Cryptography 2020 Project - Helios Voting

Bartosz Drzazga, Mateusz Jachniak

20.06.2020

Contents

1 Introduction

Voting with cryptography control sometimes called voting based on an open audit remained largely theoretical endeavor to replace standards methods. Despite dozens of fascinating protocols and recent breakthroughs in this field, there are only a few specialized implementations that few people have experienced directly. Nowadays election results can be completely verified by public observers, because ballots are physically stored. These Protocols are said to provide two properties: casting votes assurance in which each voter receives personal assurance that their vote was properly captured and universal verifiability, in which each observer can verify that all votes were appropriately collected. Some have benefited from the term open-based elections to indicate that even a public observer with no special role in the elections, may act as an auditor. Unfortunately, there is a considerable public awareness break: few understand that these techniques represent a major improvement in election control. Several implementations that exist, so far have not had big impact, as expected, largely because they require special equipment and personal experience, thus limiting their reach. [1]

Helios[1] is a voting system designed to enable practical open audits, verifiable elections with the sole support of a web browser. Helios has been used by several hundred thousand voters from various institutions and many private companies. Choosing the options to create, run, and compare elections with the sole support of a web browser has several implications. The first is to provide comprehensive, verifiable elections at your fingertips from anyone who has access to the Internet. Using Helios does not require the use of any dedicated hardware, installation of any specific software, or having a physical email address. This also applies to the entire electoral audit process: all control operations can be performed from a scratch laptop. Each voter can check if his vote is included, unchanged, in the counter, and anyone can check if the counter is correct. These control operations do not require any privileged access to certain data or infrastructure: they only require manipulation of public data. Besides, unlike traditional paper-based voting, control activities do not require constant monitoring: no supply chain needs to be maintained for a meaningful audit. As a result, independent sum checks can be carried out by anyone at any time after the end of the election, which would be impossible with traditional paper elections. Helios guarantees the confidentiality of votes using distributed encryption: the votes are encrypted directly on the voter's computer, and only then are sent to the Helios server, which never sees the decryption key unless the organizers of the election decide otherwise. The decryption keys are independently generated by the trustees and are never combined: no partial decryption key should ever leave the trustee's computer. As a result, it would be necessary to corrupt all trustees in the election to carry out illegal decryption. Despite this, Helios inherits important restrictions that seem inherent to clean Internet elections - somewhat softening them.

Helios provides voters with very limited security guarantees who rely on a compromised computer to send votes. If the malware controls the voter's computer and if this computer is the only interface that the voter uses for the election, the computer can display everything voters want to see, doing something completely different in the background (e.g. vote for another candidate or reveal the vote to a third party). On the contrary for systems that cannot be verified, Helios offers several audit options that allow voters to detect changes in votes that would have been made by a malicious computer, provided that the voter can access an honest computer at some point, a computer that doesn't even need to be connected to the internet. These observations also apply to the Helios server, which in the event of damage would have to leave evidence of any

[1]Documentation of Helios Voting System https://documentation.heliosvoting.org/home.

voting changes that it would have made, making it possible to detect if a fair device could be used. Moreover, Helios does very little to protect voters from coercion - although some forms of resistance to coercion can be obtained by external means, e.g. forcing voters to use Helios in the comfort of the booth to vote and in the absence of cameras. But if the election organizers allow all interaction between the voter and the system to take place in an unattended context (no voting booth, no registration in person), the coordinator could effectively dictate his voter behavior from start to finish and check that the voter follows his instructions. Even so, even in this case, Helios offers a limited form of protection: voters can submit as many votes as they want, and only the last one is included. This feature, in addition to the huge advantages in dealing with voters using an unreliable internet connection or uncomfortable with the browser interface, allows voters who at a certain moment would feel compelled to vote in an undesirable way to cast a different vote at a later date in a secure context.

2 Process of election

2.1 Voting in a Helios election

The first contact of a voter with a Helios election usually happens through an email invitation to vote. This email contains a description of the election, a link to the voting booth, and a voter ID and password that need to be used in order to submit a ballot. An election fingerprint, provided as a tracking number, is also provided and identifies the election in a unique way. Using the link to the voting booth, voters can submit a ballot to the election server. Helios has an open API[2], which means that anyone could program a ballot preparation system (BPS) that can be used to submit ballots in any election. In result, there has been created a lot of frameworks [2] which can be analysed of security cases. However, most voters actually use the BPS provided by the Helios website, even if using a different BPS might have some advantages in terms of the trust - a voter might feel more confident that his vote will be properly prepared if he uses a ballot preparation system provided by a candidate he supports.

