Stellot: Secure internet voting using distributed networks

Stanislaw Baranski

December 2022

There is no doubt, that traditional paper-based voting still lags behind government digitalization. Analysis of this area quickly reveals several unsolved issues.

Voting is one of the most popular mechanisms for collective decision-making, prevalent in all types of communities; from housing associations and board members, through domestic presidential elections, to all forms of global voting that take place on the internet.

The act of voting can occur in different ways including paper-based precinct voting, mail-in ballots, electronically using DRE voting machine and internet voting.

Of all of them, internet voting is the most conventional, cheapest, fastest, and safest (e.g., during the outbreak of COVID-19), and hence, a desirable method for conducting voting.

Internet voting can increase turnout and the frequency of votings, but most importantly, it can catalyze the further development of modern democracy. Enabling practical applications of direct democracy, liquid democracy, and all other sorts of voting methods like Approval voting, Alternative vote, Score voting, and many others [1].

Visions of smart cities, crypto cities [2], Decentralised Autonomous Organisations [3], and other forms of algorithmic governance [4] rely on the existence of internet voting, so there is a high demand for such systems.

Online voting is often compared to online banking. Admittedly, online voting requires much higher security than online banking. Many researchers and experts in the field doubt the possibility of conducting public voting over the internet [5]–[10]. The resistance lies—among others—in insufficient confidence in the technology and a need for trust in the authorities controlling the voting process.

"Blockchain voting is overrated among uninformed people but underrated among informed people," says Vitalik Buterin, the co-founder of Ethereum, suggesting more optimism in academia about internet voting [11].

Vitalik Buterin proposed the concept of MACI which is a base layer for coercion-resistant, secure, and private internet voting [12].

The system struggles with the assumption about the existence of a trusted-third party coordinator and is based on the Ethereum blockchain, which—due to high transaction costs and low throughput—may not be acceptable in practical uses.

Overview

The project aims to solve the problem of internet voting by employing blockchain [13], MPC [14], co-zkSNARK [15], and an ad-hoc peer-to-peer network consisting of voters' smartphones and laptops. Resulting in an internet voting protocol that achieves security, integrity, coercion-resistance, and privacy, without any trusted third party.

Value proposition

1. Convenience, and safety. No need to leave your home to participate in voting.
2. Cheap. No need to print ballot papers or hire people to coordinate the voting process.
3. Trustless, secure, transparent. Users don't need to trust the authorities that their votes have been included and that the counting process has been correct.
4. Increased turnouts and the frequency of votings.
5. Enabling direct democracy, liquid democracy, and all other sorts of voting methods like Approval voting, Alternative vote, Score voting, and many others [1]

Economic modeling

The solution is going to be open-sourced and published in a peer-reviewed journal. However, in the future, we plan to monetise it in a pay-per-voting or software licensing model.

Technical vision

We will implement the proposed protocol by incrementally enhancing the current one [13]:

1. First, we will develop an MPC protocol to compute the encryption key. This is achieved using Shamir Secret Sharing (SSS) or Distributed Key Generation (DKG) schemes [16], [17]. The MPC-generated encryption key will replace the current server-generated encryption key.

2. Then, we will replace the current storage mechanism based on Stellar transactions, with the new Stellar's Soroban smart contract platform [1].
3. Then, we will implement the MPC protocol for decryption and tally the votes stored in Soroban smart contract. This will remove the need for trusted-third-party used in the current protocol [13].
4. Then, we will develop the zkSNARK protocol for generating proof of correctness of both MPC functionalities. The generated proof of correctness will be published to smart contracts along with the calculated results.

Technical overview

Voters form a peer-to-peer network using their smartphones. Let's assume a voter consists of a smartphone and a key pair (see Figure 1).

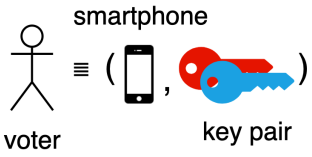

Figure 1: Notation.

