Delegated Enforcement in Secure Protocols

Raphael Yahalom

Onaro Research *
yahalom@onaro.com

Workshop on Issues in the Theory of Security - Portland 2002

Extended Abstract
Nov 10th, 2001

1 Introduction

Many secure protocols are designed to rely on the participants for verifying that particular executions completed correctly (for example, for determining whether any other participant has cheated, e.g. [2, 4, 6]).

However there are various applications in which an important design goal is to minimize the time, and the number of interactions, that each participant needs to be involved with directly. In such applications (e.g. lotteries, elections, auctions, and various types of long-lived e-commerce transactions), it is desirable to enable participants to terminate as early as possible, while ensuring that their interests will still be preserved.

The contribution of this paper is to introduce a new protocol design principle, *delegated enforcement*, which enables various (almost all) participants to terminate early, and obtain guarantees that their interests will be correctly preserved by the remaining delegates until after the completion of the full protocol, *even if some of these delegates are untrustworthy.*

* The author is performing part of this research work at MIT - Sloan School of Management

Incorporating *delegated enforcement* in protocol design may be trivial in some cases (e.g. ones in which participants may obtain guarantees from multiple similar delegates) and may be quite subtle in others.

We demonstrate the new design principle of *delegated enforcement* by introducing a new, improved, election protocol that consists of 3 non-similar delegates. In election protocols it is desirable to enable voters to complete their duty as quickly and as conveniently as possible, while providing them with guarantees regarding the eventual integrity of the elections results, as well as the privacy of their own vote.

2 Delegated Enforcement

Informally, the principle of *delegated enforcement* consists of participants obtaining sufficient guarantees from delegates before terminating. These guarantees should enable each participant to conclude, not only that the appropriate delegates possess all the participant's information to ensure correct eventual protocol completion, but also that delegates possess enough evidence to incriminate their colleagues in case one or more of them deviate from the correct execution.

The latter condition ensures that incorrect protocol outcomes will be detected and declared, and also that the motivation, and so the likelihood, of such violations will be reduced (as delegates deduce that they will be incriminated).

We consider execution environments in which each principal is pre-assigned with a private key and has reliable access to the public-key of any other participant (i.e. we assume an appropriate Public-Key Infrastructure).

A participant in a given protocol execution is considered *provably incriminated* if there is a provable internal contradiction in any statement, or across a number of statements, signed with the private key of that principal in that execution.

A subset of participants in an execution of a secure protocol are pre-designated as *delegates*.

The principle of *delegated enforcement* enables each non-delegate participant to terminate after it receives a *delegation confirmation statement* signed by one or more of the delegates, designed such that the non-delegate may conclude that only two outcomes are possible:

1. the protocol execution will complete correctly and will be verified by the delegates, or

2. the protocol execution will complete incorrectly and a guilty delegate (or multiple guilty delegates) will be declared *provably-incriminated* by the other delegates.

Designing a protocol with a *delegated enforcement* principle may be not straightforward, especially in cases in which the multiple delegates involved are not similar and in which there are constraints on the information that each delegate may possess.

In some respect*Delegated enforcement* can be considered as related to a two-level electronic contract signing (cf. [1, 8]). At one level there are contract-like interactions between a participant and multiple delegates, and on the second level there are contract-like interactions between the delegates themselves (on behalf of participants).

We demonstrate the *delegated enforcement* principle in a new, improved, election protocol that is designed to enable voters to complete their voting process in a single interaction, while relying on 2 delegates (and one passive arbitrator delegate) to ensure the appropriate outcome of the elections. Each of these delegates performs a different role in the protocol and is constrained, for voter confidentiality reasons, in the amount of information it can possess.

3 A New Elections Protocol

We introduce a new elections protocol that relies on the *delegated enforcement* design principle to attain attractive properties and to improve on the earlier protocols in [10],[6], and [5]. All these protocols were designed as practical solutions for attaining the properties desired for electronic elections in realistic setting and with low performance overhead (e.g. [7]).

The new delegated-enforcement improvement in the new protocol enables it to attain the same desirable properties (in some cases in a stronger form), while reducing to a minimum the amount of involvement required by each voter (for example eliminating the need for each voter to check at a later point in time that her vote was indeed recorded correctly).

In this protocol the non-delegate participants are voters V_i and there are 2 delegates (election servers) A and C. In addition there is a 3rd passive delegate E (called an eNvestigator in [10]) that normally does not participate in the execution but is invoked by any of the participants in case of a detected violation.

