Pretty Good Democracy
(Transcript of Discussion)

Peter Y.A. Ryan

University of Luxembourg

Thanks for showing up first thing on the second day. This talk was going to be entitled Pretty Good Democracy, but out of courtesy to Phil Zimmerman I thought I should email him and ask him if he was happy with me using this title; and initially he seemed to be quite positive about it, but he's handed the rights of the name over to a company and they're not happy about me using the brand name, so strictly speaking the scheme won't henceforth be called Pretty Good Democracy, but for the purpose of the talk I'll refer to it as PGD.

So I'll talk a little bit about the challenge, but I don't think I need to stress that particularly as Matt[1] gave a nice talk about the challenges faced in trying to get secure voting to work properly. Then I'll talk very briefly about the key ideas of this PGD scheme. I should say immediately that I've been working on voting systems for a while, and particularly the Prêt à Voter system.

Prêt à Voter is a supervised polling station type scheme, PGD is a venture into the realm of Internet remote voting, a rather tentative and nervous venture into that area, but we'll come back to that later. I'll give you an outline of the scheme, then I'll talk about some of the threats that we're already aware of, and some of the ideas to try and counter those threats and improve the scheme. I should stress this is very much work in progress, and that it's joint work with Vanessa Teague in Melbourne.

The technical requirements for a voting scheme are that first and foremost we want it to be accurate, the outcome to be guaranteed, and not only accurate but seen to be accurate, and this is often usefully broken down into three phases: the requirement that votes are cast as intended by legitimate voters, are recorded as cast, and then counted as recorded. And of course we also want to ensure ballot secrecy at the same time, and this is what makes the whole problem so intriguing, that we've got these conflicting requirements of auditability and transparency on the one hand, and the secrecy and confidentiality on the other. A lot of these schemes provide notions of voter verifiability, and our scheme for example, does. This provides a kind of voter verifiability, but is subtly different because actually the voter doesn't get a receipt as such.

And of course throughout this we would like to try and reduce the assumptions we need to make (about the technology, the officials, and so on and so forth), to an absolute minimum, and there are discussions about whether it's possible to drive that down to zero, but we do our best to minimise it.

Another requirement that's important, is that, if we're talking about a general voting system, it really has to be extremely easy to use, hopefully just a

[1] Blaze, these proceedings

B. Christianson et al. (Eds.): Security Protocols 2009, LNCS 7028, pp. 131–142, 2013.

very simple linear sequence for the voters, and also easy for the voting officials to understand, and to understand recovery mechanisms, and so on and so forth. Because there are a lot of schemes out there which are very ingenious and involve rather fancy challenge response, cut and choose protocols run between the voter and the device, and these are technically beautiful things and give very high degrees of assurance, but in fact thereby become very vulnerable to social engineering attacks similar to the one that Matt mentioned yesterday where in effect the system deceives the voter as to what the proper sequence of the protocol is.

PGD is an enhancement of code voting, which is a very simple idea, and again I think it's due to David Chaum[2] originally, so it stems if you like from the observation that Internet voting and voter client devices are fundamentally insecure. So the idea is that you distribute by some supposedly secure channel like Snailmail, you distribute so-called code sheets to the voters, an individual code sheet to each voter, and the idea is very simple, in effect they are individual sort of code books for the voter to communicate with the voting system. There are random codes against each candidate, and of course they're distinct for each code sheet.

A typical code sheet then might have the list of candidates, a separate random vote code against each, and an acknowledgement code, a separate one against each, and typically a unique serial number for each code sheet. So the voting process is now very simple, the voter logs on to a vote server, possibly with some additional form of authentication, but maybe not, and simply provides the serial number of the code sheet, and then the code for their candidate of choice. And the vote server is supposed to respond with the correct ack code, it has a database, all the appropriate information, it sends that back, and so the ack code has a sort of dual role, first of all of reassuring the voter that the correct vote code reached the server, and also some degree of authentication I guess of the vote server that they're actually talking to the right device, and not some fake device.

So that's quite nice, you can see how it sidesteps a lot of the vulnerabilities of the Internet, leaving aside denial-of-service attacks. But of course crucially it doesn't provide any end-to-end verifiability. Again, this is a term some of you have come across, a lot of these schemes try to provide the so-called end-to-end verifiability, which is a guarantee that the vote gets traced all the way from being cast to ending up in the final tabulation. You can see from this there's no guarantee of what happens to the vote. There's a degree of guarantee that it's reached the voter server correctly, but after that, whether it actually fetches up in the final count, there's nothing here to guarantee that. So what PGD is attempting to do is enhance this a little bit and strengthen the end-to-end verifiability.

