On the Decidability of a Class of XOR-based Key-management APIs

Véronique Cortier[1]* and Graham Steel[2]**

[1] Loria UMR 7503 & CNRS & INRIA Lorraine projet Cassis, France
Veronique.Cortier@loria.fr
http://www.loria.fr/~cortier
[2] School of Informatics, University of Edinburgh, Scotland
Graham.Steel@ed.ac.uk
http://homepages.inf.ed.ac.uk/gsteel

Abstract. We define a new class of security protocols using XOR, and show that secrecy after an unbounded number of sessions is decidable for this class. The new class is important as it contains examples of key-management APIs, such as the IBM 4758 CCA API, which lie outside the classes for which secrecy has previously been shown to be decidable. Earlier versions of the CCA API were shown to have serious flaws, and the fixes introduced by IBM in version 2.41 have not been formally verified in a model with unbounded sessions. In showing the decidability of this class, we also suggest a simple decision procedure, which we plan to implement in future work to finally verify the revised API.

1 Introduction

Security protocols are small programs that aim to secure communications over a public network like the Internet. The design of such protocols is notoriously difficult and error-prone. Formal methods have proved their usefulness in the rigorous analysis of security protocols. However, security protocol analysis is, in general, undecidable [7]. In recent years, a number of theoretical results have shown the decidability of particular fragments of the problem, or have proposed safe abstractions [4] that, in practice, make the problem more amenable to automated approaches. For example, we know that security is co-NP-complete if the number of sessions is bounded [11], which is often sufficient to discover attacks. Recent results are also concerned with showing decidability of the analysis of protocols that use operators with non-trivial algebraic properties. For example, the problem for protocols including XOR (exclusive OR) was shown to be decidable first for a bounded number of sessions [5, 2], and then, for certain classes of protocol, for an unbounded number of sessions [3, 13].

* This work has been partially supported by the ACI-SI project SATIN and the RNTL project POSE.

** Supported by EPSRC Grant number GR/S98139/01, ‘Automated Analysis of Security Critical Systems’.

An important example of a protocol that uses XOR is the key management transaction set of the IBM 4758 CCA API[1]. This is the application program interface of a tamper-proof hardware security module (HSM), designed for use in applications such as PIN processing. It can be thought of as a set of two-party security protocols, each describing an exchange between the HSM and the user. An intruder trying to break in to the box may compose the protocols in any order. Such hardware security modules are widely used in security critical systems such as electronic payment and automated teller machine (ATM) networks. Bond and Anderson discovered flaws in the transaction set that allowed an intruder to obtain access to PINs [1]. The attack requires the intruder to exploit the algebraic properties of XOR. IBM made changes to the API in version 2.41, but though previous formal work has been able to rediscover the flaws in the old version [12, 8], the new version of the API has not been formally verified to ensure that the required secrecy properties hold, for an unbounded number of sessions.

In this paper, we first show that the IBM CCA API is outside both the classes of protocols using XOR which have been previously shown to be decidable. We then define a new class of protocols, called WFX-class, that includes the IBM CCA API, and prove its decidability. Our proof is considerably simpler that the corresponding proofs for the two previously treated classes, and suggests a simpler decision procedure.

The paper is organised as follows: in §2, we give the formal definition of our setting. We also explain the IBM CCA API in more detail. In §3, we define our WFX-class of protocols, compare it with the existing decidable classes, and state and prove the theorems showing its decidability. This is followed by some thoughts on further work in §4.

2 Background

To provide the necessary background, we first define our (mostly standard) notation for reasoning about protocols, and then explain the nature and purpose of security APIs.

2.1 Definitions

Cryptographic primitives are represented by functional symbols. More specifically, we consider the *signature* Σ containing an infinite number of constants including some special constant 0 and two non constant symbols $\{\!|\,\_\,|\!\}_{\_}$ and $\oplus$

[1] CCA stands for 'Common Cryptographic Architecture'. See `http://www-3.ibm.com/security/cryptocards/pcicc.shtml`

of arity 2. We also assume an infinite set of variables $\mathcal{V}$. The set of *terms or messages* is defined inductively by

$$\begin{array}{lll} T ::= & & \text{terms} \\ & \mid\ x & \text{variable } x \\ & \mid\ f(T_1, \ldots, T_k) & \text{application of symbol } f \in \Sigma \text{ of arity } k \geq 1 \\ & \mid\ c & \text{constant } c \in \Sigma \end{array}$$

A term is *ground* if it has no variable.

