Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 135 (2005) 115–134
www.elsevier.com/locate/entcs

Static Validation of a Voting Protocol

Christoffer Rosenkilde Nielsen[1] Esben Heltoft Andersen[2]
Hanne Riis Nielson[3]

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, bldg. 321, DK-2800 Kongens Lyngby, Denmark

Abstract

The desired security properties of electronic voting protocols include verifiability, accuracy, democracy and fairness. In this paper we use a static program analysis tool to validate these properties for one of the classical voting protocols under appropriate assumptions. The protocol is formalised in an extension of the LYSA process calculus with blinding signatures. The analysis, which is fully automatic, pinpoints previously undiscovered flaws related to verifiability and accuracy and we suggest modifications of the protocol needed for validating these properties.

Keywords: Static Program Analysis, Voting Protocols, LYSA.

1 Introduction

Electronic voting promises a convenient and inexpensive alternative to the classical paper vote. Due to the rapid growth in computer networks, most people nowadays have access to the internet. This makes electronic voting a viable alternative for governmental elections, as well as small scale elections and surveys. However, the use of electronic voting systems introduces new ways to systematically disrupt the voting or falsify the result. If these systems are to replace the classical way of voting, the communities that hold the elections should be convinced of their correctness.

[1] Email: s991190@student.dtu.dk
[2] Email: s991546@student.dtu.dk
[3] Email: riis@imm.dtu.dk

doi:10.1016/j.entcs.2005.06.001

The first step in that direction is to identify what security properties a voting protocol should satisfy. Security properties for electronic voting systems differ from those in ordinary protocol schemes; as in [12] these can be summarized into four main properties:

- **Verifiability**: A system is verifiable if the voters independently can verify that their votes have been counted correctly.
- **Accuracy**: The accuracy of a voting system is divided into three parts: (1) it is not possible for a vote to be altered, (2) a validated vote cannot be eliminated from the final tally and (3) an invalid vote cannot be counted in the final tally.
- **Democracy**: A system ensures democracy if (1) only eligible voters can vote and (2) eligible voters can only vote once.
- **Privacy**: In a voting system the privacy is obtained if nobody can link any ballot to the voter who cast it.

Often a fifth property is added [3,14]:

- **Fairness**: No early results from the voting can be obtained.

In this paper we show that static program analysis [23] can be used to validate several of these properties, in particular the properties of verifiability, accuracy, democracy and fairness. We illustrate this for the FOO92 protocol developed for large scale elections by Fujioka, Okamoto and Ohta [14]. This protocol makes use of Chaums *blind signatures* [10,11] which is a mechanism allowing a message to be signed by another party without revealing any information about the message to the other party. Assuming perfect cryptography, blinding is a cryptographic primitive obeying the following two rules:

(Unblind 1) $unblind_b(blind_b(msg)) = msg$
(Unblind 2) $unblind_b(sign_s(blind_b(msg))) = sign_s(msg)$

Here msg is a message, b is a cryptographic key known as the blinding factor and s is a digital signature. The second rule is the most interesting one as it expresses that a signed blinded message can be unblinded without disclosing any information about the message itself; note that the signature of the message is not destroyed. The first rule simply states that blinding acts as symmetric encryption when no signature is present.

In Section 2 we present the FOO92 protocol in more detail. In order to formalise the protocol in the LySa calculus [6,7] we introduce an extension of this calculus in Section 3 and proceed by formalising the FOO92 protocol in the extended LySa calculus in Section 4. The extensions of the static analysis corresponding to the extensions of the LySa calculus are then developed in Section 5; they have been implemented in the LySaTool [9,19] which is used to

automatically produce the analysis results presented in Section 6. In Appendix A a more thorough description of the blinding extension has been provided.

2 The FOO92 Voting Protocol

The FOO92 protocol [14] involves three kinds of principals: There are multiple voters V, one administrator A and one counter C. The administrator ensures that only legitimate voters are allowed to vote and the counter collects, publishes and counts the votes. In addition to digital signatures, encryption and blinding the FOO92 protocol incorporates another cryptographic primitive, bit-commitment [22]. This is a method by which the voter can commit to a bit without revealing what it is. Later the bit can be revealed by the voter by providing the commitment key. The protocol proceeds in five phases as shown in Table 1 and is further explained below.

1. V	$\rightarrow A : V,\ sign_V(blind_b(commit_r(v)))$	Preparation Phase
2. A	$\rightarrow V : sign_A(blind_b(commit_r(v)))$	Administration Phase
3. (V)	$\rightarrow C : sign_A(commit_r(v))$	Voting Phase
4. C	$\rightarrow\ : l,\ sign_A(commit_r(v))$	Publishing Phase
5. (V)	$\rightarrow C : l,\ r$	Opening Phase

Table 1
Protocol Narration for FOO92

In the preparation phase (1) the voter V selects vote v and computes the bit-commitment $x = commit_r(v)$ using a random number r and the bit-commitment function *commit*. The commitment is then blinded using the blinding factor b and the resulting $e = blind_b(x)$, called the *ballot*, is signed $s = sign_V(e)$ and sent to the administrator A.

In the administration phase (2) A verifies that V has the right to vote, has not applied for a signature yet and that s actually is V's signature of e. If this is the case then A signs the ballot $d = sign_A(e)$ and sends it back to V.

When V receives the ballot signed by A the voting phase (3) begins. V checks that the signature on the ballot originates from A and unblinds the signed ballot $y = unblind_b(d)$ thereby obtaining a signed version of the committed vote, that is $y = sign_A(x)$. The voter then sends the signed ballot y to the counter over an anonymous communication channel, this is denoted by (V) as the sender in the narration.

In the publishing phase (4) the counter receives y, checks the correctness

of the signature and enters (l, y) onto a list as the l-th item. After all votes are received e.g. after a fixed deadline, C publishes the list with all entries.

In the last phase, the opening phase (5), the voter checks that his ballot x is in the list and sends l together with the commitment key r to C on an anonymous channel. When C receives r he is able to open the ballot and count the vote v.

Assumptions. In order to analyse this protocol, we have to make some specific assumptions. These assumptions are described in or can be derived from the original protocol description [14]:

- Bit-committed votes are unique;
- The administrator only signs one vote for each eligible voter;
- The counter C is a trusted party, ie. if the counter receives a vote then it is also counted correctly in the final tally;
- The counter must have received all votes in the voting phase before commencing the publishing phase;
- The voting is only accepted if the number of votes counted by the counter equals the number of votes signed by the administrator; and
- The voting is only accepted if the counter in the opening phase receives all the commitment keys for the votes published.

3 LySA-Calculus with Blinding

In order to apply our analysis technique we have to formalise the protocol narration as a process in the LySA-calculus. LySA is a process calculus in the π-calculus [21] tradition and uses ideas from the Spi-calculus [1] for incorporating cryptographic operations. LySA simplifies matters compared to other calculi in that all messages are sent on a global network, the *ether*.

