2012 European Intelligence and Security Informatics Conference

Structured Documents: Signatures and Deception

Aaron Hunter
Thompson Rivers University, and
British Columbia Institute of Technology
British Columbia, Canada
aaron_hunter@bcit.ca

***Abstract*—Much of the information exchanged between agents over a network is encapsulated in XML documents. An XML document has a tree structure, and the meaning of the document can be understood in terms of a set of label-value pairs. The content of a document is often secured through digital signatures applied to different sections, while the document is passed between several agents. In this paper, we illustrate that this process is insecure in the sense that a malicious agent can deceive an honest agent to hold beliefs that are untrue. We provide a formal framework for analyzing the security of structured documents, based on the implicit epistemic impact that a signed document will have on a recipient. This kind of analysis can provide significant insight into deception and fraud detection.**

I. Introduction

Information exchanged over a network is often encapsulated in *documents*, which provide structure for the information and provide constraints over admissible utterances. In practice, these are generally XML-documents, where the internal structure is a tree. It is possible for a malicious intruder to exploit this tree-structure to deceive an honest agent. In this paper, we introduce a framework for analyzing the stucture of a document with an eye towards detecting deception, and we illustrate that our framework can be used to automatically identify documents that are susceptible to attack.

Our framework is based on formal methods developed for knowledge representation and reasoning. The basic idea is to make the beliefs of communicating agents explicit, which allows us to consider how these beliefs can be manipulated by a malicious intruder. In previous work, we have focused on the manipulation of beliefs through an exchange of encrypted messages [8]. However, the notion of a "message" is not sufficient for analyzing most communication. Instead, we need to consider documents in which certain elements may be signed and manipulated in a subtle manner.

This paper makes several contributions to existing literature. The main contribution is the development of a new application of belief logics and action formalisms in information security. We illustrate that the dynamics of belief play a significant role when we consider commitments and deception in the context of signed documents. We introduce a simple formal representation of a digitally signed document. We present a precise definition of a flawed document, in terms of the belief manipulation through an iterative signing procedure. Finally, we establish some preliminary results, illustrating the potential utility of formal models of reasoning for analyzing deception.

II. Preliminaries

A. Documents

For the present paper, a *document* is an XML-like tree of label-value pairs. We are not concerned with the exact syntax of XML, we are simply concerned with the fact that structure is imposed on the data. We will provide a formal definition of a document in section 3. For motivation, one important kind of XML document is a *form*. Forms serve several purposes in practice: they provide an interface for data entry, they impose a structure on data, and they allow an agent to commit to certain data through a signature [4]. Traditionally, the security of the information entered on a form has been guaranteed by the physical properties of paper. However, the expenses related to paper forms in business have been well-documented [3], and there has been a great deal of interest in replacing paper forms with electronic forms. In the case of digital forms, the security must be guaranteed through the structure of the form and the use of digital signatures.

We use forms as the main example of an document in this paper, because XML standards have been developed for representing digital forms. The formal representation employed in this paper is inspired by the XFDL standard that is used to encode the appearance of a form in IBM's Lotus Forms software [5]. XML trees representing forms in this standard have a very simple structure. Each field is a node, and the children of the field node provide information about the field. The children of a field node might indicate many display options, including the location of the field on the page, the text beside the field, or the font size to be used for the text. The meaning of a digital signature depends on which nodes have been signed.

B. Logic and Belief

Assume an underlying set of propositional variables $\mathcal{F}$. A *state* is an interpretation of $\mathcal{F}$, that assigns each fluent symbol a truth value. Informally, a state represents a possible configuation of the world. We introduce some basic terminology and notation related to epistemic logic. For a complete introduction to the subject, we refer the reader to [7]. For representing beliefs, we will use the basic modal logic K_n with unary modal operators $B_1, \dots, B_n$. A modal formula of the form $B_i\phi$ is interpreted to mean that agent i *believes* the formula ϕ to be true.