Let's plunge straight into two very simple ideas to enhance this. The first idea is that rather than just having the vote server having access to this database and all the ack codes in a straightforward way, we actually set it up in such a way that the knowledge of the ack codes is shared amongst a set of trustees in some sort of

[2] `http://www.surevote.com/`

threshold fashion, which I'll describe in more detail shortly. The effect of that is that the vote server can't simply return an ack code when it gets a voting code, it has to consult a threshold set of trustees to acquire knowledge of the ack code and thereby return it to the voter. And the idea is that in the process of having to consult the trustees, the trustees all cooperate in registering the vote code.

And the other trick is to actually do away with the separate ack codes against each candidate, we actually just use a single ack code per code sheet. Hopefully why we do that will become clearer in a moment. The point of that is to try and make the thing more receipt free than it would be otherwise.

If the voter gets the correct ack code back, that should provide them a greater degree of guarantee that their vote code will be accurately recorded on a web bulletin board. Again we have this notion of a secure web bulletin board which underlies a lot of these systems, which I don't really have time to go into, but hopefully you've come across before.

Frank Stajano: If the paper contains a series of numbers and some entity has printed it, how do we know that this entity doesn't know how I voted?

Reply: Well I come onto the issue of how these codes might get leaked, which is one of the threats to the scheme.

So if the voter gets the correct ack code back that should give them a fairly strong guarantee — I'll describe how strong a guarantee later — that their vote code has been correctly registered on the web bulletin board, and then basically we can sort of fit a Prêt à Voter backend to the tabulation, which I won't go into the details of, but hopefully you can see roughly, particularly if you're familiar with Prêt à Voter, you can see how the tabulation goes ahead in a verifiable fashion in the backend.

Our code sheets will look slightly different, we don't now have the ack code column, just a single ack code per code sheet, otherwise it looks rather similar.

Frank Stajano: So how does the voter know that the voting server had the code of the correct candidate? The acknowledgement code does not depend on for that.

Reply: No, it's independent of which candidate, but hopefully it will become clear as I get into some of the cryptographic details, why there is a certain guarantee of the correct code.

So the cryptographic set-up, just to give you the high level overview, the idea is, on the web bulletin board we set up a table, each row will correspond to a code sheet, and has in it the voting codes encrypted under the threshold public key of the trustees, and we'll see later that in each row they're in randomised order, and there's a sort of Prêt à Voter style onion in each row which describes how the codes have been permuted in each row.

I'll describe the construction of that in a bit more detail in a few slides, but let's just talk about the voting protocol itself first. It's very simple, again it's very similar to code voting. The voter provides the serial number of their ballot form and the appropriate vote code to the server, and then the server encrypts the vote code under the public key of the trustees along with some sort of zero-knowledge

proof of knowledge of plaintext, so some kind of plaintext-aware encryption, Cramer-Shoup[3] or something (I'll say in a moment why that's important), and posts that to the appropriate row of the web bulletin board. OK, now the trustees step in. First of all they check the validity of the ZK proof, if that's OK then they go ahead and do a plaintext equivalence test of this encrypted term against the terms in the row, and if they find an equivalent they flag that, effectively it's flagged by the plaintext equivalence test. And if they find such a match, then they can do a threshold decryption of the acknowledgement code which is also in that row, and at that point it's revealed to the vote server and it can be sent back to the voiter

Ben Laurie: Does that mean the trustees know how he voted?

Reply: No it won't. The fact that there's this secret randomisation of the vote codes in the row means that the trustees don't know what that signifies in terms of the vote cast.

Michael Roe: What is zero-knowledge proving here, that they know the vote?

Reply: Well let me jump straight into the reason for that: the threat we were concerned about here is that the server might simply look at the web bulletin board, look at the encrypted terms in that row, just pick one at random and re-encrypt it, and submit that at this term.

Ben Laurie: So that may be re-encrypted in the encryption scheme?

Reply: Which we have to do, and you'll see that later.

Matt Blaze: That's a design decision, right?

Reply: Well, yes, we do it for a reason, it helps us with the distributed construction, which you'll see. So as I mentioned, the zero-knowledge proofs and PETs are posted, this is actually quite nice because it makes it harder to corrupt votes, or stuff ballots onto the web bulletin board.