The term $\{\!|\, m \,|\!\}_k$ is intended to represent the message m encrypted with the key k (using symmetric encryption). The term $m_1 \oplus m_2$ represents the message m_1 XORed with the message m_2. The constants may represent agent identities, nonces or keys for example.

Substitutions are written $\sigma = \{x_1 = t_1, \ldots, x_n = t_n\}$ with $dom(\sigma) = \{x_1, \ldots, x_n\}$. σ is *ground* iff all of the t_i are ground. The application of a substitution σ to a term t is written $\sigma(t) = t\sigma$.

The *size* of a term t, denoted by $|t|$, is defined as usual by the total number of symbols used in t. More formally, $|a| = 1$ if a is a constant or a variable and $|f(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t|$ if f is of arity $k \geq 1$. The size of a set of terms S is the sum of the size of the terms in S.

We equip the signature with an equational theory E that models the algebraic properties of the XOR operator:

$$\begin{array}{cc} x \oplus (y \oplus z) = (x \oplus y) \oplus z & x \oplus y = y \oplus x \\ x \oplus x = 0 & x \oplus 0 = x \end{array}$$

It defines an equivalence relation that is closed under substitutions of terms for variables and under application of contexts. In particular, we say that two terms t_1 and t_2 are equal, denoted by $t_1 = t_2$ if they are equal modulo the equational theory E. If two terms are equal using only the equations of the first line, we say that they are equal modulo Associativity and Commutativity (AC).

Intruder capabilities and the protocol behaviour are described using *rules* of the form $t_1, \ldots, t_n \rightarrow t_{n+1}$ where the t_i are terms.

Example 1. The intruder capabilities are represented by the following set of three rules:

$$\begin{array}{rl} x, y \rightarrow \{\!|\, x \,|\!\}_y & \text{encryption} \\ \{\!|\, x \,|\!\}_y, y \rightarrow x & \text{decryption} \\ x, y \rightarrow x \oplus y & \text{xoring} \end{array}$$

The set of deducible terms is the reflexive and transitive closure of the rewrite rules.

Definition 1. *Let $\mathcal{R}$ be a set of rules. Let S be a set of ground terms. The term u is* one-step deducible *from S if there exists a rule $t_1, \dots, t_n \to t \in \mathcal{R}$ and a ground substitution θ such that $t_i\theta \in S$ and $u = t\theta$.*

A term u is deducible *from S, denoted by $S \vdash_{\mathcal{R}} u$, if $u \in S$ or there exist ground terms $u_1, \dots, u_n$ such that $u_n = u$ and u_i is one-step deducible from $S \cup \{u_1, \dots, u_{i-1}\}$ for every $1 \le i \le n$. The sequence $u_1, \dots, u_n$ is a* proof *that $S \vdash_{\mathcal{R}} u$.*

We write $\vdash$ instead of $\vdash_{\mathcal{R}}$ when $\mathcal{R}$ is clear from the context.

Example 2. Let $\mathcal{R}$ be the set of rules described in Example 1. Let $S = \{\{\!|\, n \,|\!\}_a, a \oplus b, b\}$. Then n is deducible from S and $\{\!|\, n \,|\!\}_a, a \oplus b, b, a$ is a proof of $S \vdash n$. Indeed a is one-step deducible from $\{a \oplus b, b\}$ using the rule $x, y \to x \oplus y$ and the fact that $(a \oplus b) \oplus b = a$ and n is one-step deducible from $\{\{\!|\, n \,|\!\}_a, a\}$ using the rule $\{\!|\, x \,|\!\}_y, y \to x$.

2.2 Security APIs

Hardware security modules, such as the IBM 4758, are widely used in security critical systems such as electronic payment and automated teller machine (ATM) networks. They typically consist of a cryptoprocessor and a small amount of memory inside a tamper-proof enclosure. They are designed so that should an intruder open the casing or insert probes to try to read the memory, it will auto-erase in a matter of nanoseconds. In a typical ATM network application, this tamper-proof memory will be used to store the master keys for the device. Further keys will then be stored on the hard drive of a computer connected to the HSM, encrypted under the HSM's master key. These keys can then only be used by sending them back into the HSM, together with some other data, under the HSM's application program interface (API). The API will typically consist of a few dozen key management and PIN processing commands, and their prescribed responses. One can think of the API as defining a number of two-party protocols between a user and the HSM. However, there are some important differences between HSM APIs and typical key-exchange protocols. We are concerned only with two 'agents', the HSM itself and a user (or malicious intruder). The HSM itself is considered to be 'stateless', in the sense that no information inside the HSM changes during transactions. HSMs APIs also typically make use of bitwise operations such as XOR.