A *belief state* is a set of states representing the worlds that some agents believes to be possible. A belief change operator

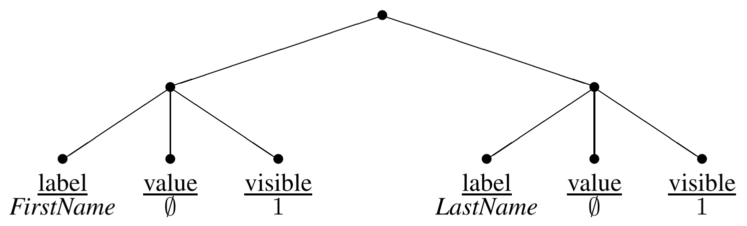

Fig. 1. A Simple Document Tree

is a function $*$ that takes a belief state κ and a formula ψ as input, and it returns a new belief state. For example, an AGM belief revision operator is a belief change operator [2]. A *belief manipulation problem* consists of an initial belief state κ, a belief change operator $*$, a set of formulas Φ and a goal ψ. A *solution* to the problem $\langle \kappa, *, \Phi, \psi \rangle$ is a seqence $\phi_1, \ldots, \phi_n \in \Phi$ such that $\kappa * \phi_1 * \cdots \phi_n \models \psi$. If we think of the formulas in Φ as messages, then a solution to a belief manipulation problem is just an argument that justifies ψ.

The notion of *authentication* refers to the process in which an agent convinces another to hold certain beliefs about identity [9]. An authentication protocol is secure if every solution to the underlying belief manipulation problem entails that the manipulator has a certain identity. It is well known that this can be analyzed through a formal analysis of changing beliefs of protocol participants [1], [6]. We will see that belief manipulation becomes more subtle to detect in the case of signed documents.

III. Formal Framework

A. Abstract Document Trees

We are concerned with documents that can be logically parsed into a tree of label-value pairs. However, to simplify the discussion, we restrict attention to a very simple class of trees. Let $\mathcal{O}$ be a distinguished set of *option types*, and let $\mathcal{V}$ be a function that maps each $O \in \mathcal{O}$ to a set $\mathcal{V}(O)$ of *option values*. A pair $\langle \mathcal{O}, \mathcal{V} \rangle$ is called a *document scenario*. Recall that a *rooted tree* T is a connected graph with no cycles, and a designated root vertex r_T. In the following definition, we let $T(k)$ denote the set of nodes in T that are connected to the root by a path of length k.

Definition 1: A *document tree* is a pair $\langle T, \Phi \rangle$ satisfying the following conditions:

1) T is a rooted tree of height 2.
2) For each $v \in T(2)$, $\Phi(v) = (O, V)$ for some $O \in \mathcal{O}, V \in \mathcal{V}(O)$.
3) If $\Phi(v_1) = (O_1, V_1)$ and $\Phi(v_2) = (O_2, V_2)$, then $O_1 = O_2$ iff $v_1 = v_2$.

We call the nodes in $T(1)$ *field nodes* and for node n we write $n \in field(F)$. We call the nodes in $T(2)$ *option nodes*. According to Definition 1, a tree consists of a set of fields which each has an associated set of options. Every option has a type and a value; fields can only have one value for each option type. This is a very restricted class of trees, but it is sufficient to represent simple documents such as XML forms.

Example Let Σ denote a finite alphabet of atomic symbols. The *standard form scenario* with bound k is the pair $\langle \mathcal{O}_k, \mathcal{V}_k \rangle$ defined as follows:

- $\mathcal{O}_k = \{label, value, visible\}$.
- $\mathcal{V}_k(label) = \Sigma_1^k$ (the set of non-empty strings of length less than k over Σ).
- $\mathcal{V}_k(value) = \Sigma_0^k$ (the set of finite strings of length less than k over Σ).
- $\mathcal{V}_k(visible) = \{0, 1\}$.

We remark that the bound k represents the maximum length of labels and variables. This bound is included simply to guarantee that the vocabulary is finite.

We conclude this section with some useful definitions and results. In the following definition and for the rest of this paper, we let ϵ denote the empty string.

Definition 2: A field node is *empty* if it has a child v such that $\Phi(v) = (value, \epsilon)$. A document tree is empty if every field node is empty.