The details of LySA are described in [6,7]; however in order to analyse the FOO92 protocol we need to extend the calculus with a blinding construct. The resulting syntax for terms is given in Table 2.

The names n will be used to represent shared keys, commitment keys as well as blinding factors and as we shall see later bit-commitments are represented as symmetric encryption using a shared key as the commitment key. As usual digital signatures are obtained using asymmetric encryption with a private key. The special construct $[\![E_1, \cdots, E_k]\!]_{E_0}$ is used for blinding the tuple $E_1, \cdots, E_k$ of values with the blinding factor E_0.

The syntax of processes is given in Table 3. In addition to the classical constructs for composing processes, LySA contains an input construct with

E	::=	x	variable
		n	name
		m^+/m^-	public/private keypair
		$\{E_1, \cdots, E_k\}_{E_0}$	symmetric encryption
		$\{\!\{E_1, \cdots, E_k\}\!\}_{E_0}$	asymmetric encryption
		$[E_1, \cdots, E_k]_{E_0}$	blinding

Table 2
Terms for LYSA with blinding

P	::=	0	nil
		$P_1 \mid P_2$	parallel
		$!P$	replication
		$(\nu\, n)\, P$	restriction (name)
		$(\nu_{\pm}\, m)\, P$	restriction (keypair)
		$\langle E_1, \cdots, E_k \rangle.P$	output
		$(E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k).P$	input
		$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$	symmetric decrypt.
		$\mathsf{decrypt}\ E\ \mathsf{as}\ \{\!\{E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k\}\!\}_{E_0}\ \mathsf{in}\ P$	asymmetric decrypt.
		$\mathsf{unblind}\ E\ \mathsf{as}\ [E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k]_{E_0}\ \mathsf{in}\ P$	unblinding

Table 3
Syntax for LYSA with blinding

matching and two decryption operations with matching; the construct for unblinding follows the same trend. In the case of input the idea is that the pattern $(E_1, \cdots, E_j; x_{j+1}, \cdots, x_k)$ must be matched towards a tuple $(E'_1, \cdots, E'_k)$ of the same length and it only succeeds if the j first components pairwise equals one another, i.e. $E_1 = E'_1, \cdots, E_j = E'_j$. If this is the case the remaining $k - j$ values $E'_{j+1}, \cdots, E'_k$ are bound to the variables $x_{j+1}, \cdots, x_k$. The idea behind the pattern matching of the constructs for symmetric and asymmetric decryption is similar with the only modification that in the case of symmetric encryption the two keys must be equal whereas for asymmetric encryption they must form a key pair. The unblinding construct takes the form $\mathsf{unblind}\ E\ \mathsf{as}\ [\![E_1, \cdots, E_j;\, x_{j+1}, \cdots, x_k]\!]_{E_0}\ \mathsf{in}\ P$. As already explained in the introduction the construct may act as an ordinary decryption using the blinding factor as a symmetric key (rule unblind1) and in this case E must take the form $[\![E'_1, \cdots, E'_k]\!]_{E'_0}$ in order for the construct to succeed and furthermore the conditions $E_0 = E'_0, E_1 = E'_1, \cdots, E_j = E'_j$ must be fulfilled. When succeeding the variables $x_{j+1}, \cdots, x_k$ will be bound to $E'_{j+1}, \cdots, E'_k$ as explained above. The more interesting alternative arises when E evaluates to a signed blinded value, i.e. has the form $\{\!|[\![E'_1, \cdots, E'_m]\!]_{E'_0}|\!\}_{E_s}$. In this case the pattern of the unblinding construct will take the special form $\mathsf{unblind}\ E\ \mathsf{as}\ [\![; x]\!]_{E_0}\ \mathsf{in}\ P$

and the match will succeed when $E_0 = E'_0$. The variable x will then be bound to the signed value $\{\!|E'_1, \cdots, E'_m|\!\}_{E_s}$ as already illustrated by the rule (Unblind 2) of the introduction. For completeness the formalisation of these rules are given in Appendix A.

Annotations in LySA with blinding. To describe the intention of protocols, the terms and syntax of the cryptographic primitives are decorated with labels ℓ called *crypto-points* and *assertions* of one of two forms:

- each encryption/blinding is annotated with a crypto-point ℓ and an assertion of the form [dest $\mathcal{L}$] meaning that the corresponding decryption/unblinding is intended to happen at one of the crypto-points mentioned in the set $\mathcal{L}$ of crypto-points.
- each decryption and unblinding operation is annotated with a crypto-point and an assertion of the form [orig $\mathcal{L}$] meaning that the value being decrypted or unblinded is intended to come from one of the crypto-points of $\mathcal{L}$.

The set $\mathcal{L}$ should of course be a subset of the entire set of crypto-points $\mathcal{C}$ occurring in the protocol. The annotations will be used in the analysis which is described in Section 5.

4 Modelling FOO92 in LySA

We are now ready to model the FOO92 protocol in LySA . This is done in two stages: First we shall refine the specification given in Table 1 into an extended protocol narration, which distinguishes between inputs and corresponding outputs and also makes clear which checks must be performed. In the second stage the extended protocol narration is translated into LySA (with blinding).

Extended protocol narration. The extended protocol narration is listed in Table 4 where we use the LySA terms and syntax for writing the cryptographic operations.

First observe that each message is extended with source and destination information along the lines of IPv4 and IPv6. Upon receipt of a message the principal will always check whether the message is intended for him; occasionally he will also check that the sender is who he expected. Note that these components of the message are sent in clear text and are therefore forgeable.

As mentioned earlier we model bit commitment (message 1) as symmetric encryption with the commitment key r. Digital signatures are modelled using asymmetric encryption with the principals private key (messages 1 and 2) and verification of a signature is then modelled using asymmetric decryption

1.	$V\rightarrow$:	$V, A, V, \{\!\|[\{v\}_r]_b\|\!\}_{K_V^-}$	
1′.	$\rightarrow A$:	y_V, y_A, y'_V, y_1	[check $y_V = y'_V$ and $y_A = A$]
1″.	A :	decrypt y_1 as $\{\!\|y_2\|\!\}_{K_{y_V}^+}$	[check V's signature]
2.	$A\rightarrow$:	$A, y_V, \{\!\|y_2\|\!\}_{K_A^-}$	
2′.	$\rightarrow V$:	x_A, x_V, x_1	[check $x_A = A$ and $x_V = V$]
2″.	V :	decrypt x_1 as $\{\!\|x_2\|\!\}_{K_A^+}$	[check $x_2 = [\{v\}_r]_b$]
2‴.	V :	unblind x_1 as $[x_3]_b$	
3.	$(V)\rightarrow$:	D, C, x_3	
3′.	$\rightarrow C$:	z_D, z_C, z_1	[check $z_C = C$]
3″.	C :	decrypt z_1 as $\{\!\|z_2\|\!\}_{K_A^+}$	[check A's signature]
4.	$C\rightarrow$:	C, D, z_1, l	
4′.	$\rightarrow V$:	x_C, x_D, x_4, x_5	[check $x_4 = x_3$, $x_C = C$ and $x_D = D$]
5.	$(V)\rightarrow$:	D, C, x_5, r	
5′.	$\rightarrow C$:	z'_D, z'_C, z_3, z_4	[check $z_3 = l$ and $z'_C = C$]
5″.	C :	decrypt z_2 as $\{z\}_{z_4}$	