So let me get into the distributed construction, which is what leads us to want to use randomising homomorphic algorithms. There is a question whether this is the right way to go, and we are considering other schemes which don't use that kind of primitive. But as it turns out actually that seems to be a very neat, simple construction which works rather well here. So let's suppose we've got N voters, we've got C candidates, so some voting authority entity generates a set of some multiple λ of $N \times C + 1$, where λ is 2 or 3 or something, depending how many you want to randomly audit subsequently, and they're all encrypted under the trustee public key. And then some further entities, voting clerks, put this whole batch through a set of re-encryption mixes, so they're repeatedly re-encrypted and shuffled. Once we've done this as many times as we want, we just assemble this into a table, λN rows and $C + 1$ columns. And just to remark in

[3] Ronald Cramer and Victor Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", Crypto 1998, LNCS 1462, pp 13–25.

general, this seems to be potentially quite a nice, generic construction, because quite often in voting schemes, and I guess other applications, when we want to produce encryptions of plaintext which are constrained in some way, and we want to prove that the plaintext satisfies some sort of constraint as in some interval or something, but without of course revealing exactly what the value is, and for that we typically have to come up with fairly fancy zero-knowledge proofs of that. Here we've actually managed to avoid this by doing a sort of batch processing which seems to be quite a sort of nice trick, which might be more generic, more generally usable.

So the result is we have a table which has rows of this form, each with $C + 1$ terms in it, for the first C terms will effectively serve as the voting codes for that code sheet, and the final column will be the acknowledgement code for that. I guess I'm assuming here that all the codes have the same form, five or six digit codes or something.

Now we come to the rather critical point, because at some point we've obviously got to print these code sheets and distribute them, and this is the pinch point in the whole scheme. There are various ways we could do this, but the most obvious way is to have a threshold set of trustees decrypt a row and print those values appropriately to the code sheet, and we might use some sort of fancy technology — pressure printing and things like this — to try and preserve the secrecy and so forth. This is the rather tricky stage in the scheme, and for that matter, in code voting. And then they're all handed over to some sort of registrar who distributes them to eligible voters, to their addresses. This is very much the weak underbelly of this scheme.

Jonathan Anderson: Is it very different from systems where you have a great big machine and some encrypted data goes in along with sheets of paper, and ballot papers just come out?

Reply: Well that's the kind of process that I have in mind, yes, how trustworthy that is really not to leak anything, you do need certain assumptions and so on in the process, which is a little bit worrying, but time permitting I'll come back to alternatives we might try to sidestep those issues. So that's given us what we call the P table from which we derived the code sheets. Now we want to produce what we refer to as the Q table (which is actually an analogy to the scantegrity scheme[4]), which is the table that we post to the web bulletin board. This differs from the P table in the sense that each of the rows has this permutation with respect to the P table that I mentioned earlier. So again, we can use a re-encryption shuffle mix to produce this: we can have a series of clerks, so the first clerk takes the P table, and on each row it does a re-encryption of the vote, the first C terms, and permutes them according to some randomly generated permutation, and it stores the information defining that permutation in an onion which it adds at the end of the row. And we can do this repeatedly, as many times as want. The net result is this table I'm calling the Q table, which has these vote codes, the vote codes have all been re-encrypted and

[4] `http://www.scantegrity.org`

multiply shuffled according to some secret shuffle, and the information defining that shuffle, that permutation, is fetched up in this onion at the end, and the ack code we just leave unchanged, and leave it in the same column. OK, so that's the table that we post to the web bulletin board, and is used for recording the votes and tabulating them subsequently. And you can see this has a flavour of Prêt à Voter if you've seen Prêt à Voter before.

Now we've basically set up all the mechanism we need to go ahead, but before we go any further we will want to do random audits of these things, so we pick out a random subset of the code sheets, and for those we require appropriate decryptions of the corresponding rows on the web bulletin board.

Matt Blaze: At this point the code sheets are not yet associated with a particular voter?

Reply: I guess at this point they wouldn't be associated, in fact in some sense ideally they won't be associated with voters at all.

Matt Blaze: At some point, they're going to put the mailing label on each vote.

Reply: Well you'll have to drop them in an envelope and send them off, but hopefully you can actually do that in a way which you don't even record the association.

James Malcolm: So this is down to a subset of the sheets before, and then the remaining ones are sent out?

Reply: Precisely, yes.

Joseph Bonneau: So if you don't know which code sheets go to which voter what's to stop somebody from breaking into the post office and stealing everybody in town's code sheet, and then voting for everybody in town, it won't be able to tell if that's happened?

