Component Based Electronic Voting Systems

David Lundin*

*University of Surrey, Guildford, Surrey, GU2 7XH, UK, d.lundin@surrey.ac.uk

***Abstract*—An electronic voting system may be said to be composed by a number of components, each of which has a number of properties. One of the most attractive effects of this way of thinking is that each component may have an attached in-depth threat analysis and verification strategy. Furthermore, the need to include the full system when making changes to a component is minimised and a model at this level can be turned into a lower-level implementation model where changes made can cascade to as few parts of the actual implementation as possible.**

I. Introduction

It appears that each researcher in the field of Electronic Voting Systems contributes to some particular aspect but rebuilds the whole system when they wish to implement this rather specific contribution. The idea presented in this paper is that in order to build an e-voting system we simply add certain distinct pieces together - and in order to improve on a particular system we swap one distinct piece for another that fits into the same slot. In short, we are proposing that we start thinking about electronic voting systems as being component based.

A benefit of this thinking is that for each component slot, i.e. a place in a layer where a component of the system can be slotted in, it is possible to define the method of assessing the computational complexity of that component as well as performing a threat analysis. Similarly, it we agree that all components of an electronic voting system should be verifiable and/or auditable then it is possible in this configuration to define for each component a method of verification or audit. When an author then makes changes to one or more components it is possible to in effect "re-run" checks on those components or to employ the same verification method on a different component.

A. *Domains captured*

As the reader goes through the description of the different layers, she may realise that the components in those layers are not strictly components from a computer systems design perspective. Instead, the components capture the full spectra of the design and implementation of an electronic voting system — in other words they capture the following domains:

- Computer system domain
- Human (user/voter) domain
- Legislative domain

We do not make distinctions between components in the hierarchy that we propose, that is to say that we do not treat components from different domains differently. To a developer of electronic voting schemes this will be perfectly natural as her overview of the system is total. When she considers all the components then they must all fall into place and completely make up the (correct) system. In fact, it is the duty of the developer to ensure that a component that is put forward does fulfill all the requirements on that component.

Others may look differently on the component based electronic voting system. A programmer (implementer) may look only on those components that are implementable in software. Similarly someone who might be consulted on the legal implications on the configuration of an electronic voting system may only consider components from the legislative domain. Both these non-developer persons are examples of people who must be able to trust that the system as a whole depends on each of their respective components and that the configuration of their components are reflections of the requirements and conform to the restrictions of those same components.

It is encouraged that a developer of electronic voting systems that conform to the proposed component based methodology considers the subset of components that are to be implemented in code and makes available to programmers some model that binds these together in some lower-level modelling scheme — this is out of the scope of this paper but seems fairly straightforward in any (favourite) modelling language.

B. *Cascading changes*

When a system is considered to be component based we believe that changes can easily cascade down the different layers, all the way down to the implementation. In a trivial sense this might simply be that when the developer changes the structure of a component this automatically becomes a list of changes for the implementer to change in the actual code. This is likely to be the scope of other papers in the modelling domain. However, it is easy to see that changes made to one component will only result in changes to the implementation of that same component and not to surrounding or distant components, restricting the work needed to implement the changes.

II. Component hierarchy

We suggest that an electronic voting system is made up of parts from four comparatively separable and distinguishable layers, each of which builds on the services provided by a lower level layer. We propose to name these layers in the following fashion:

1) Human layer
2) Election layer
3) Computational layer

Layer	Components
Human layer	Voter registration Ballot form configuration Polling station layout and management Verifiability front-end
Election layer	Election method Election management system Voter-ballot box communication channel
Computational layer	Cryptography scheme Anonymisation strategy Tallying procedure
Physical layer	Hardware authentication method Publishing strategy Transfer method

TABLE I
LAYERS AND COMPONENTS

4) Physical layer

The physical layer enables the reading in of voter choices and the transfer and publishing of the same. The computational layer contains that which is chiefly encapsulated in software and which relies on the physical services of the lower level. The election level encapsulates all those options and configurations relating to the election system being implemented by the two lower levels. Finally the human layer deals with those aspects of the electronic voting system which face the voters.

