Verifying a shuffle in zero-knowledge

Chris Patton

February 8, 2016

1 Progress report

I've completed the intro for the write up, outlined the remaining sections, and found all the references I'll need. I've worked through most of the protocol and am ready to proceed with implementation. So far, I've coded the preliminaries: ElGamal encryption and batch decryption and shuffling. It was necessary to code ElGamal from scratch, because the library available for Go rolls encoding and encryption into one function; in order to do Neff's proof, I need to be able to handle group elements. Code and test cases can be found at github.com/cjpatton/shuffle.

2 Introduction

A mixnet (sometimes called a "mix-network") is a service designed to facilitate anonymous communications over the internet [1]. Each of a set of users $\mathcal{Q} = \{P_1, \ldots, P_n\}$ wish to send a message to a user in $\mathcal{Q}$. Each encrypts their message M and the identity of the intended recipient P_i under the public key of the mixnet server (heereafter referred to as the "mix") and transmits the ciphertext. The mix server waits until it receives $m \leq n$ ciphertexts from m distinct users. It then decrypts each message and destination pair under its secret key and transmits each message to its intended destination. Crucially, the mix transmits the messages in a random order so as to hide the correspondence between senders and recipients. As long as the mix is trusted and the user's ciphertexts are faithfully transmitted to the mix, this simple approach suffices to preserve anonymity of its participants.

Normally we think of the mixnet being composed of a cascade of mix servers, each decrypting, shuffling, then transmitting to the next mix in the cascade until finally the last server outputs the users' messages. This is meant to distribute trust so that as long as the messages pass through at least one honest mix, anonymity is achieved. Still, the trust model here is rather strong. For example, the mix could send a fake message on behalf of one of the senders, or redirect it to a different recipient. We consider the problem of publically verifying that the outputs are indeed a permutation of the inputs. Moreover, we require that the correspondence not be revealed. Andrew Neff describes in [5] a method for proving a shuffle in zero-knowledge.

Neff's solution is based on the ElGamal public key encryption scheme, which is known to be semantically secure assuming the discrete log problem (decisional Diffie-Hellman, or DDH) is hard [2]. Based on this assumption, Neff proves his protocol achieves a property known as honest verifier zero-knowledge (HVZK) [4]. Informally speaking, this means that as long as the verifier faithfully executes the protocol, the mix can prove the outputs are a permutation of the inputs without revealing the permutation.

I formulate and define security for the problem and present Neff's protocol. For simplicity, I limit the discussion to a mixnet with a single mix server. In this scenario, users encrypt their messages under the public key of the mix. The mix waits until it receives a full batch of ciphertexts, decrypts them under its private key, and shuffles the output. In section 3, I specify ElGamal encryption, describe the DDH assumption, and sketch the HVZK notion. In section 4, I present the protocol and sketch its security. I implement the general k-shuffle as described in this section in the Go programming language.[1].

3 Preliminaries

Notation.

3.1 ElGamal encryption

3.2 Decisional Diffie-Hellman

3.3 Honest verifier zero-knowledge

4 The Neff shuffle

Describe what the mix does with specified syntax.

4.1 The basic protocol and its security

Not useful yet ...

4.2 Simple k-shuffle

Not useful yet ...

[1] Freely available at *github.com/cjpatton/shuffle*

4.3 General k-shuffle

This is what we actually implement.

References

[1] David Chaum, 1981. "Untraceable electronic mail, return addresses, and digital pseudonyms." `http://freehaven.net/anonbib/cache/chaum-mix.pdf`

[2] Taher Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms." Appeared in *CRYPTO'84.*

[3] D. Chaum and T.P. Pedersen. "Wallet databases with observers." Appeared in *CRYPTO'92.*

[4] Oded Goldreich, 2001. *Foundations of Cryptography.*

[5] C. Andrew Neff, 2001. "A verifiable secret shuffle and its application to e-voting." `http://freehaven.net/anonbib/cache/shuffle:ccs01.pdf`

[6] Masayuki Abe and Hideki Imai, 2003. "Flaws in some robust optimistic mix-nets." Appeared in *ACISP'03.*