Verifying a shuffle in zero-knowledge

Chris Patton

January 21, 2016

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.

However, 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 [2] a method for doing just that.

For this project, I will formulate the problem, present Neff's solution, and describe its security properties. I will also implement the shuffle in the Go programming language.

References

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

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