Table 4
FOO92: Extended protocol narration

with the corresponding public key (messages 1″, 2″ and 3″). In addition to verifying the administrators signature (message 2″) the voter must also ensure that the signed message was indeed his own ballot, unblinding of the signed ballot (message 2‴) will then result in a signed commitment of the vote in accordance with the rule (Unblind 2).

Modelling the anonymous communication channel (messages 3 and 5) is done by spoofing the source with a dummy name D. The publishing of the votes is done by sending each vote on the list to everyone on the ether - again the dummy D name can be used, now as the destination.

LySA specification. The extended narration can easily be translated into LySA by dividing the narration into 3 processes, one for each principal. The LySA specification of the protocol is given in Table 5. As we shall see shortly the analysis of LySA does not support rebinding of variables and new variables can only be introduced by input, decryption and unblinding. Therefore a small trick has to be used when a signature has to be verified but not removed: The recipient of the message has to decrypt the signature and then resign the content by using the same signature. This does not compromise the analysis as the signature of the message has already been verified. The trick is used in the model of both the voter and the counter (messages 3 and 4).

In the LySA specification we add annotations to all cryptographic operations as described earlier in Section 3. The sets of crypto-points $\mathcal{L}_{a1}$ and $\mathcal{L}_{c1}$ for the destination/origin assertions depend of the property that should be

analysed and we shall come back to those in Section 6. In all other cases the assertion sets will be equal to the entire set $\mathcal{C}$ of crypto-points no matter what property is being analysed - as an example digital signatures can be verified by anyone as it only requires knowledge of the public key.

In order to ensure that we analyse against the hardest attacker, the attacker should initially have knowledge of all the public keys. This is done in the LySA specification by sending these values in plaintext on the ether in parallel with the principals in the protocol.

$$
\begin{array}{ll}
 & (\nu_{\pm}\ K_V)\,(\nu_{\pm}\ K_A) \\
 & (\ (\nu\, v)\,(\nu\, r)\,(\nu\, b) \qquad\qquad /* \text{ Voter } */ \\
1. & \quad \langle V, A, V, \{\!|[\{v\}_r^{v1}[\mathsf{dest}\,\mathcal{C}\,]]_b^{v2}[\mathsf{dest}\,\mathcal{C}\,]|\!\}_{K_V^-}^{v3}[\mathsf{dest}\,\mathcal{C}\,]\rangle. \\
2'. & \quad (A, V;\ x1). \\
2''. & \quad \mathsf{decrypt}\ x1\ \mathsf{as}\ \{|;\ x2|\}_{K_A^+}^{v4}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in} \\
2'''. & \quad \mathsf{unblind}\ x2\ \mathsf{as}\ [;\ x3]_b^{v5}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in} \\
3. & \quad \langle D, C, \{|x3|\}_{K_A^-}^{v6}[\mathsf{dest}\,\mathcal{C}\,]\rangle. \\
4'. & \quad (C, D, \{|x3|\}_{K_A^-}^{v7}[\mathsf{dest}\,\mathcal{C}\,];\ x4). \\
5. & \quad \langle D, C, x4, r\rangle.0 \\
 & \\
1'. & |\ (V, A, V;\ y1). \qquad\qquad /* \text{ Administrator } */ \\
1''. & \quad \mathsf{decrypt}\ y1\ \mathsf{as}\ \{|;\ y2|\}_{K_V^+}^{a1}[\mathsf{orig}\,\mathcal{L}_{a1}\,]\ \mathsf{in} \\
2. & \quad \langle A, V, \{|y2|\}_{K_A^-}^{a2}[\mathsf{dest}\,\mathcal{C}\,]\rangle.0 \\
 & \\
 & |\ (\nu\, l) \qquad\qquad /* \text{ Counter } */ \\
3'. & \quad (D, C;\ z1). \\
3''. & \quad \mathsf{decrypt}\ z1\ \mathsf{as}\ \{|;\ z2|\}_{K_A^+}^{c1}[\mathsf{orig}\,\mathcal{L}_{c1}\,]\ \mathsf{in} \\
4. & \quad \langle C, D, \{|z2|\}_{K_A^-}^{c2}[\mathsf{dest}\,\mathcal{C}\,], l\rangle. \\
5'. & \quad (D, C, l;\ z3). \\
5''. & \quad \mathsf{decrypt}\ z2\ \mathsf{as}\ \{;\ z4\}_{z3}^{c3}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in}\ 0 \\
 & \\
 & |\ \langle K_V^+, K_A^+\rangle.0 \qquad /* \text{ Knowledge of the attacker } */ \\
 &)
\end{array}
$$

Table 5
FOO92 in LYSA-calculus

5 The Analysis

The analysis is specified as a Flow Logic in [6,7] for LySA (without blinding); here we shall only explain the general form of the judgements and refer the reader to the above papers for a more thorough presentation of the analysis. The analysis of the blinding constructs is given in details in Appendix A.

The aim of the analysis is to give a safe over-approximation of all possible messages communicated on the network, along with the possible value bindings of the variables. Furthermore the analysis will record all violations that there may be to the destination/origin annotations. In the analysis we assume perfect cryptography meaning that decryption can only be done using the correct key and similarly unblinding can only be done using the correct blinding factor.

The analysis of each term E will determine a superset of the possible values it may evaluate to. To do this we keep track of all potential value bindings to variables in a global abstract environment ρ:

$$\rho : \text{maps the variables to all values they may be bound to.}$$

The judgement for expressions takes the form:

$$\rho \models E : \vartheta$$

and expresses that ϑ is an acceptable estimate of the set of values that E may evaluate to in the abstract environment ρ.

In the analysis of a process P we focus on which values may flow on the network. In order to do this we keep track of all potential messages on the network ether in the abstract network environment κ:

$$\kappa : \text{includes all message tuples that may flow on the network.}$$

To obtain this information we make use of the abstract environment ρ, and the judgement for processes has the form:

$$(\rho, \kappa) \models P : \psi$$

with the error-component ψ:

$$\psi : \text{holds an over-approximation of the origin/destination violations.}$$

If $(\ell, \ell') \in \psi$ then something that was encrypted or blinded at crypto-point ℓ was unexpectedly decrypted or unblinded at ℓ'. The judgements for terms and processes are defined for LySA in [6,7] and in Appendix A we have extended these to include blinding.