The election protocol requires strong secure integrity and privacy properties:

- **Integrity:** The published election results will correctly reflect all, and only, the valid ballots submitted by eligible voters;
- **Privacy:** No one will be able to deduce unilaterally the content of the ballot of any eligible voter that submitted a valid vote;

In the notation we use below, $\{\ msg\ \}_{K_{P+}}$ represent the public-key encryption of msg with participant P 's public key, and $\{\ msg\ \}_{K_{Q-}}$ represents msg digitally signed with Q's private-key.

We denote by $\mathcal{CO}_s(msg)$ the *bit-commitment* of msg with secret value s , [9], and by $\mathcal{BL}_r(msg)$ the *blinding* of msg with a factor r , [3]. A comma $''\ ,\ ''$ denotes concatenation between message components.

$ballot_i$ contains the vote of voter V_i and also contains a secret random ballot identifier b_i. Another secret random value r_i is generated and used by V_i for the blinding of $ballot_i$.

We assume that all the cryptographic primitives used are strong in the conventional sense, and that all keys and random values are of sufficiently high quality (e.g. sufficiently long).

Furthermore we assume that each encrypted and signed message components implicitly includes a *component identifier* header, containing fields such as message type, a unique global identifier of the elections, etc.

Finally we assume that the list of all eligible voters is reliably publicly available a-priori.

The protocol consists of the following steps:

Voting Phase:

1. $V_i \rightarrow A$: $\{\ V_i\ ,\ M_{2i}\ ,\ M_{4i}\ \}_{K_{A+}}$

where:

$M_{2i} = \{\ I\ \ voted\ \}_{K_{V_i-}}$
$M_{4i} = \{\ M_{1i}\ ,\ M_{3i}\ \}_{K_{V_i-}}$
$M_{1i} = \mathcal{BL}_{r_{1i}}(ballot_i)$
$M_{3i} = \{\ r_{1i}\ \}_{K_{C+}}$

2. $A \rightarrow C$: $\{\ \{\ M_{3i}\ \}_{K_{A-}}\ \}_{K_{C+}}$

3. $C \rightarrow A$: $\{\ \{\ \{\ M_{3i}\ \}_{K_{A-}}\ \}_{K_{C-}}\ \}_{K_{A+}}$

4. $A \rightarrow V_i$: $\{\ \{\ \{\ M_{3i}\ \}_{K_{A-}}\ \}_{K_{C-}}\ \}_{K_{V_i+}}$

Results Phase:

5. $A \rightarrow C$: $\{\ \{\ L_0\ ,\ L_1\ ,\ L_2\ \}_{K_{A-}}\ \}_{K_{C+}}$

where:

$L_0 =\ \ M_{3j}\ ,\ \dots\ ,M_{3k}$
$L_1 =\ \ \{\ M_{1j}\ \}_{K_{A-}}\ ,\ \dots\ ,\{\ M_{1k}\ \}_{K_{A-}}$
$L_2 =\ \ M_{2p}\ ,\ \dots\ ,M_{2q}$

6. $C \rightarrow A$: $\{\ \{\ R_1\ \}_{K_{C-}}\ \}_{K_{A+}}$

where:

$R_1 =\ \ \{\ ballot_u\ \}_{K_{A-}}\ ,\ \dots\ ,\{\ ballot_w\ \}_{K_{A-}}$

7. $A\ \ publishes:$ $\{\ \{\ R_1\ \}_{K_{C-}}\ \}_{K_{A-}}$

Note that if C can perform traffic analysis on messages between V_i and A (and thus may possibly match these messages with the corresponding messages 2 it receives and so violate V_i's confidentiality) then A may need to introduce appropriate random delays between steps 1 and 2 and between steps 3 and 4 (and perhaps aggregate multiple messages 2 of multiple voters and forward to C together).

Also note that the entries orders in list L_2 is randomized by A before step 5, as is the order in list R_1 by C before step 6.

4 Delegated-Enforcement Analysis

In this protocol the *delegation confirmation statement* for voter V_i is

$$\{ \{ M_{3i} \}_{K_{A-}} \}_{K_{C-}}$$

(which she receives in step 4).

Assuming that A and C do not collude to cheat together (for example because they are administered by two rival political parties), after she receives this *delegation confirmation statement*, each V_i can complete her involvement in the elections and conclude after the eventual end of the elections either the results list (R_1) will reflect correctly the valid votes submitted by all the eligible voters, or one of the servers will be *provably-incriminated*.

In particular, each V_i after receiving the *delegation confirmation statement* may conclude that unless A or C is *provably-incriminated* both of the following will hold:

1. The election results list R_1 will include her own valid vote $ballot_i$ (and similarly will include the valid vote $ballot_j$ of any other eligible voter who submitted a valid vote and verified the receipt of her corresponding *delegation confirmation statement*), and

2. The election results list R_1 will not include any additional ballots, other than a single $ballot_k$ for every eligible voter that voted correctly.

