Cryptography Project

Securocracy

Patryk Majewski, Gabriel Wechta

Wroclaw University of Science and Technology
Faculty of Algorithmic Informatics (both)

June 13, 2022

Introduction

The goal of the project is to implement secure and anonymous, democratic voting via save channel (TLS). We use two-round anonymous open veto network protocol (OV-net). In this protocol participants execute the protocol by sending two-round public messages, what is more no trusted third parties are required. After the protocol is completed, because essentially it is run by the voters themselves, every participant can compute voting result by himself.

Contents

1. Problem Statement

Stated problem is following. Execute multi-party secure computations for *democratic* and *anonymous* voting. Where *democratic* means that every vote weights the same, and *anonymous* means that nobody, also observer, can tell who voted for what. All communication is public, there is no option for 'private' message exchange between participants, also no trusted third party may be involved. Each participant answers 'yes' or 'no' ($v_i \in \{0, 1\}$). After the voting everybody, also someone who did not actively participated in the protocol and only observed the communication, can tell how many participants voted 'yes' and how many voted 'no'. This information can be interpreted in any way, for example – if more than $\frac{2}{3}$ votes where 'yes' we will take LSD, otherwise methamphetamine.

In general decentralized electronic voting where the protocol is run by the voters themselves provides better voter privacy, and is thus most suitable for small-scale elections. The protocol should be designed in such a way that each participant's input remains secret. The best way to focus attention is to imagine problem as a social game where one selected person formulates a question that is sent out to participants. So for example protocol could be run in order to democratically decide on the issue "Is this problem $\mathcal{NP}$-complete?" or "Should we take drugs tonight?".

2. Open-Veto Network Protocol

Let:

- G denote a finite cyclic group of prime order q in which the Decisional Diffie–Hellman assumption holds
- g be a generator in G
- n denote number of participants
- P_i denote i-th participant
- (G, g) be agreed upon values, known to everybody.
- v_i denote P_i's vote. $v_i \in \{0, 1\}$, where v_i equals 1 if "yes" and 0 if "no"

Protocol looks in the following way. Each participant P_i selects a random secret: $x_i \in_R \mathbb{Z}_q$.

Round 1: Every participant P_i:

1. publishes g^{x_i}
2. publishes zero knowledge proof for knowing x_i
3. waits for others to complete previous steps
4. checks validity of ZKPs of other participants
5. computes

$$g^{y_i} = \frac{\prod_{j=1}^{i-1} g^{x_j}}{\prod_{j=i+1}^{n} g^{x_j}}$$

At this point every participant computed his own public ephemeral value g^{y_i}.

Round 2: Every participant P_i:

1. publishes $g^{x_i y_i} g^{v_i}$
2. publishes ZKP showing that v_i is one of $\{0, 1\}$

Counting votes: After all participants finished, every participant can count votes by computing

$$g^{\gamma} = \prod_{i=1}^{n} g^{x_i y_i} g^{v_i}$$

where γ is number of "yes" votes. Since γ is not bigger than n and n is a "small" number, γ can be guessed by exhaustive search.

3. Cryptography Construction

We base cryptography construction of our application on the work of Hao, Ryan, and Zieliński [2]. Protocol presented in this paper and also in original paper about AV-nets by Hao and Zieliński [3] assumes authenticated public channel and does not assume message encryption, so messages can be transferred publicly.

We will achieve requirements and goals of the protocol by implementing following constructions.

- In order to provide secure communication between parties, we will use TLS protocol with self-signed certificates automatically handled by the application.
- All protocol's operations will be computed in the finite cyclic group G in which the decisional Diffie–Hellman problem is intractable.
- For ZKP in **Round 1** we will use Schnorr's Signature to prove knowledge of the exponent.

- For ZKP in **Round 2** we will use technique proposed by Cramer, Damgård and Schoenmakers (CDS, for details see [1]) to demonstrate that the vote is either 0 or 1, without revealing which one.

4. Proposed Solution

We will implement client-server application that will allow Users to participate in anonymous voting accordingly to OV-net description above. One randomly and publicly chosen User will be prompted to ask one question that later will be transferred to all the other Users and they will vote secretly. Application will have terminal based UI. Application will be strongly driven by the spirit of PoC.

Application architecture: Every client (same as participant) will connect to the server. Although no third party is required in stated problem in order to make communication easier, proxy server will be implemented that will transfer messages between clients and choose one lucky User.

Libraries and dependencies: Application will be written in `Python 3` language:

- `websockets` - library for building WebSocket protocol servers and clients.
- `pycryptodome` - library for performing fast operations on elliptic curves points.
- `docker` - for running simulations of tallies.
- `openssl` - for deriving certificates for TLS.

5. Security Analysis

In voting problem always exists threat of collusion. In order to stay sane, we limit analysis to partial collusion. It is also important to notice that, by nature, voting cannot preserve the full voter's privacy under that circumstance; attackers simply need to subtract their own votes from the final tally and will gain some knowledge about others votes.

Maximum ballot secrecy: Each cast ballot is a ciphertext that is indistinguishable from random, and hence does not reveal anything about the voter's choice. In the protocol, in the first round each voter sends an ephemeral public key g^{x_i}, in the second each voter sends an encrypted ballot $g^{x_i y_i} g^{v_i}$

The value of y_i is determined by the private keys of all participants except P_i so even in a partial collusion against P_i, y_i stays secret random value.

Furthermore because of Decisional Diffie–Hellman assumption, attacker is not able to distinguish the ballot $g^{x_i y_i} g^{v_i}$ from a random element from G.

Voters will: The romantic, but not wishful, assumption of this protocol is that every participant wants to finish current voting. It is important to notice that if some voters refuse to send data in round 2, the tallying process will fail. But this attack, on the other hand will show without doubt who are the attackers/bad voters. Then bad apples may be tossed away and a new voting according to the protocol may be carried away.

Denial of Service Attack: Due to the lack of time and resources to implement appropriate countermeasure in the application threat of being flooded with requests and DoS messages has to be accepted. In future versions of the application mitigation will be introduced.

References

[1] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *Advances in Cryptology — CRYPTO '94*. Ed. by Yvo G. Desmedt. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 174–187. ISBN: 978-3-540-48658-9.

[2] Feng Hao, Peter Ryan, and Piotr Zielinski. "Anonymous voting by two-round public discussion". In: *Information Security, IET* 4 (July 2010), pp. 62–67. DOI: 10.1049/iet-ifs.2008.0127.

[3] Feng Hao and Piotr Zielinski. "A 2-Round Anonymous Veto Protocol". In: Mar. 2006, pp. 202–211. ISBN: 978-3-642-04903-3. DOI: 10.1007/978-3-642-04904-0_29.