To ensure that protocols are analysed for vulnerabilities against any attack possible they are analysed in conjunction with the Dolev-Yao attacker [13]. This attacker can perform the following actions: (1) Receive all messages sent on the ether; (2) Decrypt messages if he knows the key or unblind messages if he knows the blinding factor; (3) Construct new encryptions or blindings

from values he knows; (4) Send messages constructed from values he knows; and (5) Generate new values.

The attacker uses a special crypto-point $\ell_\bullet$ for encryption/decryption and blinding/unblinding. The knowledge of the attacker is collected in a special variable $z_\bullet$.

To better understand the analysis, consider the following flawed protocol with two principals A and B. In the protocol A generates a fresh key K and sends it in clear to B along with a message m which is symmetric encrypted under the key K (at crypto-point ℓ_A). Upon receiving the messages x_K and x, B decrypts x with the key x_K (at crypto-point ℓ_B).

$$\begin{array}{ll} ((\nu\, m)\,(\nu\, K)\,\langle A, B, K, \{m\}_K^{\ell_A}[\mathsf{dest}\ \mathcal{L}_A\,]\rangle.0 & /*\ A\ */ \\ | & \\ (A, B; x_K, x).\ \mathsf{decrypt}\ x\ \mathsf{as}\ \{; x_m\}_{x_K}^{\ell_B}[\mathsf{orig}\ \mathcal{L}_B\,]\ \mathsf{in}\ 0) & /*\ B\ */ \end{array}$$

The encryption at crypto-point ℓ_A is intended to be decrypted only at ℓ_B and correspondingly the decryption at ℓ_B should originate from the encryption at ℓ_A, hence we have the sets of crypto-points $\mathcal{L}_A = \{\ell_B\}$ and $\mathcal{L}_B = \{\ell_A\}$. The analysis of this protocol gives $\langle A, B, K, \{m\}_K^{\ell_A}[\mathsf{dest}\ \mathcal{L}_A\,]\rangle \in \kappa$ as this message is sent over the network. Since the attacker learns everything sent on the ether the analysis also gives $K \in \rho(z_\bullet)$ as well as $\{m\}_K^{\ell_A}[\mathsf{dest}\ \mathcal{L}_A\,] \in \rho(z_\bullet)$. As the attacker knows the key, he can decrypt the message $\{m\}_K^{\ell_A}[\mathsf{dest}\ \mathcal{L}_A\,]$ and hence the analysis yields the violation to the annotations $(\ell_A, \ell_\bullet) \in \psi$. The analysis also yields the violation $(\ell_\bullet, \ell_B) \in \psi$ as B does not know the key in advance, therefore the attacker can create a new key $K_\bullet$ and a new message $m_\bullet$ and send the message $\langle A, B, K_\bullet, \{m_\bullet\}_{K_\bullet}\rangle$ on the ether, which would be accepted by B.

The correctness of the analysis with respect to the operational semantics is formally established in [6,7] and extended to the blinding constructs in [2]. Hence it follows that if ψ is empty it guarantees that no violations of the annotations can exist and if a message sequence is not present in κ then it will never be sent on the network. This is of interest as the attacker may send any value he learns on the network, so if a value is not present in κ (or equivalent in $\rho(z_\bullet)$) then confidentiality of that value is guaranteed.

The analysis presented in this paper is implemented as an extension of the LySaTool [19] which already implements the analysis of [6,7].

6 Analysis Result

The FOO92 protocol is designed to run with many voters, one administrator and one counter. However as noted in [16] another interesting scenario is with many voters, many administrators and one counter. Our analysis covers

both of these scenarios and we have in particular analysed the protocol for an arbitrarily large number rather than than a fixed number of principals, as this is one of the strengths of the LySaTool. For readability in the following LySa specifications we have only one principal for each of the roles in the protocol (voter, administrator, counter and attacker). However as mentioned above the protocol is analysed with an arbitrary number of principals acting as voters, administrators and attackers such that interference within multiple principals is also considered.

In our scenarios the attacker is also an eligible voter. If the security properties are satisfied in this scenario they are obviously also satisfied if the attacker is not allowed to vote.

For each property we shall write the assertions that describes the property, that is we shall specify the sets $\mathcal{L}_{a1}$ and $\mathcal{L}_{c1}$ of crypto-points left unspecified in Table 5. We shall now discuss each of the properties of interest in turn and summarise the results at the end of the section.

Verifiability. A system is verifiable if the voters independently can verify that their vote has been counted correctly. A voter can be sure of this when he is certain that the counter has received the committed vote due to the assumption defined in Section 2, namely that the counter is trusted and that the voting will be dismissed if not all commitment keys for the votes published are received. This means that the verifiability property concerns authentication of the list published by the counter. The input (message 4') in the LySa specification must originate from the counter but in LySa we cannot add annotations to plaintext messages. We can however encode this assertion by symmetric encryption of the message from the counter with a key K, known also by the attacker, thereby not restricting the analysis. This addition to the LySa specification of the protocol is done as follows:

$$
\begin{array}{lll}
4'. & (C, D;\, x4). & /* \text{ Voter } */ \\
4''. & \mathsf{decrypt}\ x4\ \mathsf{as}\ \{\!|x3|\!\}^{v7}_{K_A^-}[\mathsf{dest}\,\mathcal{C}\,];\ x5\}^{v8}_{K}[\mathsf{orig}\,\mathcal{L}_{v8}\,]\ \mathsf{in} & \\
\vdots & & \\
4. & \langle C, D, \{\!|z2|\!\}^{c2}_{K_A^-}[\mathsf{dest}\,\mathcal{C}\,], l\}^{c4}_{K}[\mathsf{dest}\,\mathcal{C}\,]\rangle. & /* \text{ Counter } */ \\
\vdots & & \\
 & \mid \langle K_V^+, K_A^+, K\rangle.\mathbf{0} & /* \text{ Knowledge of the attacker } */
\end{array}
$$

By taking $\mathcal{L}_{v8} = \{c4\}$ we require that the publication of the list must originate from the counter. This specification has been analysed with the LySaTool together with the requirements that the sets $\mathcal{L}_{a1}$ and $\mathcal{L}_{c1}$ of crypt-points equal $\mathcal{C}$, the complete set of crypto-points. The LySaTool reports a potential attack, namely that publishing of the list can originate from the attacker: $(\ell_\bullet, v8) \in \psi$.

The description of FOO92 [14] requires that the publication is accessible

to all voters, but it does not say anything about the authentication of the list. In recent work [18] and in our interpretation of the protocol description, publication of the list is modelled by sending it on the ether, but as the analysis shows, this is not sufficient to guarantee verifiability.

