The Helios e-Voting Demo for the IACR

Stuart Haber Josh Benaloh Shai Halevi

May 24, 2010

In conjunction with the evaluation of possible electronic election systems for use by the IACR, the election system evaluation committee (consisting of the authors of this report) invited David Chaum and Ben Adida (representing the Punchscan and Helios systems, respectively) to run small demonstration elections for the IACR Board in its August 2009 meeting. The Board heard from both David Chaum and Ben Adida during its August 2009 meeting, and saw a Helios demonstration election. (The Punchscan presentation did not include a demonstration.) At the conclusion of that meeting, the IACR board decided to invite demonstration elections of these two systems for the entire IACR membership.

Since the Punchscan team declined to set up a live demo for the IACR membership, in the winter of 2010 we ran a demo election using only the Helios e-voting system. The goal of that demo election was to solicit more feedback from the IACR membership about the move to electronic elections, and also to help us evaluate the suitability of Helios for the purpose of future IACR elections. This report includes the lessons that we learned and the feedback that we received from the membership during this demo election.

1 Description of the Demo Election

The election server was run at `https://iacr-helios.appspot.com/` using the Google App Engine. The client side was a Javascript/Java combination, which ran in voters' browsers. Extensive documentation of the Helios system, including the source code for the server and client sides, is available on the Helios site at `http://documentation.heliosvoting.org/`.

Each IACR member was sent an email with a link to the election server, and a username and password to use for the elections. We also provided several links to the election server from the IACR web site. The IACR server had several informational pages to describe the process and explain the ballot questions, but it played no other role in the election process. In addition to the "built in" monitoring in the Helios client, Olivier de Marneffe and Olivier Pereira from Université catholique de Louvain implemented an independent Helios Elections Monitor, available at `http://www.uclouvain.be/crypto/electionmonitor/`.

The ballot for the demo election included six questions, three in a choose-one-of-two format, two in a choose-one-of-many format, and the last one in a mark-all-that-apply format. The questions, answers, and vote results were as follows:

1. *Should the IACR keep its current voting system, based on double envelopes sent via postal mail, or switch to electronic voting over the Internet?* (choose one of two):
   - Keep current system: 32
   - Switch to electronic voting: 344
2. *Is the Helios voting system, used for these demo elections, appropriate for the purpose of holding future IACR elections?* (choose one of two):
   - Yes: 293
   - No: 39
3. *Distributing hard copies of the Journal of Cryptology is rather costly. How do you prefer to access the journal?* (choose one of two):
   - Both hard copy and web access, as is done today: 153
   - Electronic form only: 224
4. *In what format would you like to obtain your copy of the proceedings when you attend an IACR conference or workshop?* (choose one of four):
   - A printed book, as done today: 128
   - Only on a USB stick (cheaper than a book): 120
   - Both a printed book and a USB stick (more expensive than just a book): 56
   - I don't need a copy at the conference. Web access to the proceedings is sufficient: 74
5. *Do you think that IACR should move in the direction of open-access publications, where all our publications are freely available on the web?* (choose one of three):
   - No, the current publication structure of the IACR is fine: 61
   - Yes, the IACR should move in the direction of open-access publications: 147
   - Yes, move toward open-access publications, but only if the move does not endanger the impact-factor ranking of our publications: 169
6. *How do you typically access IACR publications?* (mark all that apply):
   - Read them in hard copy from the proceedings or the printed journal: 175
   - Via the IACR reading room at Springer : 174
   - Via the IACR archive: 160
   - Via other means (CDs, USB sticks, etc.) : 100
   - I rarely read IACR publications: 13

There were a total of 379 cast votes, out of the 1542 eligible voters. (This can be compared to the 325 properly-cast ballots received for the IACR elections last fall. But this count was likely influenced by the reminder sent to members about a week before the close of the election – mimicking the process that we may use during an actual IACR electronic election.)

2 Deployment Choices

Hosting the election server. The first choice that we had to make when deploying Helios was whether or not to host the election server on the IACR server. We chose not to do so, for several

reasons. One reason was simplicity of deployment and maintenance, but a more important reason was the threat model for IACR elections.

The stakes in these elections are relatively low, and it seems unlikely that external attackers will go to great lengths to attack them. On the other hand, IACR insiders have much higher stakes in the outcome, and so they should be considered as more likely attackers. In particular, IACR officers (membership secretary, webmaster, etc.) have almost unlimited control over the IACR server, and at the same time are likely to run as candidates for board directorship. Although the tallies produced by a Helios election are universally verifiable, hosting the elections on the IACR server could offer opportunities for ballot stuffing and targeted denial-of-service attacks, and could be perceived as letting the fox guard the henhouse.