The layers and components are shown in Table I. We will now go through these layers from the bottom up so as to first provide a solid foundation and then explain how all the aspects of electronic voting systems may fit into this model.

A. Physical layer

The physical layer is of course the most basic layer as it supports all other layers with a physical infrastructure to facilitate different hardware based aspects of the system. We divide the physical layer into the following components:

1) Hardware authentication method: The different physical components of the electronic voting system must be identified using some authentication strategy, for example an asymmetric key pair and a public key infrastructure (PKI) with established trust among voters.

2) Publishing strategy: Voter verifiable electronic voting systems depend on the information that voters need to perform the verification being delivered to them in a way that they can trust. An observation is that a single source may be an unreliable publishing strategy but a combination of web sites and newspapers and other independent outlets may be more reliable.

3) Transfer method: All electronic voting systems capture some information from the voters and transfer it to some repository, be it central or distributed, before the information is interpreted and tallied. In this slot fits methods for transporting such digital information from polling stations to such repositories, for example Secure Socket Layer (SSL) over the Internet or storage on flash drives that are manually distributed.

B. Computational layer

In the computational layer are defined all the components that are performed implicitly by software. These include the following components:

1) Cryptography scheme: The cryptography scheme employed probably underpins much of the operation of other components in this layer so there is an intra-layer dependence. However, the cryptography scheme is more easily defined in its own component after which the other components of this layer (and others that may rely on it, although restricting the use of the cryptography scheme to the computational layer only would greatly simplify the definition of a scheme in this model) may refer to it. One example of this may be where the cryptography scheme employed is Elgamal and re-encryption mix networks are used as anonymisation strategy in that component.

2) Anonymisation strategy: We believe that electronic voting schemes that are both “receipt free” and voter verifiable must use some anonymisation strategy to break the link between an encrypted receipt and the plain text vote. This strategy is captured in this component although it most likely is heavily dependent on the cryptography scheme defined in the previous. However, in some cases the anonymisation scheme may rely on “any” cryptographic scheme and so changing the cryptography scheme component does not necessarily interfere with the anonymisation strategy component. One example of where this is true is where the anonymisation strategy is a re-encryption mix network. It is logical that the cryptography scheme may be Elgamal or Paillier and one scheme may be removed and the other slotted in without this requiring the anonymisation strategy to change.

3) Tallying procedure: The tallying procedure component is simply an encapsulation of the procedure and software that performs the tallying of the decrypted votes. This may seem like a trivial component at first glance but when the component properties that we introduce below are considered it appears that a verification strategy and computational complexity analysis for example is done better across this component than across some external “election administration” software.

C. Election layer

This layer is very much derived from the laws that govern the election. It seems likely that when this layer is implemented it may result in some components not being turned into code and others may be external software. As an example of the latter we can take that the implementation of a component may actually be a formal specification of a piece of software that can successfully audit an election. This is then published and any number of interested parties may write their own piece of software.

The election layer consists of the following components:

1) Election method: This component contains a specification of the election method that is used and into it is placed simply as much information about the election that is available. If the model is scrutinised as a whole it may be beneficial to have in one part of it the specification of the election at such a high level.

2) Election management system: All schemes require an election to be set up by some clerks or civil servants — or politicians or some trusted third party. This component is, much like the previous, incorporated into the model for completeness although it does require to communicate with other components.

3) Voter-ballot box communication channel: This may seem like a much more low-level component but in fact it is included in this layer to enable the high-level definition of the secrecy of the election. In other words it is possible to describe here the full procedure that the voter goes through to cast a vote in a way that is understood not only by the developers of the system but also the general public. This definition may then be used as reference when other components at lower levels in the model are composed.