Reply: Because the mail system is a secure channel, I mean, this kind of problem happens with absentee voting as well.

Joseph Bonneau: So in this scheme as soon as you have any code sheet that enables you to cast a vote?

Reply: Effectively it does, at least in the absence of other authentication mechanisms when you vote in, and that is an issue, and that means of course that vote selling and coercion is a problem. I'll come onto that as well, in a moment. So hopefully you get the gist of the construction and how we audit it.

I've already hinted at some of the threats, and I probably don't have time to go into much detail. I think it's clear that the really weak part of this scheme is the threat of leaking code information, and particularly if we have to at some point print them all to code sheets, that's clearly going to be very difficult to ensure that there isn't some leakage. This is really quite fundamental, because the leakage of the codes isn't just a confidentiality threat, in the scheme, perhaps I should have made it clearer, the key point is that the vote server shouldn't know information of alternate codes, and the point of that is to prevent it when it gets

a vote coming in from trying to guess alternate codes for that code sheet, it doesn't really have an option other than to pass the correct code, the vote code, onto the trustees, OK, in the absence of knowledge of alternate codes for that code sheet, it would just have to guess, and hopefully we've got mechanisms to detect multiple guesses and so forth. Right, so that's the key point. But of course if codes start leaking then we undermine the integrity guarantees as well as the confidentiality, and that's the fundamental worry about this style of scheme, which might be enough to sink it. I hinted that one of the countermeasures is to use these plaintext-aware crypto, there are other possible countermeasures which I'll maybe come to.

Another thing we need to worry about is recovery mechanisms, I think this came out in discussion in Matt's talk. A lot of these schemes talk a lot about how you detect errors, corruption, and so on, but tend to talk very little about well what the recovery mechanisms are when you do start detecting. If you detect certain patterns, what do you do? Clearly we've got to think quite hard about this, one of the issues is if a voter gets an incorrect ack code back, or no ack code, what action should they take? So we have to have clearly defined procedures, and people that they report to, and probably alternative vote servers perhaps they can go to to try again, and so on. Some code voting schemes suggest the use of a finalisation code, a third column in the scheme, so only if the voter gets the correct ack code back do they actually submit the corresponding finalisation code. I'm personally not convinced this is a very effective mechanism, but some people seem to like it.

Jonathan Anderson: Well then the voter would want an ack to their finalisation, so you would need to send another of these?

Reply: Well exactly, this is why it doesn't really seem to buy you that much.

I think that it's the threat of confidentiality, of leakage of vote codes, which is the key weakness here. So we've been toying with various ideas to try and counter that, strengthen that. One obvious thing is perhaps to have dual channels of distribution, so not just send out these codes, print them and send them out over Snailmail.

There are various tricks.

One might be to go back to using visual crypto, there was a scheme long back by Naor and Pinkas[5] to do authentication using visual crypto, you'd send out a transparent sheet with a pattern of pixels on it, and then online you'd send another in effect sheet with pixels, and you'd overlay that on the screen and you'd see the password.

Paul Syverson: Are those schemes based on the one Adi Shamir came up with, and David Chaum's original design?

Reply: Well Adi Shamir did the original visual crypto, David Chaum came up with the use of it in a voting scheme, but there's another scheme which was to do online authentication, which is the one I was alluding to.

[5] Moni Naor and Benny Pinkas, Visual Authentication and Identification, Crypto '97, LNCS 1294, Springer-Verlag, pp 322–336.

Paul Syverson: Do you verify the screen and hold it up, I thought that was part of this?

Reply: No, in David's scheme you print two sheets, but you don't recombine it in the same way. So that might be one possible approach, but it strikes me that's going to be impractical, you know, getting things to the right size so they overlay is not going to be practical.

Another possibility is to have some sort of long term secret the voters hold which they can add to the codes, and then you can use this homomorphism to add them on a web bulletin in some sense, you have to be careful obviously that the algebra meshes. Another possibility, for which I haven't quite got the crypto to work, is to have some kind of scheme where you use a distributed construction and you send out these codes, keep them in their encrypted form, send them out online, but you arrange for voters to effectively get individual decryption keys for each of these codes, so the voter just provides to their device the appropriate decryption key, and so on. But if you can see how to make the crypto work, please let me know.

Ben Laurie: All of this sounds fantastically unusable to me, and the scheme itself sounds pretty unusable. Are you planning to test it?

Reply: Well it depends what you mean by test.

Ben Laurie: I observed the London elections, and if people can't even tick boxes correctly, the chance of them operating this kind of scheme is zero.