The first step that the network of voters' smartphones has to do is to form a peer-to-peer (or rather, smartphone-to-smartphone) network [18], [19] ^86890b

A distributed network consisting of eligible voters runs a blockchain and MPC software. Consequently, two functionally separate networks (blockchain and MPC) are running on voters' devices. Blockchain network act as bulletin boards, where all transactions are collected and accessible to anyone. MPC network offers two functionalities: 1) jointly generating encryption key; 2) jointly decrypting and tallying votes, along with producing a zk-SNARK proof. The networks are communicating with each other for fetching votes and published results. A big-picture idea of the proposed system is presented in Figure 2.

The voting process consists of three phases:

1. bootstrapping p2p network phase presented in section ;
2. distributed encryption key generation phase presented in section and figure 3;
3. casting votes phase presented in section and figure 4;
4. tally phase presented in section and figure 5.

[1] https://soroban.stellar.org

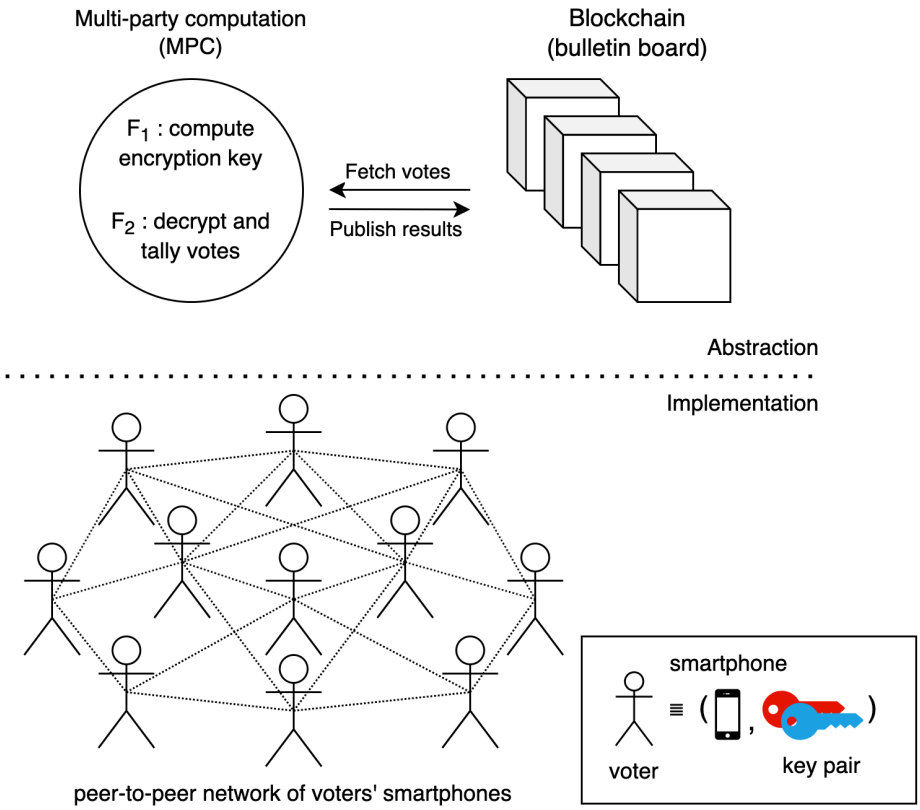

Figure 2: Overview of the proposed system

Bootstraping p2p network

The first step that the network of voters' smartphones has to do is to form a peer-to-peer (or rather, smartphone-to-smartphone) network [18], [19]. The network should be closed-membership, meaning that only eligible participants are allowed to join. This is achieved by publishing a predefined list of public keys $R = \{pk_1, ..., pk_n\}$ corresponding to the eligible voters, who holds secret keys sk, such that, $pk \equiv g * sk$ for some known generator g. We use SNARK-friendly EdDSA public-key cryptosystem.