We can now define the notion of an *instance* of a empty document. An empty document is a document with labels, but no values. Informally, this is a document in where there are fields or sections to hold data, but no data has been entered.

Definition 3: Let $F = \langle T, \Phi \rangle$ be an empty document tree. An *instance* of F is a document tree $F_1 = \langle T, \Phi_1 \rangle$ such that:

- If $\Phi(v) = (O, V)$ and $\Phi_1(v) = (O_1, V_1)$, then $O = O_1$.
- $\Phi \neq \Phi_1$.

Hence, an instance of an empty document tree is a document tree where the node labels are identical, but some node has a non-empty value. We write $F \Rightarrow F_1$ to indicate that F_1 is an instance of F. If $F \Rightarrow F_1$ and $F \Rightarrow F_2$, we write $F_1 \Leftrightarrow F_2$.

Proposition 1: Each document F is an instance of exactly one empty document.

Proposition 2: The relation $\Leftrightarrow$ is an equivalence relation on document trees.

B. Signed documents

Let $\mathcal{A}$ denote a fixed set of *agents*. We would like to extend our definition of a document in a manner that allows us to represent documents that have been signed by some agent.

Intuitively, signing a document commits the signer to the information in the document. A signature also prevents further modification. An important feature of digital signatures is that they need not apply to a whole document.

Definition 4: A *section* in a document $F = \langle T, \Phi \rangle$ is a subgraph of T.

Let $\mathbf{P}_F$ denote the collection of all sections of F. We can now define the most basic kind of signature, which is a signature applied to a section in an arbitrary document.

Definition 5: An *unsigned unit* is a pair $[F, p]$ where F is a document tree and $p \in \mathbf{P}_F$.

The most important feature of unsigned units is that they can be signed by agents.

Definition 6: A *signed unit* is a pair consisting of an unsigned unit and an agent A. We write $[F, p]_A$ to indicate that the section p in F has been signed by agent A.

Definition 7: A *partially signed document* is a tuple $\langle x_1, \ldots, x_k \rangle$ such that for some $m \leq k$:

1) x_i is a signed unit for all $i \leq m$.
2) x_i is an unsigned unit for all $i > m$.

For partially signed documents, we use the shorthand $\langle F_S, F_U \rangle$ where F_S is a complete signed document of length m and F_U is an unsigned document of length $k - m$. If $m = 0$, it is an *unsigned* document. If $m = k$, it is a *complete* signed document.

C. Document Content

Define the propositional signature $\mathcal{P}$ as follows:

$$\mathcal{P} = \{a_{(L,V)} \mid L \in \mathcal{V}_s(label), V \in \mathcal{V}_s(value)\}.$$

Hence, $\mathcal{P}$ consists of propositional variables corresponding to label-value pairs. Intuitively, $a_{(L,V)}$ will be true just in case $L = V$ is true.

Let $F = \langle T, \Phi \rangle$ be a document tree, let $p \in P_F$ and let $n \in field(F) \cap p$. If n has children n_1, n_2 such that $\Phi(n_1) = (label, L)$ and $\Phi(n_2) = (value, V)$, then define $content(n) = a_{(L,V)}$. If n does not have such children, define $content(n) = \bot$.

Definition 8: The *complete content* of a section p in a document F is the following formula:

$$content(F, p) = \bigwedge \{content(n) \mid n \in field(F) \cap p\}.$$

The complete content of a document is not always visible to a signing agent. For any $n \in field(F) \cap p$, n is *invisible* if n has a child $n_1 \in p$ such that $\Phi(n_1) = (visible, 0)$. We write $n \in visible(F, p)$ if n is not invisible.

Definition 9: The *visible content* of a section p in a document F, is the following formula:

$$viscontent(F, p) = \bigwedge \{content(n) \mid n \in visible(F, p)\}.$$

Informally, an agent is only able to see the visible content of a document. We will see that some security violations are related to confusion about the visible content, as opposed to the complete content.

D. Incorporating Beliefs

We introduce a belief operator B_A for each agent A. Intuitively, $B_A(\phi)$ is true if and only if agent A believes that ϕ is true.

Definition 10: If $[F, p]_A$ is a signed unit, define $Bel([F, p]_A) = B_A(viscontent(F, p))$.