Instead of hosting the election server on the IACR server, we chose to host the election server on the Google App Engine infrastructure. This means that we had to trust Google for the availability of our election, and also to a much lesser extent for its correctness. (The reason that correctness is less effected is that the elections are verifiable: If we give people a non-web-based verification method, then it suffices that a few people use that method to ensure that misbehavior will be detected with high probability.)

Authenticating IACR members. The next question to address is how to identify IACR members to the election server. Besides being simple for users and administrators, here too the main concern is to minimize the threat of attacks by IACR insiders. This means, for example, that we do not want members to use their IACR reference-number and password for voting (since the IACR server administrators can see these credentials).

Instead we chose to use a system which is very similar to the one used for the current paper-based elections, as follows: An IACR administrator pulls from the server a list of all the current IACR members, and then sends that list to the election administrator, who uploads it to the election server. (In the current system, the IACR administrator who supplies the list is the Assignments & Database Manager at the UCSB Campus Conference Services. This UCSB manager can play the same role if we switch to electronic elections.) Once the list is uploaded, the election server generates one-time username/password pairs for members and sends these credentials to the members via email. Thereafter, each member needs to use these credentials to cast his/her vote.

Election administration. In choosing the administration functions for the elections, we tried to mimic the roles in the current IACR paper-based elections. Specifically, the election committee chair is the one who should set up the elections, enter all the ballot questions into the system, and upload the list of voters. The returning officer is the one who should initiate the tallying after the voting is over, and all the members of the election committee serve as trustees for the elections (i.e., as the parties who holds the secret keys to decrypt the results).

For the demo election, however, we did not try too hard to stick to these roles, and instead just did all the administration tasks in a cooperative manner. Currently, the Helios server has just one administrator account, so the three of us just shared the password for that account. It might make sense to separate these roles, and give them separate administrative accounts, in real IACR elections.

It should be noted that for the demo elections, Ben Adida (who develops Helios and owns the site

`iacr-helios.appspot.com`) did several of the administration tasks. If we adopt Helios for real IACR elections, then Ben's role (and the associated trust model) will have to be clarified. For example, it is likely that we will still want him to be available for support and debugging, but perhaps he should not be included on the "critical path" of election administration.

It should also be mentioned that the administration interfaces of Helios are not as well polished as the end-user interfaces (e.g., some functions did not work on all browsers). This will probably need to be fixed.

3 Members' Feedback

3.1 The use of Java

By far the most common feedback that we received from members was about the need to have Java installed on the client machine. These feedback messages ranged from merely mentioning this fact, to complaints about inconvenience, to serious reservations about using Java. For example, one member wrote:

> *Requiring Java seems like a non-starter for me. There's been a bunch of security vulnerabilities in the Java runtime that had remained unpatched for too long, and I have explicitly and consciously decided to uninstall it. To ask me to make the trade-off between the convenience of electronic voting and the risk of enabling Java is probably not a great idea: I'm going to say that the risks of using Java is not worth it.*

Another member had the following comment in the same vein:

> *I don't have Java for a reason [...] When I talk about trusting Sun, I'm not talking about trusting Sun with my vote. I'm talking about trusting Sun, and the code produced by Sun, with all the information on my computer, which is more valuable than my vote in an IACR election.*

Given this feedback, it seems clear that if we adopt Helios we must offer members an additional Java-free solution. The browser-based implementation of Helios uses Java to implement the bignum operations that are needed for cryptography, since Javascript implementations would be very slow. Avoiding Java therefore seems to require solutions that are not browser-based. Some options for doing this include using the existing Python client that Helios already has, or developing yet other clients that members can download, so that each member can decide which client he/she wants to use. It would also be possible to offer a Javascript option with a warning that encryption of a ballot could take 30-60 minutes to complete.

Yet another option is to provide a server-based implementation. A member who chooses to use this option would have to trust the server to protect his/her privacy, but would not have to install any client software. (We note that even in a server-based solution, a user can still do cut-and-choose verification to ensure that the server recorded his/her vote correctly.)

3.2 Downloaded code

Beyond Java-specific issues, some members pointed out a "deeper" problem with using downloaded code. One member wrote:

> *Once I started the voting system, I had no clue whether the applet running in my browser was actually the Helios voting application or some other application trying to mimic the Helios voting system. [...]*

Another member added:

> *More generally, the use of downloaded code, whether Java, Javascript, Flash, .NET, or any other language, is unacceptable. Because the code is downloaded every time anew from the server, the user has no assurance that the code that has undergone a security review is the code that is currently running in the browser.*

Here too, offering a few different clients may help mitigate this issue to some extent. But these comments point to a fundamental problem with current frameworks for downloaded code. (Indeed, promoting research into this area may in itself be an argument for adopting such systems by the IACR.)