Depending on the scale and type of voting, such a list can be created by organizers, or locally by the voters themselves. In order to prevent selling keypairs to briber, the keypairs should be associated with some stake: proof of citizenship of a country, high reputation in a local community, proof of humanity, or some amount of cryptocurrency, in such a way that the value of a stake is worth more than a vote from such address.

To establish a peer-to-peer (p2p) network over the internet it is required to solve many network-related problems. Namely,

- Most of smartphones connected to the Internet are behind the NAT or some kind of firewall. Such devices work in asymmetric access policy, in which they can establish connections to other devices, but others can not establish connections to them. In settings where all devices work in asymmetric access policly it is impossible to start any connection [20]. To solve the issue we can use techniques like Traversal Using Relays around NAT (TURN), Circuit Relays, Rendezvous servers [21], [22], Hole Punching, Session Traversal Utilities for NAT (STUN) [23], Interactive Connectivity Establishment (ICE)[24], and WebRTC [25].
- In a p2p network the number of connections between peers grows quadratically, i.e., n^2, where n is the number of peers in the network. Moreover, there may be multiple connections between peers for each used protocol. To reduce the number of connections, we can use multiplexing like QUIC, Yamux, or Mplex [26]), which allows re-using established connections for several protocols.
- The connections must be authenticated and encrypted. To solve the issue, we can use techniques like TLS [27] or Noise [28].
- Peers must be able to discover each other. To do this, depending on network conditions, we can use Bootstrap peer list, Multicast DNS (mDNS), or Rendezvous servers.

To solve those issues we piggyback on the libp2p [29], an open source library, which addresses all of the mentioned issues.

Distributed Encryption Key Generation

Once the network is established nodes execute the first MPC functionality F_1, which lets nodes jointly compute $\text{PK} \equiv \text{SK} * g$, without reconstructing SK on any

single device. This is achieved using Shamir Secret Sharing (SSS) or Distributed Key Generation (DKG) schemes [16], [17].

Symbolically, the functionality F_1 is defined as follows:

$$F_1(\mathrm{SK}_1, ..., \mathrm{SK}_N) = DerivePubKey(DKG(\mathrm{SK}_1, ..., \mathrm{SK}_N)) \rightarrow PK$$

The reconstruction of a SK is possible only if a sufficient number of votes (predefined threshold) collude. It would require executing malicious functionality $F_m(\mathrm{SK}_1, ..., \mathrm{SK}_N) = DerivePrivKey(\mathrm{SK}_1, ..., \mathrm{SK}_N) \rightarrow \mathrm{SK}$. The situation should not happen under the honest majority assumption.