Reply: Well OK, when I said it seems simpler than other schemes, I was thinking of other cryptographic verifiable schemes.

Matt Blaze: I'm a little confused. Going back to some of your design constraints, you're assuming the postal system is usable as a secure channel *to* the voter. So why not simply use the postal system as the secure channel *from* the voters? If you have to mail something to someone and you assume the mail can't be tampered with, why not just have them mail their ballot back?

Reply: Well that would just be postal voting.

Matt Blaze: But this is still a postal system, so this seems to combine any trust issues you have with the postal system with any trust issues you have with the cryptography in this system. This means the system can be made strictly more secure by simply having them mail the ballot back without any actual usability issues, and their internet connection doesn't have to work.

Michael Roe: It's different from what a postal voting system is relying on, the difference is between saying that when you mail something through the post it goes to the person that the address is to, and the authentication that when you get the ballot coming in, you know who sent it.

Paul Syverson: Sure, you don't get that from a postal system.

Matt Blaze: But you will have a serial numbered ballot, on some piece of paper that I can tell is the one I sent you, and that's randomised in some way. What problem does what you are doing solve?

Reply: Well that's a good question, and one of the things we're hoping to get here is some degree of end-to-end verifiability, which I don't think postal voting gives.

Matt Blaze: Right, I think that's a good answer. Because you get the ack back, there's a two step protocol with standard absentee ballots, this is in fact a three step protocol, because of the acknowledgement code.

So you get a confirmation that it's been received, right.

Reply: Yes, and hopefully even slightly stronger than that: subject to certain assumptions there's a degree of end-to-end verification here. If you get the ack code that should imply that your vote gets accurately counted in the final tabulation. That's really what we're trying to achieve here. The starting premise was, well people have proposed code voting, it's not end-to-end, can we make it a bit more end-to-end, that was the starting intellectual challenge. Whether the final thing is really viable for anything practical is still I think open to question.

Virgil Gligor: Democracy is probably the most important multi-party computation. Now the part of that multi-party computation that security research is focusing on is integrity of the input. But in computation in general, input is probably not the most interesting part, and in all these stolen elections in recent years, very few of them have been stolen by people really cracking the crypto of submitting the vote. So there is the interaction between how we submit these inputs, and what is done in computation with these inputs, and whether people are turned away from inputting anything at all.

I'm wondering whether the techniques developed would justify maybe extending or zooming out from just input operations, and looking at broader security properties that one might require from democracy rather than just the integrity of the vote. So how the votes are counted, in a different kind of research about democracy they say, you give me who you want to win and you give me a set of preferences, and then I am going to design the counting system which will give you the desired winner. Is there any chance of interaction between these two threads of research?

Reply: Well I think it's starting to happen in the community, there is a bit of a community looking at the science of voting, in fact I think they're thinking of trying to start a journal precisely on that topic, which would go as far as looking at decision theory, and census theory, as well as things like securing the voting process and so on, so I think that is emerging.

Virgil Gligor: Can you say how you think cryptographic techniques and social choice techniques interleave?

Reply: Well to a large extent I think they're orthogonal. If you want to come up with one of these cryptographic schemes and you want something fancy, say single transferable vote, then you're going to have to make sure that you can

carry the data items, the encoding of the right vector or something. One of the nice things about Prêt à Voter is that it seems to be rather good at doing that kind of thing in contrast to some of the other cryptographic schemes.

Matt Blaze: I think I can achieve what you're doing with a postal system. I think I can improve all your properties with a postal system simply by sending out a serial numbered ballot to everyone, they return the ballot, and then the ballots are available for inspection, all ballots are available for public inspection. So, if I want I can go after the count and I can confirm that my ballot was received, and I can count them myself if I want.

Reply: OK, but isn't that running into receipt-freeness type issues?

Matt Blaze: Well in any postal system we're not coercion free. If I'm mailing you something I'm still subject to coercion by somebody who supervises when I open the envelope and cast my vote, so we still have that problem.

Reply: I was going to come on to coercion.

Matt Blaze: If we're willing to sacrifice that property, which the PGD system seems to do, then I think we can achieve this without any crypto at all.

Reply: You're just posting anonymised ballots, and then you can go and look at the list of all the ballots and check that mine is there.

Matt Blaze: Oh yes, I can see that mine is there, and I can see that this other one who I don't know who's it was is also there.

Reply: Yes, and then you anonymise at the level of your voting precinct or whatever.