Any voter can repeat the whole ballot preparation and submission procedure any number of times: only the last received ballot will be taken into account, and voters can check this thanks to their ballot tracker.

2.2 Election management

During the creation of new elections, the administrator of the elections first determines the election's name, as well as some general features: whether the elections are intended to run with a closed or open voter list, regardless of whether voter aliases are to be used and what contact address should be offered when voting in favor.

When questions and voters were defined and trustees have returned their public key, elections can be suspended. From this moment, no changes can be made to any of the election parameters, and the electoral fingerprint is calculated as the abbreviation for all electoral parameters. This electoral fingerprint should be broadcast by different channels: depending on the elections, it was printed in the institutional press, displayed on electoral pages institution and/or included in the invitation to vote. When the election is frozen, voters may be invited to vote and Helios offers a shipping mechanism that includes various templates and supports distribution of electoral credentials and optional aliases. Voters can then submit their ballot papers. Every time a card is received the Helios server checks its validity and stores it for inclusion in the summary. When the voter submits more than one ballot paper, only the last one is used in counting votes, while others are archived.

After the voting time has elapsed, the Helios server calculates the encrypted election counter, adding up the last valid vote received from each voter, using the homomorphic property of the encryption scheme used for protecting the votes. It is a public operation that anyone can execute just as easily as the Helios server. Then the trustees are asked to decrypt this sum. By connecting with Helios' trustee interface via browser, trustees can download encrypted tally, see it is a fingerprint, load their private browser key (those will never leave the browser), decrypt the summary partially, and upload the result of this decryption to the Helios server. As with voting preparation, there is no need to use the Helios web interface for this purpose: other software managed independently, can perform the same operations and send partial decryption of the sum. No information can be obtained until one of the trustees sends partial decryption of the sum. But as all confidants did their duty,

[2]GitHub repository, which contains APIs used in Helios Voting https://github.com/benadida/helios-booth.

Helios server combines partial decryption in the full electoral register and provides this electoral register. This combination again is a public operation. After the calculation, all information needed for verification elections becomes available from the Helios server. In addition, secret keys held by trustees may be destroyed: they are not auditable, and this destruction reduces the risk of a key compromisation in the future.

2.3 Election audit

There are three types of verification:

1. Cast-as-intended verification - enables any voter to obtain the assurance that the ballot he submits captures his vote intent.
2. Recorded-as-cast verification - enables any voter to obtain the assurance that his ballot has been properly recorded on the Helios server.
3. Tallied-as-recorded verification - enables anyone to verify that all the valid recorded votes are included in the tally.

Together, these verification steps provide what is often called end-to-end verifiability.

3 Cryptography in Helios

3.1 Arithmetic and computational assumption

The protocols implemented in Helios make use of a multiplicative cyclic group G of prime order q, in which the Decisional Diffie-Hellman (DDH)[3] problem is believed to be hard [3]. This means that, given a generator g of G and a triple g^a, g^b, g^c where a and b are chosen at random in Z_q, it is believed to be hard to decide whether c has also been chosen at random in Z_q, just as a and b, or whether c = ab.

3.2 Encryption

For encryption ElGamal has been used, which security relies on the DDH problem. This encryption scheme guarantees indistinguishably of ciphertexts if the DDH problem is hard in G: anyone who would be able to derive any single bit of information about the plaintext corresponding to a given ciphertext would also be able to solve the DDH problem in G. This encryption scheme is used in Helios to protect the votes, and indistinguishably of ciphertext implies, in particular, that no one will be able to even recognize if two ciphertexts encrypt the same vote or not.

3.3 Zero-Knowledge proofs

The verifiability of the election and confidentiality of the votes heavily rely on the use of zero-knowledge proofs, used to prove three types of statements:

1. Trustees are required to prove that they know the private key that matches the public key they are publishing.
2. Trustees are required to prove that they honestly contribute to the tally of the elections.
3. Voters are required to prove the validity of the ballot they submit.