3.3 Potential attacks

Two of the comments that we received described potential avenues for attacking the Helios system. One potential attack specifically exploits the verifiability property of systems like Helios:

> *If I claim that the code/fingerprint is wrong/malicious, I can cast doubt upon the whole election system. The people who run the election will say everything is okay, but most voters have no way of deciding who is right and who is wrong, and by the time everything is figured out the elections are long past.*

Indeed, an important component in any deployment of a "verifiable scheme" is a complementary complaint-resolution system. In the case of the IACR, such a system could be quite simple: Perhaps it is sufficient to entrust the election committee chair with resolving any complaints (and requiring that he/she reports them to the board). However, this comment points out the importance of devising and publicizing the complaint-resolution process ahead of time.

A much more speculative scenario was described in this comment:

> *I get the invitation email and I mark it as spam. The spam-filters on our email servers learn from this, look at other similar emails, and treat those as spam too. Result: the other IACR members on our email servers never see the invitation, and therefore don't vote. Cost to me: one click of the mouse. Risk to me: zero. Even if people find out what I did, I simply claim that I thought it was spam. Effect on election: I can suppress the vote of half a dozen other members.*

Although quite innovative, this line of attacks on availability does not seems to be very significant, as Ben Adida pointed out in his reply:

> *The current use of Helios depends on IACR being able to send its members an email successfully most of the time and, when that fails, on enough IACR members being aware that an election is going on that they would ask why they haven't received their email-ballot (much like they would have to ask if they hadn't received a paper ballot.) I think that's quite a mainstream requirement for just about any online activity, not unlike "having Internet access."*

It was also pointed out that this member could currently grab the ballots from the unsecured mailboxes of office colleagues, and, in the unlikely event of someone walking in at an inopportune time, simply claim an intention to deliver the ballots personally.

One IACR member noted during the Crypto 2009 rump session that the feature of Helios to allow links to candidate statements could pose a threat. The proposed attack would be to inject targeted malware that could present a fraudulent Helios client from a site included in a candidate statement's link. In response to this observation, a feature was added to the Helios implementation that was used in the membership demonstration election, wherein "challenge" ballots could be posted to the Helios server. The correct decryption of these challenge ballots can now be checked by any election verifier, along with verification of the correctness of the tally. (This relieves individual voters of the need to do independent verification of ballot decryption. Instead, voters only need to check that correct challenge ballot hashes are properly posted, together with the candidates selected in the challenge ballots.) Nevertheless, it might still be worth considering the option of hosting full candidate statements on an IACR server and disallowing explicit links within a candidate statement.

3.4 Other comments

In addition to the "high level" comments above, we received a fair number of comments about various details of the Helios system, from font size, to confusing terminology, to issues with specific browsers on specific platforms, etc. These comments were forwarded to Ben Adida for consideration.

Some members commented specifically about our decision to not host Helios on the IACR sever. For example:

> *The voting system was running from the* `appspot.com` *domain. Even though the TLS/SSL connection to the server authenticated the server for me, as an ordinary user, I have no clue whether* `appspot.com` *is authorized to run the voting application, or whether they were just trying to steal my credentials and run their own voting session against the real voting server.*

Noting that also the email messages were sent from a non-IACR address, another member commented:

> *I did not notice that there was an IACR link in the message. That helps but I would still prefer that the message came directly from an IACR address. Also, I noticed that the PK fingerprints appear in yet another domain* `appspot.com`*. Since this is (supposedly)*

a main verification step then posting the fingerprints in the IACR site would be more appropriate.

Perhaps a reasonable way of addressing these comments is to put several links on the IACR server that point to the election server (as we did in this demo), but also to send email from the IACR server to all the members with these links in them.

4 Conclusions

Overall, the IACR demo election was a fairly smooth experience. With the exception of the Java issue described above, neither we (as administrators) nor the IACR membership (as voters) experienced any serious issues with the functionality of the system. We therefore think that Helios could be an appropriate tool for holding IACR elections, provided that more client options are offered to members.

We also want to point out the excellent support that we received from the Helios development team (Ben Adida, Olivier de Marneffe, and Olivier Pereira). They were very eager to reply to our requirements and to address the issues that came up during this test.

Of course, all the other "generic" issues of e-Voting still apply. In particular, if the IACR is to switch the Internet voting then it should be accompanied by a very strong public statement stating that this technology is NOT appropriate for political elections.