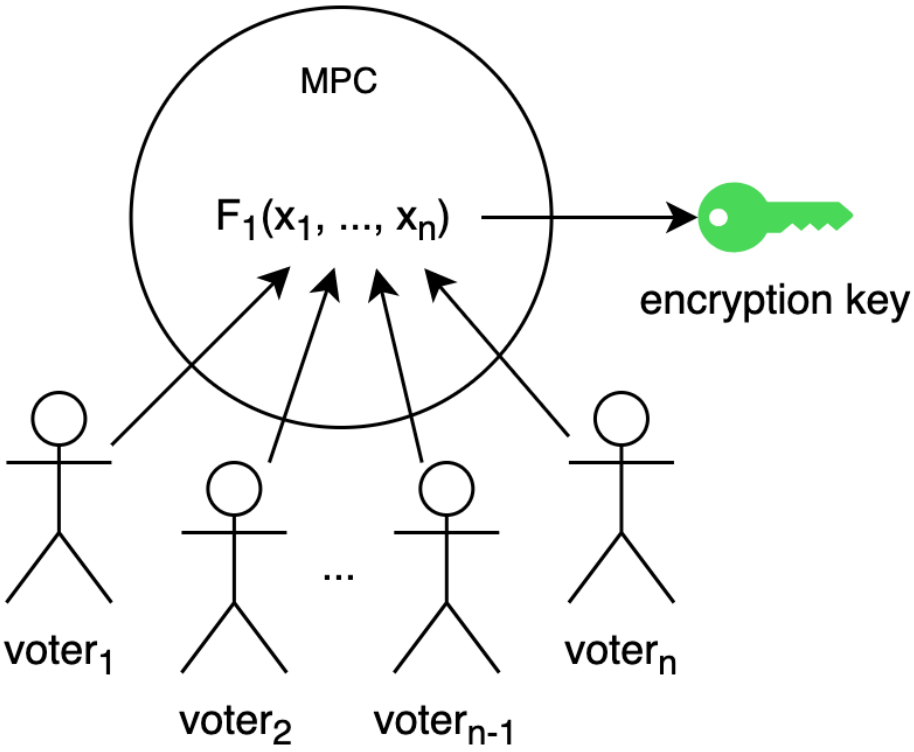

Figure 3: Setup phase of the proposed i-voting system.

Voting phase

Following MACI [30] protocol. Initially, at time T_{start}, nodes begins with an internal state $S = \{i : (key = pk_i, vote = \varnothing)\}$ for $i \in 1..n$. Let us assume $\mathbb{O}$ to be a set of all vote options.

Between T_{start} and T_{end}, each eligable voter $pk \in R$ is allowed to send a command *cmd* that change his vote and/or key state:

1) **not change vote, not change key**, $cmd := (i, key := S_i[key], vote := S_i[vote])$, where $S_i[key]$ is the current public key, and $S_i[vote]$ is the current vote option associated with the voter of index i.
2) **change vote, not change key**, $cmd := (i, key := S_i[key], vote :=$ newVote), where $S_i[key]$ is the current public key associated with the voter of index i, and newVote $\in \mathbb{O}$ is a new vote option.
3) **not change vote, change key**, $cmd := (i, key :=$ newPk$, vote := S_i[vote])$, where newPk is newly derived public key, and $S_i[vote]$ is the current vote option associated with the voter of index i.
4) **change vote, change key**, $cmd := (i, key := newPk, vote := newVote)$, where newPk is newly derived public key, and newVote $\in \mathbb{O}$ is a new vote option for the voter of index i.

Each command *cmd*, before broadcasting to the network, must be 1) signed and 2) encrypted.

The digital signature proves both, the integrity of the content of the transaction, and the (zero-knowledge) access to secret key k. In our case, signatures also guarantee the cohersion-resistance property. Once the public key is changed, all commands signed with the previously set public key do not pass the signature verification, therefore are invalidated.

Encryption prevents associating voters with their commands, such that there is no way to prove that the key has been changed or not. Commands are encrypted using asymmetrically derived encryption/decryption key via Elliptic Curve Diffie-Hellman (ECDH). Concretely, the voter encrypts the transaction using a shared key derived from the voting's public (encryption) key PK and his secret key sk_i using ECDH. The decryption will be possible using different pairs of keys, i.e. voting's secret key SK and the voter's public key pk_i. This is possible thanks to ECDH scheme, which relies on the equation

$$sk_i * PK = sk_i * SK * g = SK * sk_i * g = SK * pk_i.$$

Since the encryption key is different for each command, only the sender and the recipient (inside the MPC) can decrypt it.

Moreover, each encryption should append a random value, called salt, to make two ciphertexts of is same votings indistinguishable.

The voting process is illustrated in Figure 4.

Tally phase

Once the election period has finished, voters agree on a common state of the recorded transactions on the blockchain.

Then, using the second functionality F_2, voters jointly perform decentralized decryption and tallying of the ballots without reconstructing the decryption key on a single device.