This flaw has to the best of our knowledge not been reported in previous literature, but a simple amendment to the protocol can correct it. If the counter signs the list before publishing, as shown below, it turns out that the protocol is verifiable; that is, the analysis tool gives $\psi = \emptyset$. Note that the public key K_C^+ of the counter is published on the ether in order to let the attacker know it. The amendments of the protocol are:

$$
\begin{array}{lll}
4'. & (C, D; x4). & /* \text{ Voter } */ \\
4''. & \mathsf{decrypt}\ x4\ \mathsf{as}\ \{\!|\{\!|x3|\!\}^{v7}_{K_A^-}[\mathsf{dest}\,\mathcal{C}\,];\ x5|\!\}^{v8}_{K_C^+}[\mathsf{orig}\,\mathcal{L}_{v8}\,]\ \mathsf{in} & \\
\vdots & & \\
4. & \langle C, D, \{\!|\{\!|z2|\!\}^{c2}_{K_A^-}[\mathsf{dest}\,\mathcal{C}\,], l|\!\}^{c4}_{K_C^-}[\mathsf{dest}\,\mathcal{C}\,]\rangle. & /* \text{ Counter } */ \\
\vdots & & \\
 & \mid \langle K_V^+, K_A^+, K_C^+\rangle.0 & /* \text{ Knowledge of the attacker } */
\end{array}
$$

The remaining properties to be discussed below are validated for this version of the protocol.

Accuracy. As mentioned in the introduction, accuracy of a voting protocol is obtained when (1) it is not possible for a vote to be altered, (2) invalid votes must not be counted in the final tally and (3) all validated votes must count in the final tally.

We will begin with property (2). A valid vote is a committed vote signed by the administrator, which is obtained by the voter after the unblinding at crypto-point $v6$. In order for a vote to count in the final tally it has to be accepted by the counter at crypto-point $c1$. Hence we shall take $\mathcal{L}_{c1} = \{v6\}$ whereas the other sets $\mathcal{L}_{a1}$ and $\mathcal{L}_{v8}$ of crypto-points are set to $\mathcal{C}$.

Analysing the protocol yields a violation to the assertions: $(a2, c1) \in \psi$ ie. the blinded, signed ballots can be accepted by the counter without being unblinded first. Inspecting the protocol shows that this is indeed possible because the counter accepts any new ballot which is signed by the administrator without being able to verify the content of the ballot, hence the counter is not able to distinguish between a committed vote or a blinded committed vote. This means that the attacker is able to get an arbitrarily large number n of ballots accepted by the counter in the voting phase. He can do this by blinding his ballot $n-1$ times before having it validated by the administrator and then unblinding the signed result $n-1$ times, thereby obtaining n unique values all signed by the administrator. He will of course only be able to supply the counter with a commitment key for one of these values (the one unblinded

$n-1$ times) and therefore this attack will not violate accuracy but only force the voting to be disqualified.

Clearly the existence of this denial of service attack is not very satisfactory and it can be avoided by extending the specification. As the header of an encrypted value often contains information on the type of encryption used, we extend the LySA specification by adding a header BIT to the committed vote in the first message:

$$1.\quad \langle V, A, V, \{\!|[BIT, \{v\}_r^{v1}[\mathsf{dest}\,\mathcal{C}\,]]_b^{v2}[\mathsf{dest}\,\mathcal{C}\,]|\!\}_{K_V^-}^{v3}[\mathsf{dest}\,\mathcal{C}\,]\rangle.$$

And then step $3''$ only succeeds when the received messages is an unblinded message with BIT in the header:

$$3''.\quad \mathsf{decrypt}\ z1\ \mathsf{as}\ \{\!|BIT;\ z2|\!\}_{K_A^+}^{c1}[\mathsf{orig}\,\mathcal{L}_{c1}\,]\ \mathsf{in}$$

Additionally we let the attacker know the value BIT by sending it in plaintext on the ether, as this value merely models a standard header.

Analysing the protocol in a scenario where the attacker is allowed to vote we get the violations of the assertion; $(a2_0, c1) \in \psi$ and $(c2, c1) \in \psi$. The first violation $(a2_0, c1) \in \psi$ means that the attacker (indexed 0 in the analysis result) may get his validated vote accepted by the counter without unblinding it; this is equivalent to saying that the attacker can get his vote validated by the administrator without blinding it. The corresponding attack is as follows:

$$\begin{array}{lll}
1.\ DY \rightarrow A & : & V,\ sign_V(BIT, commit_r(v)) \\
2.\ A \rightarrow DY & : & sign_A(BIT, commit_r(v)) \\
3.\ DY \rightarrow C & : & sign_A(BIT, commit_r(v)) \\
4.\ C \rightarrow DY & : & sign_C(l,\ sign_A(BIT, commit_r(v))) \\
5.\ DY \rightarrow C & : & l,\ r
\end{array}$$

This attack shows that the attacker can choose not to blind his committed vote and hence be un-anonymous. However this does not violate that only valid ballots can be accepted by the counter as the attacker still needs to get his ballot validated by the administrator. Therefore we can extend the assertion $\mathcal{L}_{c1}$ to include $a2_0$.

Turning to the violation $(c2, c1) \in \psi$ we observe that the assumption, that the counter must have received all votes in the voting phase before commencing the publishing phase, contradicts that something encrypted at crypto-point $c2$ can be decrypted at $c1$ and hence we extend the assertion $\mathcal{L}_{c1}$ to include $c2$ as well. Now analysing the protocol with the sets $\mathcal{L}_{a1}$ and $\mathcal{L}_{v8}$ equal to $\mathcal{C}$ and $\mathcal{L}_{c1} = \{v6, a2_0, c2\}$ we obtain an empty ψ-component which means that no invalid votes can get accepted by the counter and therefore cannot count in the final tally.

For part (1) of the accuracy property we note that, as we assume perfect cryptography, it is not possible for a vote to be altered when it has been

validated. That the vote cannot be altered before validation can be observed from the possible variable bindings of $x3$ in the analysis result, $\rho(x3) = \{\{v\}_r\}$. Knowing that the analysis is an over-approximation we can be certain that only the unaltered vote can be validated and accepted by the voter, and we have that (1) is satisfied.

That all validated votes are counted, as required by part (3) of the accuracy property, relies on our assumptions and the previous parts. We know from part (2) that invalid ballots cannot be counted in the final tally. With the assumptions that the administrator only signs one ballot for each voter, that the number of accepted votes by the counter must be the same as the number of validated votes by the administrator and that every accepted ballot is unique, we can conclude that all validated votes must be counted in the final tally and thus that accuracy is satisfied.

Democracy. Democracy is obtained if (1) only eligible voters can vote and (2) they can only vote once.