The attacks on the 4758 CCA were first discovered by Bond and Anderson [1]. They exploit the way the 4758 constructs keys of different types. The CCA supports various key types, such as data keys, key encrypting keys, import keys and export keys. Each type has an associated 'control vector' (a public value), which is XORed against the master key to produce the required key type. So if *data* is the control vector for data keys, then all data keys will be stored outside the 4758 box encrypted under $km \oplus data$. Only particular types of keys will be accepted by the HSM for particular operations. For example, there is a data encryption command, which allows data keys to be used to encrypt given plaintext, but disallows, for example, PIN derivation keys from being used for such a purpose. In our formalism, we write the data encryption command like this[2]:

$$\mathrm{x}, \{\!|\, \mathrm{xk1} \,|\!\}_{\mathrm{km}\oplus\mathrm{data}} \rightarrow \{\!|\, \mathrm{x} \,|\!\}_{\mathrm{xk1}} \qquad \textbf{(Encrypt Data)}$$

In Bond's attack, the intruder changes the type of a key by exploiting the properties of XOR. The attack assumes that a would-be intruder has access to one part of a three part key. Keys are divided into parts to allow separate people to physically transfer key information to another 4758 module, and thereby to facilitate secure communication between two HSMs (for example, between a module in an ATM and a module in a bank). The idea of dividing the key into three parts is that attacks requiring collusion between three employees are considered infeasible. In Bond's attack, the APIs key part import command is used to generate keys with a known difference. The attack is assumed to be carried out by the third member of a 3-person team, who has the third part, call it $k3$, of a 3 part key $kek = k1 \oplus k2 \oplus k3$. This key kek is to be used as an import key with type *imp*. He is supposed to add $k3$ to the previous parts using the third key part import command:

$$\{\!|\, \mathrm{xk1} \oplus \mathrm{xk2} \,|\!\}_{\mathrm{km}\oplus\mathrm{kp}\oplus\mathrm{xtype}}, \mathrm{xk3}, \mathrm{xtype} \rightarrow \{\!|\, \mathrm{xk1} \oplus \mathrm{xk2} \oplus \mathrm{xk3} \,|\!\}_{\mathrm{km}\oplus\mathrm{xtype}}$$
$$\textbf{(Key Part Import (3))}$$

For the attack, he first XORs his key part $k3$ against the (publicly known) control vectors for data keys and PIN derivation keys, *data* and *pin*. Then he uses the key part import command like this:

$$\{\!|\, \mathrm{k1} \oplus \mathrm{k2} \,|\!\}_{\mathrm{km}\oplus\mathrm{kp}\oplus\mathrm{imp}}, \mathrm{k3} \oplus \mathrm{data} \oplus \mathrm{pin}, \mathrm{imp} \rightarrow \{\!|\, \mathrm{kek} \oplus \mathrm{pin} \oplus \mathrm{data} \,|\!\}_{\mathrm{km}\oplus\mathrm{imp}}$$

The key import command can then be used to change a key of one type to any other type. The rule for the key import command looks like this:

$$\{\!|\, \mathrm{xkey} \,|\!\}_{\mathrm{xkek}\oplus\mathrm{xtype}}, \mathrm{xtype}, \{\!|\, \mathrm{xkek} \,|\!\}_{\mathrm{km}\oplus\mathrm{imp}} \rightarrow \{\!|\, \mathrm{xkey} \,|\!\}_{\mathrm{km}\oplus\mathrm{xtype}}$$
$$\textbf{(Key Import)}$$

[2] Symbol names beginning with x or y denote variables, other lower case letters denote ground atoms. There is a full listing of the protocol in Appendix 1

Having obtained the tampered key part
$\{\!|\, \mathrm{kek} \oplus \mathrm{pin} \oplus \mathrm{data} \,|\!\}_{\mathrm{km} \oplus \mathrm{imp}}$, the intruder uses the command like this:

$$\{\!|\, \mathrm{pdk} \,|\!\}_{\mathrm{kek} \oplus \mathrm{pin}}, \mathrm{data}, \{\!|\, \mathrm{kek} \oplus \mathrm{data} \oplus \mathrm{pin} \,|\!\}_{\mathrm{km} \oplus \mathrm{imp}} \rightarrow \{\!|\, \mathrm{pdk} \,|\!\}_{\mathrm{km} \oplus \mathrm{data}}$$