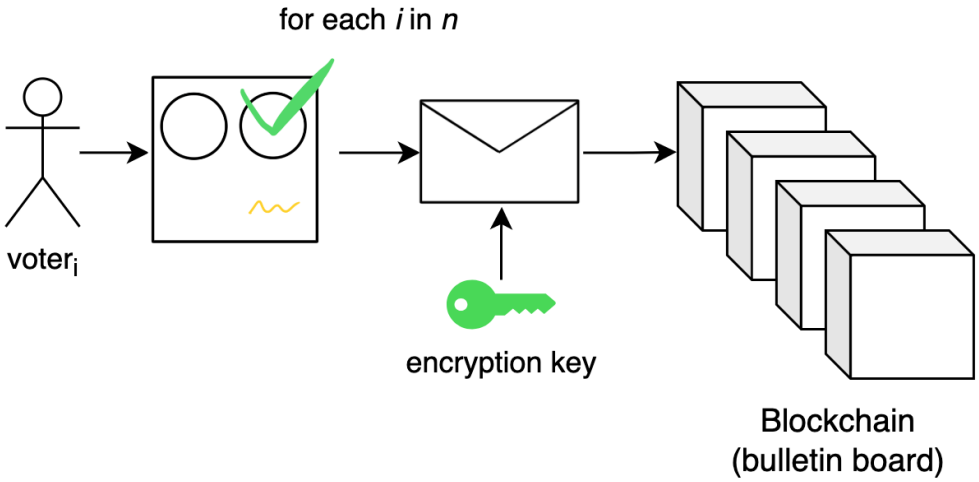

Figure 4: Voting phase, voters casting votes to the blockchain

Internally, F_2 validates all the recorded transactions and modifies their internal state accordingly to the content and the type of the transaction. Process each transaction as follows:

Processing a message $msg = (enc_{ek}(cmd, sig), epk)$, where cmd is a command, sig is a signature, epk is an ephemeral public key allowing for deriving encryption key via ECDH.

---

procedure $\mathrm{F}_2(SK, \mathrm{msgs}) \to r$:

each msg in msgs:

$ek \leftarrow ECDH(SK, epk)$

$(cmd, sig) \leftarrow enc_{ek}(\mathrm{msg})$

$(i, vote, key) \leftarrow cmd$

ensure $\mathrm{verify}_{S_i[key]}(sig, hash(cmd)) = True$

ensure $vote \in \mathbb{O}$

$S_i[key] \leftarrow key$

$S_i[vote] \leftarrow vote$

each o in O:

$r_o := |\{S_i[vote] \in S : S_i[vote] = o\}|$

output r

Vector r contains the total number of votes for each option $o \in O$.

Since the tallying and verification of messages' signatures is done after the voting period it's impossible to prove that the transaction is valid or invalid within the voting period — hence the cohersion-resistance property of the protocol.

Proving the correctness of multi-party computation for distributed secrets is a topic under active research. A naive solution would be to embed zk-SNARK prover into MPC; however, Stanford researchers recently proposed a method called "Collaborative zk-SNARKs" allowing for a much more efficient approach [15].

Finally, the key shares S_i should be destroyed from all voters' devices. As long as the majority of the nodes follow the procedure, the decryption key can not be reconstructed, that is, the malicious functionality F_m can not be performed.

The results and certificate are published on the Blockchain (bulletin board). See Figure 5.