Helios makes use of sigma protocols to prove all these statements, The first context in which Helios requires proofs is during key generation by the trustees. If all trustees collude or have their secret key stolen, then no privacy is guaranteed. We however want to make sure that, as long as one trustee behaves honestly, the privacy of the votes is protected. Honest key generation is crucial for privacy. It does not guarantee, however, that the decryption of the tally is performed correctly. A single trustee could indeed try to manipulate the outcome of an election by cheating when computing his decryption factor. A Helios ballot contains a series of questions, each with a number of possible answers [4]. For each question, the voter is allowed to pick a number of answers, defined by the election rules: this can be just one answer, any number of answers (for approval voting), or a number within a fixed range.

3.4 Protocol analysis

Helios is expected to offer end-to-end verifiability. The privacy of the voters relies on the honesty of at least one trustee. Moreover, the security of the voting client is also crucial for privacy: a malware recording

[3]The decisional Diffie–Hellman (DDH) assumption https://en.wikipedia.org/wiki/Decisional_Diffie-Hellman_assumption.

all actions of the voters would easily violate privacy. Internally, a malicious BPS could transmit votes in clear, in parallel with the normal ballot preparation. Coercion resistance is only offered in a very weak form, as usual for unsupervised voting systems: voters have the possibility to re-vote if they felt coerced to vote at some moment, but the coercer will be able to observe on the ballot tracking center that a new ballot has been submitted.

4 Our command-line implementation of a client

The Helios voting server is written in `Python 2`. The web page that offers election management and voting booth uses `Django` library. The original ballot preparation system (BPS), which is a part of the voting booth, is written in `JavaScript`.

The command-line voting client is basically a voting booth outside a web browser. The minimal Helios voting client features:

- downloading election data from server
- collecting elections of a user in an interactive session
- sealing a ballot (BPS)
- casting an encrypted vote

Our implementation of the client is written in `Python 3`. We chose this programming language because the server itself is written in `Python`. The same underlying technology allows us to easily use the same or similar data types, libraries and tools.

The alpha version fully supports connection with a server and downloading election details. The client partially supports collecting a user's elections and encrypting answers. Approval voting questions need more work. The program is able to output computations results in `JSON` format that is in part compatible with Helios server (key order is correct, but so far we output values as they are represented internally, i.e. integers are output as integers, while Helios uses string representation of all large integers). The command-line implementation does not support user authentication and vote casting yet.

For security reasons, the client accepts only `TLS` connections, where server's certificate is verified. The command-line client itself does not use client certificate because the scheme does not need client authentication with a certificate. The Helios server does not expect a certificate from the client. Users authenticate by signing in with their credentials or their login in external services like Google and Facebook. Election administrator can control eligible voters list by providing e-mail addresses or by making elections public. Helios uses end-to-end encryption so that, even after SSL decryption, the vote is still encrypted with the election key and is considered extremely private.

5 Conclusion

Helios is the first publicly available implementation of a web-based open-audit voting system. It fills a niche, that elections for small clubs, online communities, and student governments need trustworthy elections without the significant overhead of coercion-freeness. That means, that Helios can be a useful educational resource for open-audit voting by providing a valuable service - outsourced, verifiable online elections – that could not be achieved without the paradigm-shifting contributions of cryptographic verifiability.

References

[1] Ben Adida. Helios: Web-based open-audit voting. Technical report, Harvard University. https://www.usenix.org/legacy/event/sec08/tech/full_papers/adida/adida.pdf.

[2] Sufyan T. Faraj Al-Janabi and Noor Hamad. A framework for i-voting based on helios and public-key certificates. Technical report, University of Anbar, 2019. https://www.researchgate.net/publication/335857977_A_Framework_for_I-Voting_based_on_Helios_and_Public-Key_Certificates.

[3] Dan Boneh. The decision die-hellman problem. Technical report, Stanford University. https://crypto.stanford.edu/~dabo/pubs/papers/DDH.pdf.

[4] David Bernhard, Veronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinsch. Adapting helios for provable ballot privacy. Technical report, University of Bristol, England and LORIA - CNRS, France and Universit´e Catholique de Louvain, Belgium. https://bensmyth.com/files/Smyth11-provably-secure-Helios.long.pdf.