Proof: (*sketch - the full proof is omitted from this extended abstract*)

We assume that n voters completed their votes and received their correct *delegation confirmation statements.*

There are two possible classes of violations:

1. At least one $ballot_i$ of these n voters is not included in R_1;

2. At least one $ballot_x$ is in R_1 which was not submitted by any of these n voters.

1. For violation class 1 there are 2 possible cases:

   Case 1.1: M_{3i} or M_{1i} were not included by A in message 5. C detects the inconsistency and possesses evidence (messages 2 and 5) to *provably-incriminate* A.

   Case 1.2: M_{3i} or M_{1i} were included by A in message 5 and C omitted $ballot_i$ from R_1. A detects the inconsistency when the number of entries in R_1 (message 6) is different than the number of entries in L_1. A possesses evidence (all of messages 3 and message 6) to *provably-incriminate* C, unless C can prove that M_{3i} and M_{1i} result in an invalid ballot. In such a case (with the help of E to preserve the appropriate voter confidentiality guarantees) it is established (using also M_{4i} obtained from A) whether A is *provably-incriminated* or voter i submitted an invalid ballot.

2. For violation class 2 there are 4 possible cases:

   Case 2.1: There is an internal inconsistency in message 5: either M_{3x} and M_{1x} appear more than once, or the number of entries in L_0 and L_1 is not equal to the number of entries in L_2. C detects the inconsistency and possesses evidence to *provably-incriminate* A.

   Case 2.2: M_{3x} and M_{1x} are included by A in message 5 and C has not signed M_{3x} in message 3. C detects the inconsistency and possesses evidence to *provably-incriminate* A.

   Case 2.3: M_{3x} and M_{1x} are included by A in message 5 and C has signed M_{3x} in message 3 (among the n it signed), but M_{3x} has been generated by A and has not been submitted by any of the n voters. In such a case at least one of the n voters (whose signatures appear as entries in L_2) did not receive a proper *delegation confirmation message* (signed also by C) - contrary to our assumption.

   Case 2.4: R_1 contains $ballot_x$ more than once. A detects the inconsistency (each ballot has been originally generated with a unique random identifier) and possesses evidence to *provably-incriminate* C.

Furthermore it can be established that privacy is not compromised - even if one of the servers violates the correct execution it cannot obtain any information about the vote of any valid voter V_i.

Finally it can be established that no framing is possible - a server that did not violate the correct execution cannot be *provably incriminated*.

Note that the above protocol may be extended to address denial-of-service challenges as well as scenarios in which multiple servers may cheat in collusion. We do not discuss such extensions in this paper.

5 Conclusions

A new design principle for secure protocols, *delegated enforcement*, was introduced and demonstrated by a new improved elections protocol. The new principle can be advantageous for a variety of protocol classes.

There are a few related interesting areas of current research including extending the scope of *delegated enforcement* and determining in general when, and how efficiently, can this principle be used.

References

[1] N. Asokan, V. Shoup, and M. Waidner "Optimistic Fair Exchange of Digital Signatures" *Proceeding of EUROCRYPT'98*, 1998, pp. 591-606.

[2] J. Benaloh and M. Yung "Distributing the Power of a Government to Enhance the Privacy of Voters", *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing*, 1986, pp. 52-62.

[3] D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete", *Communication of the ACM*, Vol 28, No 10, pp. 1030-1044 Oct 1985.

[4] R. Demillo, N. Lynch, and M. Merritt, "Cryptographic Protocols", *Proceedings of the 14th Annual Symposium on the Theory of Computing*, 1982, pp.383-400.

[5] A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Applications", *Advances in Cryptology - AUSCRYPT '92, Lecture Notes in Computer Science*, Vol 718, pp. 244-251, Springer-Verlag, 1992.

[6] M. Herschberg, “Secure Electronic Voting over the World Wide Web”, Master thesis, MIT - Laboratory of Computer Science, May 1997.

[7] R. Mercuri, ”The Electronic Voting Enigma: Hard Problems in Computer Science” Talk at The Cryptography and Information Security Group of MIT's Laboratory for Computer Science, October 19, 2001.

[8] S. Micali, ”Certified E-mail with Invisible Post Offices” Talk at RSA Security Conference, 1997.

[9] M. Naor, “Bit Commitment using Pseudo-Randomness”, *Advances in Cryptology - Crypto '89, Lecture Notes in Computer Science*, Vol 435, pp. 128-136, Springer-Verlag, 1990.

[10] R. Yahalom, ”Optimistic Trust with Realistic eNvestigators”, *Security Protocols - Lecture Notes in Computer Science*, Vol 1550, pp. 193-202, Springer-Verlag, 1998.