Being able to vote (1) has two issues in the FOO92 protocol. Firstly, if and only if you are an eligible voter you must be able to get your ballot validated. Secondly, only validated ballots and all validated ballots must be accepted by the authority of the tallying, but this was already established by the validation of accuracy.

That only eligible voters are able to vote is modelled by $\mathcal{L}_{a1} = \{v3\}$, meaning that the vote being validated by the administrator does indeed originate from the voter it is being validated for. Analysis of the protocol with this assertion (and $\mathcal{L}_{c1}$ and $\mathcal{L}_{v8}$ equal to $\mathcal{C}$) yields an empty ψ-component and hence the first part of democracy holds.

That eligible voters are only allowed to vote once (2) can be validated if no replay attacks can be made on the first two messages sent from the voter (messages 1 and 3). No replay attack on the first message ensures that each voter can only get one valid vote, and no replay attack on the second message ensures that validated votes are only accepted once. According to the taxonomy of replay attacks [24] there are the following categories of replay attacks:

- From which session does the replayed message come from?

(1) Parallel/old/current session between same pair of players as in the attacked session.
(2) Parallel/old session between a different pair of players.

- Who is the recipient of the replayed message?

(a) Intended recipient.
(b) Different recipient (sender of the message or third party).

- Is the message used as intended in the protocol?

(i) Replayed message is used with intended purpose.
(ii) Replayed message is used with different purpose (type attack).

A replay attack can be classified with triple from the set $\{1,2\}\times\{a,b\}\times\{i,ii\}$. If a protocol is annotated properly the LySaTool finds attacks of the types $(2,*,*)$, $(*,b,*)$ and $(*,*,ii)$ where the entries $*$ can be chosen arbitrarily [20]. The only remaining replay attack is type $(1,a,i)$, which is when a message is re-sent to the intended recipient and used with intended purpose, in a parallel or new session.

A type $(1,a,i)$ replay attack on the validation from the administrator would mean that the same voter had two or more votes validated by the administrator, but this contradicts our assumptions. A type $(1,a,i)$ replay attack on the counter would mean that the counter accepted the same vote twice, but again this contradicts our assumption that all committed ballots are unique and that the counter only accepts one of each. Hence this type of replay attack is not possible according to our assumptions in Section 2 and as LySaTool does not report any violations to our assertions we can conclude that democracy is satisfied.

Fairness. A voting protocol is fair when early results from the voting cannot be obtained; in the FOO92 specification [14] this is defined as being before the opening phase. We shall model this by eliminating the opening phase in the LySa specification and claim that if the votes are then not in the knowledge of the attacker, fairness is obtained. Running the analysis in this scenario (with all $\mathcal{L}_\ell$ equal to $\mathcal{C}$) we do indeed observe that $v \notin \rho(z_\bullet)$ thereby validating the fairness property.

It is interesting to note that the fairness property is still satisfied even when the administrator and the counter conspire. This can be validated by letting the attacker know the secret keys for both the administrator and the counter; in this way he can act on behalf of both. As already mentioned we obtain $v \notin \rho(z_\bullet)$ also in this scenario.

Privacy. Privacy is obtained when no one can link any ballot to the voter who cast it. Validation of this property cannot be observed directly from the analysis result from LySaTool as the analysis result is an over-approximation of the values that the attacker may learn, but it does hold any information about how these values are related. In FOO92 the attacker learns both the vote and the identity of the voter, however only if the attacker can link them together, privacy is violated.

In [18] the privacy property is proven to be satisfied using equivalence theory. The proof also shows that an additional assumption is required, namely

that all voters should have finished the administration phase before the voting phase begins.

Summary. Table 6 contains the version of the FOO92 protocol that we have successfully validated using the LySaTool . The sets $\mathcal{L}_{a1}$, $\mathcal{L}_{c1}$ and $\mathcal{L}_{v8}$ have been selected individually to capture the property of interest as described in this section.

$$
\begin{array}{ll}
 & (\nu_{\pm}\ K_V)\,(\nu_{\pm}\ K_A)\,(\nu_{\pm}\ K_C)\,(\nu\ BIT) \\
 & (\ (\nu\ v)\,(\nu\ r)\,(\nu\ b) \qquad\qquad /* \text{ Voter } */ \\
1. & \quad \langle V, A, V, \{\!|[BIT, \{v\}_r^{v1}[\mathsf{dest}\,\mathcal{C}\,]]_b^{v2}[\mathsf{dest}\,\mathcal{C}\,]|\!\}_{K_V^-}^{v3}[\mathsf{dest}\,\mathcal{C}\,]\rangle. \\
2'. & \quad (A, V;\ x1). \\
2''. & \quad \mathsf{decrypt}\ x1\ \mathsf{as}\ \{\!|;\ x2|\!\}_{K_A^+}^{v4}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in} \\
2'''. & \quad \mathsf{unblind}\ x2\ \mathsf{as}\ [;\ x3, x4]_b^{v5}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in} \\
3. & \quad \langle D, C, \{\!|x3, x4|\!\}_{K_A^-}^{v6}[\mathsf{dest}\,\mathcal{C}\,]\rangle. \\
4'. & \quad (C, D;\ x5). \\
4''. & \quad \mathsf{decrypt}\ x5\ \mathsf{as}\ \{\!|\{\!|x3, x4|\!\}_{K_A^-}^{v7}[\mathsf{dest}\,\mathcal{C}\,];\ x6|\!\}_{K_C^+}^{v8}[\mathsf{orig}\,\mathcal{L}_{v8}\,]\ \mathsf{in} \\
5. & \quad \langle D, C, x6, r\rangle.0 \\
 & \\
1'. & |\ (V, A, V;\ y1). \qquad\qquad /* \text{ Administrator } */ \\
1''. & \quad \mathsf{decrypt}\ y1\ \mathsf{as}\ \{\!|;\ y2|\!\}_{K_V^+}^{a1}[\mathsf{orig}\,\mathcal{L}_{a1}\,]\ \mathsf{in} \\
2. & \quad \langle A, V, \{\!|y2|\!\}_{K_A^-}^{a2}[\mathsf{dest}\,\mathcal{C}\,]\rangle.0 \\
 & \\
 & |\ (\nu\ l) \qquad\qquad /* \text{ Counter } */ \\
3'. & \quad (D, C;\ z1). \\
3''. & \quad \mathsf{decrypt}\ z1\ \mathsf{as}\ \{\!|BIT;\ z2|\!\}_{K_A^+}^{c1}[\mathsf{orig}\,\mathcal{L}_{c1}\,]\ \mathsf{in} \\
4. & \quad \langle C, D, \{\!|\{\!|BIT, z2|\!\}_{K_A^-}^{c2}[\mathsf{dest}\,\mathcal{C}\,], l|\!\}_{K_C^-}^{c4}[\mathsf{dest}\,\mathcal{C}\,]\rangle. \\
5'. & \quad (D, C, l;\ z3). \\
5''. & \quad \mathsf{decrypt}\ z2\ \mathsf{as}\ \{;\ z4\}_{z3}^{c3}[\mathsf{orig}\,\mathcal{C}\,]\ \mathsf{in}\ 0 \\
 & \\
 & |\ \langle K_V^+, K_A^+, K_C^+, BIT\rangle.0 \qquad /* \text{ Knowledge of the attacker } */ \\
 &)
\end{array}
$$