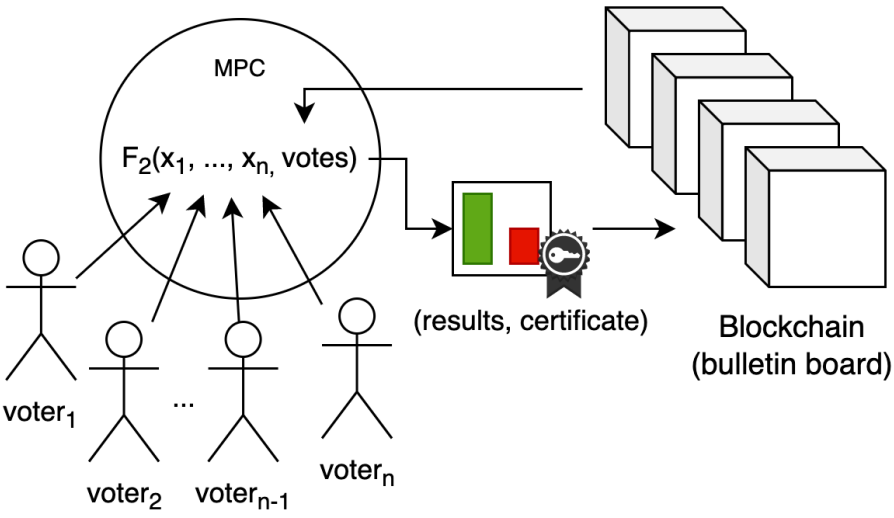

Figure 5: Tally phase of the proposed i-voting system.

Verification

After the results and the corresponding certificate has been published on the blockchain. Anyone—not only the voters taking part in the voting—can verify the correctness of the results using zk-SNARK verifier.

Open questions

- How to design a protocol resilient to nodes' unavailability?
- How much RAM and disk space does the proposed protocol use?
- How long does it take to execute both functionalities on a mid-range smartphone?
- How to prevent DDoS attacks?
- How many votes per second can handle the proposed protocol?
- Which MPC framework to use (e.g., obliv-c, EMP-toolkit, ABY, PyPC or others [31])?
- Which circuit compiler to use (e.g. libsnark, bellman, ZoKrates, Snarky, Circom, or others)
- Which proving system to use (libsnark, bellman, dalek bulletproofs, snarkjs, or others)
- Should we use a public Uniform reference string (URS) or a private Structured reference string (SRS)?
- What are the trust assumptions implied by each setup?
- Can we reuse the already established trusted setup? (e.g. from Ethereum, ZCash)

References

[1] J.-F. Laslier, "And the loser is... Plurality Voting," Jul-2011 [Online]. Available: https://hal.archives-ouvertes.fr/hal-00609810. [Accessed: 17-Sep-2022]

[2] V. Buterin, "Crypto Cities," 2021. [Online]. Available: https://vitalik.ca/general/2021/10/31/cities.html. [Accessed: 22-Aug-2022]

[3] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized Autonomous Organizations: Concept, Model, and Applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, Oct. 2019, doi: 10.1109/TCSS.2019.2938190.

[4] "Government by algorithm," *Wikipedia*. 18-Aug-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Government_by_algorithm&oldid=1105084102. [Accessed: 22-Aug-2022]

[5] S. Park, M. Specter, N. Narula, and R. L. Rivest, "Going from bad to worse: From internet voting to blockchain voting," *Journal of Cybersecurity*, vol. 7, no. 1, p. tyaa025, 2021.

[6] L. Mearian, "Why blockchain-based voting could threaten democracy," 12-Aug-2019. [Online]. Available: https://www.computerworld.com/article/3430697/why-blockchain-could-be-a-threat-to-democracy.html. [Accessed: 30-Aug-2022]

[7] S. Shankland, "No, blockchain isn't the answer to our voting system woes," 2018. [Online]. Available: https://www.cnet.com/news/privacy/blockchain-isnt-answer-to-voting-system-woes/. [Accessed: 30-Aug-2022]

[8] T. B. Lee, "Blockchain-based elections would be a disaster for democracy," 06-Nov-2018. [Online]. Available: https://arstechnica.com/tech-policy/2018/11/blockchain-based-elections-would-be-a-disaster-for-democracy/. [Accessed: 30-Aug-2022]

[9] B. Schneier, "On Blockchain Voting," 2020. [Online]. Available: https://www.schneier.com/blog/archives/2020/11/on-blockchain-voting.html. [Accessed: 30-Aug-2022]

[10] B. Schneier, "Blockchain and Trust," 2019. [Online]. Available: https://www.schneier.com/blog/archives/2019/02/blockchain_and_.html. [Accessed: 30-Aug-2022]

[11] V. Buterin, "Blockchain voting is overrated among uninformed people but underrated among informed people," 2021. [Online]. Available: https://vitalik.ca/general/2021/05/25/voting2.html. [Accessed: 22-Aug-2022]