D. Human layer

Although some components in the previous layer have become human readable and in fact only serve to further the understanding of the lower layers (or, arguably, define the lower layers in a human readable way which then cascades down) this layer deals with all those aspects of the electronic voting system that are facing the voter. Therefore the components we expect to find in this layer are:

1) Voter registration: The procedure for and timing of voter registration along with the criteria that makes a voter eligible to register are all enshrined in the law relevant to the use of the implementation of the modelled system. They are included in the model in this component so as to provide completeness. As mentioned in Section I the completeness of the system is entrusted to the electronic voting system developer and so a change in this component must trigger any necessary changes to other components — and these may of course be components that are eventually implemented in code.

2) Ballot form configuration: The ballot form configuration is very important in any electronic voting system and it seems likely that this component has links to other components, for example the cryptography component in the computational layer.

3) Polling station layout and management: This is another component that is directly derived from the applicable national and regional legislation. If the component is precise, by which we mean it is directly derived from legislation which sets out exactly the layout and management, then this may dictate the configuration of other components in the model. If it on the other hand is permissive then this would imply that only a few guidelines are given and that if necessary the layout and management of polling stations may be adapted to suit the electronic voting system.

4) Verifiability front-end: To present a unified verifiability front-end to voters and concerned groups, this components acts to capture the requirements of such a front-end at a high level and thus restrict other components that must comply therewith.

E. Component interaction

As has now become abundantly clear it is impossible to make all components of an electronic voting system fully autonomous — and we simply are not striving to do that. Instead, by making each component as distinct and autonomous as possible and then providing links between components we can reach some compromise between all the good properties brought by a component based model and the necessity of component interaction.

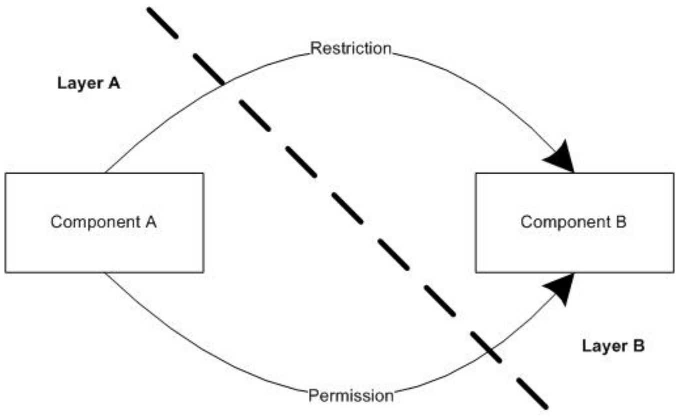

Fig. 1. Component restriction and permission

The links we propose are not communication links as such, in fact we regard the communication between different implemented components to be modelled at a lower level. This is supported by the fact that only some components at this level of abstraction are implemented and only some of those communicate — in fact some of these components become entangled in an implementation, for example the cryptography component which is used in an implementation of the anonymisation strategy component in the computational layer.

The component links that we propose are of only two kinds and each link is a mono directional vector. The link carries some fact attached to it and it either defines a restriction by one component on another or a permission given by one component to another.

The links are fairly simplistic and can easily be illustrated in a graph, and example of which is shown in Figure 1. If the underlying fact or facts are coded in some way (perhaps just a unique numerical reference) within the imposing component then the link can easily be shown in a graph simply using an arrow with a name corresponding to that code (reference).

III. Component properties

The components in this model of electronic voting schemes have a number of properties that can be defined much easier than similar properties for the system as a whole. The properties that distinguish one component from another include:

A. Layer and slot into which it fits

The hierarchy of components is determined before the model is set up for a particular electronic voting system, and logically declaring the different parts of “a” system before filling in those slots with a particular implementation does seem to restrict the developer in some sense but on the other hand it does result in a system that can be more easily explained and developed. From a verifiability perspective such a system could also have associated with each component slot methods for checking that those components are correct.

B. Origin, requirements and constraints posed by the slot

This set of properties describe the requirements on the component which fits into the slot. As the requirements come from the model, rather than being made up on the fly by the developer of the component, it is possible to change between two components.