This definition is easily extended to signed documents in general.

Definition 11: Let $F_S = \langle [F_1, p_1]_{A_1}, \ldots [F_k, p_k]_{A_k} \rangle$ be a complete signed document. Then $Bel(F_S) = B_{A_k}(viscontent(F_k, p_k n) \wedge Bel(F_S, p_{k-1}))$.

Note that this formula involves nested belief operators with depth k. The following simple example illustrates the basic idea.

Example Let F be the form from the previous example, with the values "John" and "Smith" as values for the first and last name fields. Let $F_S = \langle [F, p_1]_A, [F, \emptyset]_B \rangle$. In order to simplify the notation, let $john = a_{FirstName,John}$ and let $smith = a_{LastName,Smith}$. By Definition 11:

$$Bel(F_S) = B_B B_A(john \wedge smith).$$

Intuitively, this formula simply states that B believes that A believes his first name is John and his last name is Smith.

Definition 12: For any set $\mathcal{A}$ of agents, an *initial belief state* for $\mathcal{A}$ is a function β that maps every agent in $\mathcal{A}$ to a set of interpretations over the set of formulas of the document $A_{L,V}$.

Informally, we write $B_A(\phi)$ if ϕ is true in every interpretation in $\beta(A)$. We think of the interpretations in $\beta(A)$ as the set of states that A is willing to sign. This set of states can also be interpreted as an abstract representation of a *proposition* that stands for the information that the agent is committing to in signing the document.

IV. Security of Signed Documents

A. Actions

Documents are signed through an iterative process in which an empty document is passed among potential signers. Each signer modifies the document, then signs the document, then gives it to another agent. In practice, agents can modify the document by filling in values or adding new fields. This idea is captured in the following definition.

Definition 13: Let $A \in \mathcal{A}$. Let $\langle F_S, F_U \rangle = \langle x_1, \ldots x_k \rangle$ be a partially signed document, where x_m is the last signed unit in the sequence. Then $\langle x'_1, \ldots x'_k \rangle$ is a *valid transformation* of $\langle F_S, F_U \rangle$ if and only if:

1) For $i \neq m$, $x_i = x'_i$.
2) For $i = m$, $x'_m = [F, p]_A$ for some F such that $\mathbf{P}_F$ contains each p_j contained in x_j, for $j < m$.

Intuitively, a valid transformation is a partially signed document that extends all of the signed units that have preceded the current signature.

Let D be a distinguished agent called the *designer*. Let $\mathcal{A} = \{D, E_1, \ldots, E_k\}$, where each E_i is an agent called a *signer*. An m-round signing procedure is executed as follows.

The General Signing Procedure

1. D chooses an unsigned document $\langle F_U, F_S \rangle$ and a recipient $A \in \mathcal{A}$.
2. A does one of the following:
   (a)Creates a valid transformation $\langle F'_U, F'_S \rangle$ and a recipient $A' \in \mathcal{A}$.
   (b) Abort the game and halt.
3. If $\langle F'_U, F'_S \rangle$ is completely signed, halt.
4. Otherwise, return to step 2 with inputs $\langle F'_U, F'_S \rangle$ and A'.

B. Defining Attacks

Several kinds of attacks can be defined. For example, a *meaning change* attack cccurs when the meaning of a document is changed, following signature. By contrast, a *meaning deception* attack occurs when the actual meaning of a document is different from the apparent meaning. In this paper, we are interested primarily in meaning deception attacks.

Definition 14: Let $\mathcal{A}$ be a set of agents. An unsigned document of length m is *susceptible to a deception attack* relative to the initial belief state β if there is a complete run of the signing procedure such that the outcome $\langle F_U, F_S \rangle$ has the following properties:

- $F_U = \emptyset$
- $\beta(A) \not\models Bel(F_S)$, where A is the last agent to sign F_S

Hence, a document is susceptible to an attack just in case the last agent to sign is committing to a set of visible values that are not believed to be true.

We remark that Definition 14 defines susceptibility to an attack with respect to a given initial belief state. In general, we will say that a document is *susceptible to attack* if it is susceptible to a deception attack *for some initial state* β.