Table 6
Amended FOO92 in LySA-calculus

7 Conclusion

In previous work [6,7,8,15] static program analysis has proved to be a simple and effective approach for validating confidentiality and authentication properties of key exchange protocols. In this paper we have successfully used

the very same approach for validating the somewhat different kind of security properties that apply for electronic voting protocols, namely verifiability, accuracy, democracy and fairness. We have studied one of the first voting protocols presented in the literature, the FOO92 protocol [14]. It is based on blinding signatures and part of the work presented here has been to accommodate this as a primitive in the LySa framework and the associated tool [2].

The original protocol specification is in the form of a classical protocol narration leaving out many details. As usual these are crucial when performing a formal analysis of the protocol - as is a clear formulation of the assumptions under which the protocol is analysed. Our initial analysis results for verifiability and accuracy pinpoints flaws in the protocol: we have identified a denial of service attack which could force the counter to repeatedly disqualify the voting process and we have identified a flaw which allows the attacker to forge the publishing of votes. The four security properties has subsequently been validated for the amended protocol given in Table 6.

Some of the security properties for the FOO92 protocol are studied formally by Kremer and Ryan in [18]; in particular the properties fairness, democracy (part 1) and privacy. They formalise the protocol in the applied π-calculus in order to use automated analysis with the tool ProVerif [5] to validate the first two properties. The third property, privacy, is proved by hand using equivalence theory; they prove that if the voter V_1 votes v_1 and voter V_2 votes v_2 it is observational equivalent to V_1 voting v_2 and V_2 voting v_1.

Future work includes applying the analysis to other voting protocols based on blinding signatures; also we would like to investigate the adaption of the approach to handing protocols with homomorphic encryptions [4,17].

Acknowledgement. This work is partly supported by the *LoST* project funded by the Danish Natural Science Research Council.

References

[1] Abadi, M. and A. D. Gordon, *A Calculus for Cryptographic Protocols The Spi Calculus*, Research Report 149, SRC - Systems Research Center (1998).

[2] Andersen, E. H. and C. R. Nielsen, "Static Analysis of Voting Protocols," M.Sc. thesis, IMM-2005-1218, Technical University of Denmark (2005), (to appear).

[3] Asano, T., T. Matsumoto and H. Imai, *A study on some schemes for fair electronic secret voting*, in: *The Proceedings of the 1991 Symposium on Cryptography and Information Security*, SCIS91-12A, 1991, (in japanese).

[4] Benaloh, J. D. C., "Verifiable Secret-Ballot Elections," Ph.D. thesis, Yale University (1987).

[5] Blanchet, B., *An efficient cryptographic protocol verifier based on prolog rules*, in: *CSFW '01: Proceedings of the 14th IEEE Workshop on Computer Security Foundations* (2001), p. 82.

[6] Bodei, C., M. Buchholtz, P. Degano, F. Nielson and H. R. Nielson, *Automatic Validation of Protocol Narration.*, in: *CSFW*, 2003, pp. 126–140.

[7] Bodei, C., M. Buchholtz, P. Degano, H. R. Nielson and F. Nielson, *Static Validation of Security Protocols*, Journal of Computer Security (2004).

[8] Buchholtz, M., "Automated Analysis of Security in Networking Systems," Ph.D. thesis, IMM-PHD-2004-141, Technical University of Denmark (2004).

[9] Buchholtz, M., *Implementing Control Flow Analysis for Security Protocols*, Technical Report WP6-IMM-I00-Int-003, DEGAS IST-2001-32072 (2004).

[10] Chaum, D., *Blind Signatures for Untraceable Payments*, Advances in Cryptology Proceedings of Crypto 82 (1982), pp. 199–203.

[11] Chaum, D., *Security without Identification : Transaction Systems to Make Big Brother Obsolete*, Communications of the ACM **28** (1985), pp. 1030–1044.

[12] Cranor, L. F. and R. K. Cytron, *Design and Implementation of a Practical Security-Conscious Electronic Polling System*, Research Report WUCS-96-02, Department of Computer Science, Washington University (1996).

[13] Dolev, D. and A. Yao, *On the Security of Public Key Protocols*, Proc. 22th IEEE Symposium on Foundations of Computer Science (1981), pp. 350–357.

[14] Fujioka, A., T. Okamoto and K. Ohta, *A Practical Secret Voting Scheme for Large Scale Elections*, Lecture Notes in Computer Science: Advances in Cryptology - AUSCRYPT '92 **718** (1992), pp. 244–251.

[15] Hansen, S. M., J. Skriver and H. R. Nielson, *Using Static Analysis to Validate the SAML Single Sign-On Protocol*, in: *WITS '05: Proceedings of the 2005 workshop on Issues in the theory of security* (2005), pp. 27–40.

[16] Herschberg, M. A., "Secure Electronic Voting Over the World Wide Web," M.Sc. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (1997).

[17] Hirt, M. and K. Sako, *Efficient receipt-free voting based on homomorphic encryption*, Lecture Notes in Computer Science **1807** (2000), pp. 539+.

[18] Kremer, S. and M. D. Ryan, *Analysis of an electronic voting protocol in the applied pi-calculus*, in: *Proceedings of the 14th European Symposium on Programming (ESOP'05)*, Lecture Notes in Computer Science **3444** (2005), pp. 186–200.

[19] *Webpage of LySaTool* (2004).
URL http://www.imm.dtu.dk/cs_LySa/

[20] Maidl, M., *Finding Replay Attacks In LySa* (2005), invited talk, LySa workshop.

[21] Milner, R., "Communicating and mobile systems: the π-calculus," Cambridge University Press, 1999, fifth edition edition.

[22] Naor, M., *Bit Commitment Using Pseudo-Randomness*, in: *CRYPTO*, 1989, pp. 128–136.

[23] Nielson, F., H. R. Nielson and C. Hankin, "Principles of Program Analysis," Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[24] Syverson, P., *A Taxonomy of Replay Attacks*, in: *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, 1994, pp. 187–191.

A Extending the Analysis for the LySA-Calculus with Blinding

The operational semantics and the analysis of LySA is provided in [6,7] (see also [8] for a thorough description). In this appendix we specify how to extend the semantics and the analysis to the new constructs.

