c_{\phi^{-1}(l)}} v' \prod_{j=1}^{N} v_j'^{c_j} \quad .$$

Thus, if $\zeta^{c_{\phi^{-1}(l)}} = 1$, Equation (1) holds, and this happens at least with probability $\frac{1}{\xi} + O(1/q)$. The analysis for Scenario 2 is similar.

For Scenario 3 we note that by construction we may rewrite Equation (1) as

$$\zeta^{s_l} y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{c_{\phi^{-1}(l)}} \zeta^{\alpha_l} v' \prod_{j=1}^{N} v_j'^{c_j} \quad .$$

Thus, if $\zeta^{s_l} = \zeta^{c_{\phi^{-1}(l)}} \zeta^{\alpha_l}$, Equation (1) holds. This event occurs exactly when $s_l = c_{\phi^{-1}(l)} + \alpha_l \bmod \xi$, and by definition there is an m such that $s_l + mq = c_{\phi^{-1}(l)} + \alpha_l$. We apply lemma 4 to conclude that the probability of this event is at least $\frac{1}{2}$.

From independence follows that the adversarial strategy fails with negligible probability.

Proof (Corollary 1). Similarly as for Scenario 1 we may rewrite Equation (1) as $y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{\sum_{\iota=1}^{t} c_{\phi^{-1}(l_\iota)}} v' \prod_{j=1}^{N} v_j'^{c_j}$. If $\zeta^{\sum_{\iota=1}^{t} c_{\phi^{-1}(l_\iota)}} = 1$, Equation (1) holds, and this happens at least with probability $\frac{1}{\xi} + O(1/q)$. The analysis for Scenario 5 is similar.

For Scenario 6 we note that by construction we may rewrite Equation (1) as $\zeta^{\sum_{\iota=1}^{t} s_{l_\iota}} y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{\sum_{\iota=1}^{t} c_{\phi^{-1}(l_\iota)}} \zeta^{\sum_{\iota=1}^{t} \alpha_{l_\iota} \alpha_l} v' \prod_{j=1}^{N} v_j'^{c_j}$. Equation (1) holds if we have $\zeta^{\sum_{\iota=1}^{t} s_{l_\iota}} = \zeta^{\sum_{\iota=1}^{t} c_{\phi^{-1}(l_\iota)}} \zeta^{\sum_{\iota=1}^{t} \alpha_{l_\iota}}$, . This event occurs exactly when $\sum_{\iota=1}^{t} s_{l_\iota} = \sum_{\iota=1}^{t} (c_{\phi^{-1}(l_\iota)} + \alpha_{l_\iota}) \bmod 2$. Write $c_{\phi^{-1}(l_\iota)} + \alpha_{l_\iota} = a_\iota + a'_\iota q$, where $0 \leq a_\iota < q$. It follows that the congruence holds precisely when $\sum_{\iota=1}^{t} a'_\iota$ is even. This clearly happens with probability $\frac{1}{2} + \epsilon$ where ϵ is negligible.

From independence follows that the adversarial strategy fails with negligible probability.

Proof (Proposition 4 (Corollary 2)). We give a joint proof for both the proposition and the corollary. We must show that the attack is not detected, and that m'_l (or $m'_{l_1}, \ldots, m_{l_t}$) indeed is the message sent by S_z (or $S_{z_1}, \ldots, S_{z_t}$) with high probability.

From Lemma 2 (Corollary 1) follows that the proofs of knowledge produced by the corrupt mix-servers, including M_1 and M_k, are deemed valid by any verifier with overwhelming probability. Honest mix-servers on the other hand essentially computes a single iteration of the adversarial strategy in Scenario 3 (Scenario 6) defined in Section 6.2. From Lemma 2 (Corollary 1) follows that the proof produced by an honest mix-server is valid with probability at least $\frac{1}{2}$. From independence follows that the attack is detected with probability at most $2^{-k'}$.

Since all mix-servers except M_1 and M_k behave honestly in the first step, there is a unique $l_j \in \{1, \ldots, N\}$ ($l_{j,\iota} \in \{1, \ldots, N\}$ for $\iota = 1, \ldots, t$) for each $j = 1, \ldots, k$ such that $v_{j,l_j} \notin G_q$ ($v_{j,l_{j,\iota}} \notin G_q$ for $\iota = 1, \ldots, t$), and by construction we have $l = \phi_1^{-1}(z)$ ($l_\iota = \phi_1^{-1}(z_\iota)$ for $\iota = 1, \ldots, t$). We have that $v_{k,l}^q \neq 1$ precisely when $v_{k,l}^q \neq 1$. Thus the adversary identifies the message m'_l sent by S_z (the *set* of messages $\{m_{z_1}, \ldots, m_{z_t}\}$ sent by $S_{z_1}, \ldots, S_{z_t}$) correctly.

Proof (Proposition 5). The simulator of the strengthened protocol runs the original simulator to get a valid proof

$$(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda')$$

for the original proof system and then outputs:

$$\begin{aligned}(&t^{1/\kappa}, \nu^{1/\kappa}, \omega^{1/\kappa}, \mu^{1/\kappa}, \{\mu_i^{1/\kappa}\}, \{\tilde{g}_i'^{1/\kappa}\}, \tilde{g}'^{1/\kappa}, u'^{1/\kappa}, v'^{1/\kappa}, \{t_i^{1/\kappa}\},\\ &\{\dot{\nu}_i^{1/\kappa}\}, \dot{\nu}^{1/\kappa}, \{\dot{\omega}_i^{1/\kappa}\}, \dot{\omega}^{1/\kappa}, s, \{s_i\}, \lambda') \ .\end{aligned}$$

This gives a correctly distributed transcript.