Matt Blaze: Right.

Reply: Well arguably yes, and subsequently I'm not making great claims for this scheme.

Matt Blaze: Oh no, I'm just trying to get at understanding what problems we're solving here, given that we're assuming the postal service is a secure channel[6]. If we're willing to do that it seems that we can achieve many properties.

[6] Editors' Note: The paradox here is similar to that which occurs with classical one-time pad. If a leak-proof channel, authenticated at both ends, exists to distribute the key from Alice to Bob, why not use that channel to send the message, which is of the same length? There are two obvious answers. The first is that Bob may be in a hurry at the point when he wishes to send the message, but ample time may be available beforehand for Alice to distribute the key. Quick voting outcome online may be a goal. The second, usually more important reason is that the slow key distribution channel need only be leak-evident, rather than leak-proof. Unsuccessful key distribution can be corrected. The final point to note is that if the message to be sent from Bob to Alice is short (relative to the key), then a uni-directional leak-evident channel going the "wrong" way between authenticated endpoints (i.e. conveying information only from Alice to Bob) suffices to provide confidentiality, authentication, and integrity for a message sent from Bob to Alice over an open, untrusted channel. A particular bit pattern can be reserved to mean "the key was compromised".

Reply: Well maybe, we should talk a bit more about this. I've wondered about how much you can do with pure voting and degrees of traceability subsequently in the tabulation process. We should come back to that.

George Danezis: I think the problem you describe, is a problem with your abstraction of elections. Effectively, as someone else said before, the process starts when people start submitting votes. You assume that there is already a secure registration process and a secured interaction process between the citizens and the State, to actually start. Getting a list of voters, let's say, already requires you to have some kind of secure channels and all that stuff, so probably integrating this phase into the voting scheme will help you solve this problem, rather than making the problem more complicated.

Reply: I take your point, it's certainly true of all these schemes, that it's all part of a much larger system which is the setting up of the electoral roll.

George Danezis: So far we've been shying away from looking at the registration processes which are key, because a lot of the fraud happens there. We fear they would make these systems more complicated, but I think they would actually simplify a lot of the things by making the assumptions concrete.

Reply: Yes, I think that's probably true, I agree.

Joseph Bonneau: Vote selling in this system is basically equivalent to mail absentee voting. Is that correct?

Reply: Well let's go with it for the moment.

Joseph Bonneau: It seems like vote selling is actually a lot worse now though because you can create a website and say, give me your vote code for Candidate A, and then the website will cast your vote and when it sees your ack code, and then it will send you a dollar on PayPal or whatever, and that seems much more likely to happen than somebody actually going and finding somebody to sell their absentee ballot in person.

Jonathan Anderson: But there's no way to confirm that it is the code for that candidate.

Bruce Christianson: You can give it any vote and still get your dollar, because the ack code is the same for all candidates.

Reply: Right. I'd better try and wind things up because we're overrunning. I was going to touch on coercion resistance and just say that as it stands it clearly isn't coercion resistant or resistant to vote buying. Potentially we could add extra mechanisms like the Juels Catalano Jakobsson type tokens, although it's actually not so easy to see how to integrate it with this scheme because with the posting of material to the web bulletin board, if we're going to have to do a sort of re-voting process it's not quite clear how we overwrite or append further information to the web bulletin board, so that's kind of tricky.

So let me just wind up on the discussion. All I would say for this scheme is that it perhaps does buy you something in degrees of convenience. I claim it's slightly

easier for the voter than some other schemes because for example with Prêt à Voter you cast your vote, you get some kind of receipt, and then subsequently you go to the web bulletin board and check that it's correctly posted there.

That's a bit more of a palaver, and there is some question whether people would bother to do that checking process. At least here everything happens in a single session, in theory you get your ack code back during the same session, you can check it immediately, so in that sense compared to some verifiable schemes it does seem to be a bit easier and simpler, more immediate for the voter.

I certainly wouldn't claim that a scheme like this is suitable for general political elections, but it maybe OK for student elections and things like this. The International Association of Cryptologic Research have been thinking recently about moving to Internet voting.

Matt Blaze: Perhaps not the most hotly contested elections in the world.

Paul Syverson: But one of the few where you will have people who will actually try to hack it themselves.

Reply: Yes, exactly.

There's been a long and rather fascinating debate about whether this is a good idea for all kinds of reasons you can imagine, not least that if it's successful, it might be seen as showing a precedent which politicians will then go along and say, if it's good enough for the International Association of Cryptologic Research, it's good for ...