C. Verification strategy

In our opinion the electronic voting system research community can no longer concern itself with systems that are not fully verifiable. In fact, we should reserve the term “electronic voting system” for something that is transparent and verifiable. Thus the requirement on every single component of the component based electronic voting system is that there exists some verification strategy — again, in our opinion, a combination of voter verifiability and public verifiability should be able to cover all components.

This may be one of the most attractive aspects of the component based methodology. By providing some overview of the verifiability of each component the developer can show that the full system is verifiable. By slotting one component out and another in, the same verification strategy can still be in place, reducing the work needed to compose a new component for a particular slot.

As an example of this we can mention the anonymisation strategy component in the computational layer. Let us say that a decryption mix network makes up this strategy (using the cryptography defined in the cryptography scheme component). The verification strategy on this component is a variant of zero knowledge proof. If a developer slots this anonymisation strategy component out and replaces it with a re-encryption mix network instead, the verification strategy remains the same.

D. Location in the system (authority-close or voter-close)

A component has a particular position in the system as a whole and this may be described by whether it is close to the voter and controlled by her or if it is close to an election authority and controlled by them.

One example of this might be the ballot form component in different versions of Prêt à Voter. In Prêt à Voter 2005 [1] the ballot form is quite authority-close in that it is set up by an election authority before the election and this authority holds all information needed to decrypt the receipt. To combat the chain voting attach [6] Prêt à Voter 2006 [7] introduces re-encryption mixes whereby the creation of the decryption information is distributed among a number of trusted parties. The form is printed on demand by the voter in the booth and can thus be regarded to be voter-close.

E. Threat analysis

Each component, just as it has a verification strategy, can have a threat analysis attached to it. This analysis holds for any component that is slotted into the same place. The output of such a threat analysis is a number of requirements on the component.

F. Computational complexity analysis

Seemingly more dependent on the particular component, this property is part of the computational complexity analysis of the complete system. By decomposing the system into smaller parts this analysis also becomes much easier, aiding the implementation onto hardware.

IV. EXAMPLES

In this section we apply this model to two already existing schemes, in the first instance Punchscan [4], [2], [3] and in the second Prêt à Voter [1], [5], [6], [7]. These simplistic decompositions are included to illustrate that the model would be applicable to schemes that already exist and that the further development of those schemes could be done in a component based methodology.

A. Punchscan decomposed

Punchscan relies heavily on a central election authority to set up, manage and guarantee the safety, security and reliability of the voting system. The anonymisation strategy can be audited using a zero knowledge proof method.

A simple decomposition of Punchscan is shown in Table II. A Punchscan developer can of course make the decomposition much, much more detailed — at which point it becomes useful. Perhaps such a decomposition of one of the current electronic voting schemes is a suitable Master of Science project. When the decomposition has been made of course the continued development of the system is performed in the component oriented methodology.

B. Changes to cryptography component in Prêt à Voter

In the original Prêt à Voter [1] the cryptography scheme component is RSA, providing services to the anonymisation strategy component, which consisted of a decryption mix network. This system suffered from the authority knowledge (or chain voting) problem [6] and thus the next version [7] slotted in the RSA cryptography component and slotted in an Elgamal cryptography component. This new component was able to support a re-encryption mix network in the anonymisation strategy component and this in turn meant the the ballot form could be printed on demand in the booth, affecting the ballot form component.

C. Using Punchscan style ballot form in Prêt à Voter

Quite interestingly we can show in this section that the “traditional” Prêt à Voter [1] ballot form is completely interchangeable with the “traditional” Punchscan [3] ballot form. The ballot form is thus a prime candidate for a component in the electronic voting system decomposition.