The extractor of the modified protocol runs the original extractor, and simulates an original prover to the original extractor as follows. If the modified prover outputs $(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda')$ it gives the list $(t^\kappa, \nu^\kappa, \omega^\kappa, \mu^\kappa, \{\mu_i^\kappa\}, \{\tilde{g}_i'^\kappa\}, \tilde{g}'^\kappa, u'^\kappa, v'^\kappa, \{t_i^\kappa\}, \{\dot{\nu}_i^\kappa\}, \dot{\nu}^\kappa, \{\dot{\omega}_i^\kappa\}, \dot{\omega}^\kappa, s, \{s_i\}, \lambda')$ to the original extractor. It follows that the modified extractor is successful if the original extractor is.

Now we give details on how the prover may form a modified proof without much additional effort. First of all we let $\tilde{h}, \tilde{h}_1, \ldots, \tilde{h}_N \in G_q$ be a set of uniformly and independently distributed elements, i.e. no participant knows any non-trivial relation among these elements. Then we define $\tilde{g} = \tilde{h}^\kappa$, and $\tilde{g}_i = \tilde{h}_i^\kappa$ for $i = 1, \ldots, N$. Thus we have $\tilde{h} = \tilde{g}^{1/\kappa}$, and $\tilde{h}_i = \tilde{g}_i^{1/\kappa}$, which is very useful. We define the modified prover below.

1. P chooses $\sigma, \rho, \tau, \alpha, \alpha_i, \lambda, \lambda_i \in \mathbb{Z}_q$ uniformly and independently at random.
2. P computes:

$$t = g^\tau, \quad \nu = g^\rho, \quad \omega = g^\sigma, \quad \mu = g^\lambda, \quad \mu_i = g^{\lambda_i}$$
$$\tilde{g}'_i = \tilde{g}^{\frac{1}{\kappa} r_i} \prod_{j=1}^N \tilde{g}_j^{\frac{1}{\kappa} A_{ji}}, \quad \tilde{g}' = \tilde{g}^\alpha \prod_{j=1}^N \tilde{g}_j^{\alpha_j}, \quad u' = g^\alpha \prod_{j=1}^N u_j^{\alpha_j}, \quad v' = y^\alpha \prod_{j=1}^N v_j^{\alpha_j}$$
$$\dot{\nu}_i = g^{\kappa \sum_{j=1}^N 3\alpha_j^2 A_{ji} + \rho r_i}, \quad \dot{\nu} = g^{\kappa^2 \sum_{j=1}^N 3\alpha_j^3 + \kappa\tau\lambda + \kappa\rho\alpha},$$
$$\dot{\omega}_i = g^{\sum_{j=1}^N 2\alpha_j A_{ji} + \sigma r_i}, \quad \dot{\omega} = g^{\kappa \sum_{j=1}^N \alpha_j^2 + \kappa\sigma\alpha}, \quad t_i = g^{\sum_{j=1}^N 3\alpha_j A_{ji} + \kappa\tau\lambda_i}$$

3. P computes a challenge:

$$(c_1, \ldots, c_N) = H\big(t^\kappa, \nu^\kappa, \omega^\kappa, \mu^\kappa, \{\mu_i^\kappa\}, \{\tilde{g}_i'^\kappa\}, \tilde{g}'^\kappa, u'^\kappa, v'^\kappa, \{t_i^\kappa\}, \{\dot{\nu}_i^\kappa\}, \dot{\nu}^\kappa, \{\dot{\omega}_i^\kappa\}, \dot{\omega}^\kappa, g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}\big) \ .$$

4. P computes $s = \kappa\alpha + \sum_{j=1}^N r_j c_j$, $s_i = \kappa\alpha_i + \sum_{j=1}^N A_{ij} c_j$, and $\lambda' = \kappa\lambda + \kappa \sum_{j=1}^N \lambda_j c_j^2$. The proof consists of the following tuple:

$$(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda') \ .$$

The proof of the modified prover corresponds to a proof where τ is replaced by $\kappa\tau$, ρ is replaced by $\kappa\rho$, σ is replaced by $\kappa\sigma$, and so on. It is therefore easy to see that the constructed proof is correct. Since κ generates $\mathbb{Z}_q$ the distribution of the new random elements are not altered by this modification.

Note that we may replace $\tilde{g}'_i = \tilde{g}^{\frac{1}{\kappa} r_i} \prod_{j=1}^N \tilde{g}_j^{\frac{1}{\kappa} A_{ji}}$ above by $\tilde{g}'_i = \tilde{h}^{r_i} \prod_{j=1}^N \tilde{h}_j^{A_{ji}}$, i.e. this requires no additional exponentiation or κ-multiplication. This is why we introduced the $\tilde{h}$ and $\tilde{h}_i$ elements. It is not necessary to verify that $\tilde{h}, \tilde{h}_i \in G_q$ explicitly. A more efficient method is choosing $h, h_i \in \mathbb{Z}_p^*$ randomly and defining $\tilde{h} = h^\kappa$ and $\tilde{h}_i = h_i^\kappa$.

Note that $\tilde{h}_i$ and $\tilde{g}_i$ may be reused for several proofs, i.e. in the mix-net of Section 5.1 each mix-server M_j computes the $\tilde{g}_i$ only once. This requires $N + 1$ κ-exponentiations.

Apart from this both the prover and the verifier computes $5N + 9$ additional κ-exponentiations, and the prover computes $2N + 6$ additional κ-multiplications.