[12] V. Buterin, "Minimal anti-collusion infrastructure - Applications," 04-May-2019. [Online]. Available: https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413. [Accessed: 30-Aug-2022]

[13] S. Barański, J. Szymański, A. Sobecki, D. Gil, and H. Mora, "Practical I-Voting on Stellar Blockchain," *Applied Sciences*, vol. 10, no. 21, p. 7606, Oct. 2020, doi: 10.3390/app10217606. [Online]. Available: https://www.mdpi.com/2076-3417/10/21/7606. [Accessed: 19-Jul-2022]

[14] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2–3, pp. 70–246, 2018.

[15] A. Ozdemir and D. Boneh, "Experimenting with Collaborative {zk-SNARKs}: {Zero-Knowledge} Proofs for Distributed Secrets," presented at the 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 4291–4308 [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/ozdemir. [Accessed: 23-Dec-2022]

[16] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2007.

[17] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," presented at the International Workshop on Public Key Cryptography, 2003, pp. 31–46.

[18] T. Zhuang, P. Baskett, and Y. Shang, "Managing ad hoc networks of smartphones," *International Journal of Information and Education Technology*, vol. 3, no. 5, p. 540, 2013.

[19] G. Aloi, M. Di Felice, V. Loscrì, P. Pace, and G. Ruggeri, "Spontaneous smartphone networks as a user-centric solution for the future internet," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 26–33, Dec. 2014, doi: 10.1109/MCOM.2014.6979948.

[20] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators." in *USENIX Annual Technical Conference, General Track*, 2005, pp. 179–192.

[21] T. Reddy, A. Johnston, P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," RFC Editor, RFC8656, Feb. 2020 [Online]. Available: https://www.rfc-editor.org/info/rfc8656. [Accessed: 04-Oct-2022]

[22] libp2p, "Circuit Relay." [Online]. Available: https://docs.libp2p.io/concepts/circuit-relay/. [Accessed: 04-Oct-2022]

[23] M. Petit-Huguenin, G. Salgueiro, J. Rosenberg, D. Wing, R. Mahy, and P. Matthews, "Session Traversal Utilities for NAT (STUN)," RFC Editor, RFC8489, Feb. 2020 [Online]. Available: https://www.rfc-editor.org/info/rfc8489. [Accessed: 04-Oct-2022]

[24] A. Keranen, C. Holmberg, and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal," RFC Editor, RFC8445, Jul. 2018 [Online]. Available: https://www.rfc-editor.org/info/rfc8445. [Accessed: 04-Oct-2022]

[25] J. Uberti and G. Shieh, "WebRTC IP Address Handling Requirements," Internet Engineering Task Force, Internet Draft draft-ietf-rtcweb-ip-handling-09 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rtcweb-ip-handling-09. [Accessed: 04-Oct-2022]

[26] libp2p, "Stream Multiplexing." [Online]. Available: https://docs.libp2p.io/concepts/stream-multiplexing/. [Accessed: 04-Oct-2022]

[27] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC Editor, RFC8446, Aug. 2018 [Online]. Available: https://www.rfc-editor.org/info/rfc8446. [Accessed: 04-Oct-2022]

[28] T. Perrin, "The Noise protocol framework," *PowerPoint Presentation*, 2018.

[29] "Libp2p." [Online]. Available: https://libp2p.io/. [Accessed: 04-Oct-2022]

[30] Ethereum Foundation, *Minimal Anti-Collusion Infrastructure*. Privacy & Scaling Explorations (formerly known as appliedzkp), 2022 [Online]. Available: https://github.com/privacy-scaling-explorations/maci/blob/9b1b1a631090ee89d2bc12f4bcef7763e42caef0/specs/01_introduction.md. [Accessed: 02-Sep-2022]

[31] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, "Sok: General purpose compilers for secure multi-party computation," presented at the 2019 IEEE symposium on security and privacy (SP), 2019, pp. 1220–1237.