C. Basic Results

In order to find attacks on a specific document, it is necessary to produce the corresponding document tree and then search for sequences of signatures that constitute an attack on the document. In order to facilitate the discussion, we assume that the set of agents is $\{D, A\}$.

In the case of paper documents, we use an informal rule that states that a signature applies to all fields that occur above the signature on a page. In digital documents, the section p to be signed must typically be specified by the document designer.

Proposition 3: Let $[F, p]$ be an unsigned unit. There is an initial belief state β, an agent A and a section p_1 such that $Bel([F, p]) \neq Bel([F, p_1])$.

Informally, this proposition states that $[F, p]$ is susceptible to an attack if A does not know the section p to be signed. This is difficult to address in practice, because many documents do not provide a clear visual representation of the nodes being signed. Another result that should be clear is the following.

Proposition 4: If two option nodes have the same label, the document is susceptible to a deception attack.

Formally, two option nodes with the same label is not a problem in our framework provided that the values are different. However, using the same label for two nodes will make it possible for a user to commit to inconsistent values which trivially leads to attacks.

Proposition 5: Suppose p_1 is a subtree of p_2. If $[F, p_2]$ is susceptible to an attack, then $[F, p_1]$ is susceptible to attack.

This result states that security flaws are, in a sense, monotonic. If a flaw exists for some given section, then the same flaw exists for every sub-section. For this reason, it is intuitively safer to specify a larger section to be signed.

V. Conclusion

We have provided a formal framework for the analysis of digitally signed documents. Documents are represented by trees and signatures are represented symbolically. Security flaws are defined in terms of deception with respect to the commitments that an agent makes when a document is signed.

One direction for future work is to implement our approach to experiment with real XML documents. We have implemented a viewer that defines filters that essentially translate a document created by Lotus forms designer into a tree that is suitable for formal analysis. This is helpful when searching for attacks by hand. However, such proofs are difficult to produce when a document is large. This can be accomplished by treating documents as a special kind of message, then finding attacks my analysing traces in a message passing system.

The development of automated tools is not the only direction for future research. We have made the assumption that document signing is a linear process, but this is too restrictive in some cases. For example, there are cases when two agents are intended to sign a document simultaneously (for example in a joint loan application). In such cases, we would need a partial ordering over signatures, or even a more general graph structure. Such a generalised signing structure seems to offer no significant problems in principle; we leave this extended case for future work.

References

[1] Agray, N.; van der Hoek, W.; and de Vink, E. 2002. On BAN logics for industrial security protocols. In Dunin-Keplicz, B., and Nawarecki, E., eds., *Proceedings of CEEMAS 2001*, 29–36.

[2] Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50(2):510–530.

[3] Bertrand, R.; Hearn, J.; and Lett, B. 1995. The North American Pre- and Post-Processing Equipment Market: Capturing the Benefits and Avoiding the Pitfalls. Strategic Analysis Report, Gartner Group.

[4] Blair, B. and Boyer, J. 1999. XFDL: Creating Electronic Commerce Transaction Records Using XML. Proceedings of the Eighth International World Wide Web Conference.

[5] Boyer, J. 2005. Enterprise-level Web Form Applications with XFDL and XForms. Proceedings of XML 2005 Conference and Exposition.

[6] Burrows, M.; Abadi, M.; and Needham, R. 1990. A logic of authentication. *ACM Transactions on Computer Systems* 8(1):18–36.

[7] Chellas, B. 1980. Modal Logic: An Introduction. Cambridge University Press.

[8] Delgrande, J.P.; Hunter, A.; and Grote, T 2009. Modelling Cryptographic Protocols in a Theory of Action. *The Ninth International Symposium on Logical Formalizations of Commonsense Reasoning.*

[9] Guttman, J., and Thayer, J. 2000. Authentication tests. In *Proceedings 2000 IEEE Symposium on Security and Privacy*.

[10] Syverson, P., and Cervesato, I. 2001. The logic of authentication protocols. In Focardi, R., and Gorrieri, R., eds., *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*. Springer-Verlag. 63–136.