Operational semantics. In Table A.1 we formalise the functionality of blinding in an operational semantics as already explained intuitively in Sections 1 and 3.

(Unblind 1)

$$\frac{\wedge_{i=0}^{j} \lfloor E_i \rfloor = \lfloor E_i' \rfloor \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\mathsf{unblind}\ [E_1, \cdots, E_k]^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ [E_1', \cdots, E_j'; x_{j+1}, \cdots, x_k]^{\ell'}_{E_0'}[\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P}$$
$$\rightarrow_{\mathcal{R}} P[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]$$

(Unblind 2)

$$\frac{\lfloor E_0 \rfloor = \lfloor E_0' \rfloor \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\mathsf{unblind}\ \{\!|E_1, \cdots, E_k|\!\}^{\ell^{sig}}_{E_0^{sig}}[\mathsf{dest}\ \mathcal{L}^{sig}]\ \mathsf{as}\ [; x]^{\ell'}_{E_0'}[\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P}$$
$$\rightarrow_{\mathcal{R}} P[\{\!|E_1, \cdots, E_k|\!\}^{\ell^{sig}}_{E_0^{sig}}[\mathsf{dest}\ \mathcal{L}^{sig}]/x]$$

Table A.1
Operational semantics for blinding, $P \rightarrow_{\mathcal{R}} P'$, parameterized on $\mathcal{R}$

We write $\lVert E \rVert$ for the term E with all annotations removed, and in (Unblind 1) and (Unblind 2) the condition $\lVert E_0 \rVert = \lVert E_0' \rVert$ models *perfect* blinding. Note that (Unblind 1) is identical to (Decr) in [6,7]. In (Unblind 2) the variable x is bound to the signed value $\{\!|E_1, \cdots, E_k|\!\}_{E_0^{sig}}$ along with it's assertions. The operational semantics $P \rightarrow_{\mathcal{R}} P'$ with the relation $\mathcal{R}$ is considered in two variants; the standard semantics with reduction relation ($\rightarrow$) discards the annotations and takes $\mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})$ to be universally true. The reference monitor semantics with reduction relation ($\rightarrow_{\mathrm{RM}}$) takes advantage of the annotations and takes $\mathrm{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) = \ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L}$. The semantics of the unblinding rules is given by the reference monitor semantics; thus, unblindings may only occur at crypto-points designated when the corresponding blindings were made and vice-versa, otherwise the configuration is stuck.

Analysis. As mentioned in Section 5 the analysis is specified by judgements of the forms $\rho \models E : \vartheta$ and $(\rho, \kappa) \models_{\mathrm{RM}} P : \psi$. The clauses defining the judgements may be found in [6,7] for the LySA calculus so Table A.2 only specify these for the blinding construct. The rule for the blinding term is very straightforward and identical to that of symmetric encryption. To produce the set ϑ, the rule for k-ary blinding finds the set ϑ_i for each term E_i, collects

$$\frac{\wedge_{i=0}^{k}\,\rho \models E_i : \vartheta_i \wedge \quad (\forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \vartheta_i \Rightarrow [V_1, \cdots, V_k]_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] \in \vartheta)}{\rho \models [E_1, \cdots, E_k]_{E_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] : \vartheta}$$

$$\frac{\begin{array}{l}\rho \models E : \vartheta \wedge \wedge_{i=0}^{j}\,\rho \models E_i : \vartheta_i \wedge \\ (\forall [V_1, \cdots, V_k]_{V_0}^{\ell'}[\mathsf{dest}\,\mathcal{L}'] \in \vartheta : \wedge_{i=0}^{j} V_i \,\mathsf{E}\, \vartheta_i \Rightarrow \wedge_{i=j+1}^{k} V_i \in \rho(\lfloor x_i \rfloor) \wedge \\ \qquad (\neg \mathrm{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell', \ell) \in \psi) \wedge \\ \qquad \rho, \kappa \models_{RM} P : \psi) \\ \wedge \\ (j = 0 \wedge k = 1 \Rightarrow \forall \{\!|[V_1, \cdots, V_{k'}]_{V_0}^{\ell'}[\mathsf{dest}\,\mathcal{L}']|\!\}_{V_0^{sig}}^{\ell^{sig}}[\mathsf{dest}\,\mathcal{L}^{sig}] \in \vartheta : V_0 \,\mathsf{E}\, \vartheta_0 \Rightarrow \\ \qquad \{\!|V_1, \cdots, V_{k'}|\!\}_{V_0^{sig}}^{\ell^{sig}}[\mathsf{dest}\,\mathcal{L}^{sig}] \in \rho(\lfloor x_1 \rfloor) \wedge \\ \qquad (\neg \mathrm{RM}(\ell, \mathcal{L}', \ell', \mathcal{L}) \Rightarrow (\ell', \ell) \in \psi) \wedge \\ \qquad \rho, \kappa \models_{RM} P : \psi)\end{array}}{\rho, \kappa \models_{RM} \mathsf{unblind}\ E\ \mathsf{as}\ [E_1, \cdots, E_j; x_{j+1}, \cdots, x_k]_{E_0}^{\ell}[\mathsf{orig}\,\mathcal{L}]\ \mathsf{in}\ P : \psi}$$

Table A.2
Analysis of terms and processes for blinding: $\rho \models E : \vartheta$ and $(\rho, \kappa) \models_{RM} P : \psi$.

all k-tuples of values $(V_0, \cdots, V_k)$ taken from $\vartheta_0 \times \cdots \times \vartheta_k$ into values of the form $[\![V_1, \cdots, V_k]\!]_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}]$ and requires these values to belong to ϑ.

The rule for the unblinding process consists of two parts. The first part is similar to the rule for symmetric decryption, so we will not describe it further. If the process has the form $\mathsf{unblind}\ E\ \mathsf{as}\ [\![; x_1]\!]_{E_0}^{\ell}[\mathsf{orig}\,\mathcal{L}]\ \mathsf{in}\ P$ the second part of the rule is used. For each value that is signed and blinded; $\{\!|[\![V_1, \cdots, V_{k'}]\!]_{V_0}^{\ell'}[\mathsf{dest}\,\mathcal{L}']|\!\}_{V_0^{sig}}^{\ell^{sig}}[\mathsf{dest}\,\mathcal{L}^{sig}] \in \vartheta$ it is checked whether the value V_0 is included into ϑ_0, here the faithful membership E for matching ignores annotations. If the check is successful then the value $\{\!|V_1, \cdots, V_{k'}|\!\}_{V_0^{sig}}^{\ell^{sig}}[\mathsf{dest}\,\mathcal{L}^{sig}]$ must be contained in x_1, additionally the ψ-component must contain (ℓ, ℓ') if the destination/origin assertions might be violated.

The implementation of the analysis is carried out according to [9] and the complete analysis, the implementation and a proof for soundness of the analysis is presented in [2].