The ballot form in Prêt à Voter, shown in Figure 2, has a randomly ordered candidate list in the left of two columns. The right column consists of a grid where the voter marks her choice(s). Below this grid is printed the *onion* which encapsulates the order of the candidate list under a number of cryptographic layers. When the form is torn along a vertical perforation between the two columns and the left column

Layer	Component	Punchscan
Human layer	Voter registration	The voter returns a form delivered to her house by local government. When she is registered a polling card is delivered by post.
	Ballot form configuration	The ballot form is made up of two pages, the first has holes through which the second can be seen. On the first page is printed the races and the candidates in those races. Next to each candidate can be found a symbol which matches a symbol on the second page, visible through one of the holes. To vote the voter marks both pages using a bingo marker ("dauber") over the second page symbol which corresponds to the candidate she wishes to vote for.
	Polling station layout and management	When the voter enters the polling station her name is checked against the register. She identifies herself using the polling card.
	Verifiability front-end	Voters can check their receipts on a web site as well as in the local media. The audit of the full election can be viewed online or through media reports.
Election layer	Election method	The local representative is elected by first past the post and thus each voter gives her vote to a single candidate.
	Election management system	Before election day the election authority which oversees the election calls a meeting of election officials and representatives of the different political parties. In this meeting the election is set up on a diskless workstation running trusted, audited software. The output of this process is a database which is kept secret by the election authority for the purpose of decrypting votes after the election together with instructions to a print company which will print the ballot forms.
	Voter-ballot box communication channel	The voter is allowed to fill out the ballot form in the privacy of a voting booth. Before leaving the booth she is able to create the encrypted receipt, making it hard for anyone to capture information from both layers of the ballot form. The encrypted receipt is scanned (with the aid of poll station workers) and transmitted electronically.
Computational layer	Cryptography scheme	Although no cryptography as such is used in Punchscan, the vote is hidden behind a number of random translations that are in turn captured in a number of interlinked *decryption tables*. There exists one translation for each of the two pages in each decryption table (the translation may be "no translation") and a number of decryption tables may be linked together.
	Anonymisation strategy	The decryption tables are set up in advance such that when a ballot form is made into an encrypted receipt, the application of all the translations for that form in all decryption tables reveals the voter's intention. In order to anonymise a plaintext vote the full batch of encrypted receipts are secretly mixed after the application of each decryption table. The decryption is audited by making the authority commit to the decryption of a batch and then challenge it to reveal a number (but not all) links between the input and output batches. By not auditing all links the full link from an encrypted receipt through to a plaintext vote is broken at some stage.
	Tallying procedure	The encrypted receipts are published to the web, the election authority applies its decryption tables and anonymisation strategy and publishes the result of each step. When the plaintext votes appear at the end the authority, and anyone wishing to check the result, can perform the tally.
Physical layer	Hardware authentication method	Each polling station machine is issued a cryptographic key pair and an identity.
	Publishing strategy	All encrypted receipts received by the central repository are published, in some predetermined batch mode, to a publicly accessible read-only online resource.
	Transfer method	Communication between the polling station and the central repository is encrypted using SSL.

TABLE II
LAYERS AND COMPONENTS IN PUNCHSCAN

shredded, the encrypted receipt remains. The onion value uniquely identifies the ballot form.

The Punchscan ballot form consists of two pages, shown in Figure 3, on the first of which is printed the candidate list in the canonical order. Next to each candidate on this page is also printed a symbol which corresponds to the same symbol shown, through holes in the first page, on the second page. The voter marks her choice using a bingo marker/dauber. This makes a mark on both pages at the same time and when one page is randomly selected and the other shredded, what remains is an encrypted receipt. The ballot form serial number printed in the top-right corner uniquely identifies the ballot form.

In both schemes the encrypted receipt is scanned and transmitted digitally. We can now describe both these forms

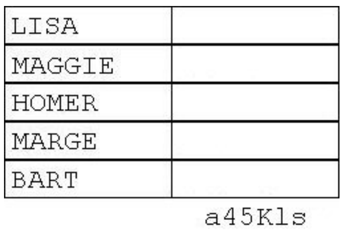

Fig. 2. The Prêt à Voter ballot form

with a component with the following configuration:

1) Layer and slot into which it fits: Human layer, Ballot form configuration slot.

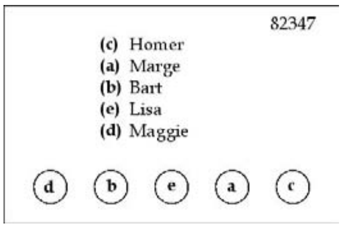

Fig. 3. The Punchscan ballot form

2) Origin, requirements and constraints posed by the slot: The ballot form must list the candidates in the race for which it is valid on one half of the form. The other half must accept the voter's intention. If the two halves are separated one remaining half must not reveal the candidate(s) for which the vote was cast but must be decryptable to reveal this information.

The form must be printed on paper, security paper is permitted. It must be scannable and shreddable.

3) Verification strategy: A reference printed on the form must uniquely identify it so that the voter may search for it in an online resource after the close of the election. During the election the voter may also use this reference to audit a form which will not be used for voting.

4) Location in the system (authority-close or voter-close): The ballot form is created in advance by an election authority, distributed under guard to safeguard the secret of the form and picked out at random by the voter.

5) Threat analysis: The ballot form must never be seen by anyone other than the voter before one half is removed to form the encrypted receipt as this may remove the system's coercion resistance [6]. The ballot form is therefore to be distributed within an envelope that some physical procedure must guarantee that only the voter may open within the booth.

6) Computational complexity analysis: Creating the ballot form involves creating some decryption information, applying it a number of times over and then printing the form.

V. Discussion

This section provides a quick overview of some of the caveats with the component based model that we have foreseen.

A. Impossible to have strict boundaries between components

Defining electronic voting schemes as component based systems provides researchers with the opportunity to focus development on a particular component or to compare two different components that fit the same slot to determine which is best. However, when the component based model is turned into an implementation there may be complications. If an implementation is made of one particular component based model and one or more components are changed in the model then cascading these changes to the implementation may not be trivial. For example, the implementation of the mixing strategy may be heavily dependent on the cryptography scheme used and a change of the latter most likely results in the change of code in the earlier.

B. Restrictions on the developer

Some may wish to make the developer of electronic voting systems completely without bounds — but this also implies that the developer will have no framework and the structure of the system developed will not be easily decomposed by others who wish to further the development. This in turn may seem like an academic-only exercise — but we are academics.

C. Requirements must come from the model

A developer of a component may look at the technical contribution of that component and make up the requirements of the component from that. This is easy to do but then suddenly the resulting system does not contain components that may be re-used or changed easily. Therefore it is important that the electronic voting system developer looks at the system as a whole and fully defines the requirements on each component before proceeding to create those components.

VI. Summary

We have presented a first overview of a component based methodology for developing electronic voting systems. By not changing the complete system we hope that developers may be encouraged to look in depth on a particular component or set of components, providing a complete threat analysis as well as verification strategy for each.

A. Acknowledgements

Many thanks to the WOTE reviewers for their comments and to Zhe Xia for interesting conversations.

References

[1] D. Chaum, P. Ryan, and S. Schneider. A practical voter-verifiable election scheme. *Proceedings of the tenth European Symposium on Research in Computer Science (ESORICS'05)*, pages 118–139, 2005. LNCS 3679.

[2] K. Fisher, R. Carback, and T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *PRE-PROCEEDINGS*, pages 19 – 29. IAVoSS Workshop On Trustworthy Elections, 2006.

[3] S. Popoveniuc and B. Hosp. An introduction to punchscan. In *PRE-PROCEEDINGS*, pages 27 – 34. IAVoSS Workshop On Trustworthy Elections, 2006.

[4] Punchscan. Punchscan website. http://www.punchscan.org.

[5] P. Ryan. A variant of the chaum voter-verifiable scheme. *Proceedings of the 2005 Workshop on Issues in the Theory of Security*, pages 81–88, 2005.

[6] P. Ryan and T. Peacock. Prêt à voter: a system perspective. *Technical Report of University of Newcastle*, CS-TR:929, 2005.

[7] P. Ryan and S. Schneider. Prêt à voter with re-encryption mixes. *Proceedings of ESORICS*, 2006. LNCS.