This allows a PIN derivation key, pdk, of type pin, to be converted to a data key, and which can then used to encrypt data, which is a critical flaw: a customer's PIN is just his account number encrypted under a PIN derivation key. So, the attack allows a criminal to generate a PIN for any card number. Bond and Anderson discovered several variations of the attack, suitable for intruders with differing initial knowledge.

IBM responded to the attack by publishing a 7-page document describing how the attack could be prevented[3]. This is divided into three possible solutions. The first is to disable the key parts import mechanism completely and use asymmetric cryptography for importing keys. The second is to use access control to prevent any single user from using both the key part import commands and the key import command. The third is to use procedural controls to check that key parts have not been not tampered with. The aim of our current project is to use formal techniques to analyse these recommendations: to prove them secure if they really are secure, or to provide counterexamples if they are not. The work in this paper is concerned with verifying the second recommendation, where key part import is still enabled, but can only be used in a restricted way. Formally, this is represented by analysing the API against an attack by two distinct intruders: the first has access to all the commands except the key import command. The second has access to all the commands except the key part import commands. The complete formalism of the initial and corrected of the 4758 CCA API protocol is presented in Appendix.

Previous formal work has focused almost entirely on rediscovery of the flaws in the old version of the API. Ganapathy *et al.* used model checking [8], Steel used theorem proving with XOR-constraints in daTac [12], and Youn *et al.* the theorem prover Otter [14]. None of this work was able to verify any of the IBM fixes to be secure for an unbounded number of sessions (i.e., an unlimited number of command invocations, in any order). The work in this paper aims to address this problem. Note that in Bond's original paper, a fix to the API was proposed which involved scrapping the use of XOR to create key types, and using one way functions instead. Courant and Monin formally verified this fix using the Coq proof assistant [6]. However, this is of limited relevance to real-world systems, since the fix would not be backwards-compatible, and so is very unlikely to be implemented.

[3] Available from:
`http://www.cl.cam.ac.uk/~mkb23/research/CVDif-Response.pdf`

3 A New Decidable Class

Here we first define our class using the notation of §2.1, then compare it to existing decidable classes, and finally give the proof of decidability.

3.1 Definition of WFX-Class Protocols

Rather than restricting the use of variables in protocol rules, we take advantage of the form of API-like protocols, noticing that they only perform simple operations.

Definition 2. *A term t is an* XOR *term if $t = \bigoplus_{i=1}^{n} u_i$, $n \geq 1$ where each u_i is a variable or a constant.*

A term t is an encryption *term if $t = \{\!|\, u \,|\!\}_v$ where u and v are XOR terms.*

A term t is a well-formed term *if it is either an encryption term or an XOR term. In particular, a well formed term contains no nested encryption.*

A rule $t_1, \ldots, t_n \to t_{n+1}$ is well formed *if*

- *each t_i is a well-formed term.*
- *$Var(t_{n+1}) \subseteq \bigcup_{i=1}^{n} Var(t_i)$ (no variable is introduced in the right-hand-side of a rule).*

A proof is well-formed *if it only uses well-formed terms.*

Definition 3. *The* WFX-class protocol *consists of a pair $(\mathcal{R}, S)$, where $\mathcal{R}$ is a finite set of well-formed rules, and S is a finite set of ground, well-formed terms.*

Intuitively, the rules in $\mathcal{R}$ represent the commands of the API and the intruder capabilities, and the ground terms S the initial knowledge of the intruder. We call our class WFX since these are well-formed protocols using the XOR operator.

In particular, the rules representing the intruder capabilities (defined in Example 1) and the rules representing the 4758 CCA API protocol (introduced in §2.2 and listed in full in the appendix) are all well-formed.

3.2 Comparison to Existing Classes

To the best of our knowledge, there exist only two decidable classes for protocols with XOR, for an unbounded number of sessions. In both cases, the main difference with our class is that we make restrictions on the combination of functional symbols rather than on the occurrences of variables. As a consequence, our class is incomparable to the two existing ones.

In the class defined in [3], each rule $t_1, \ldots, t_n \to t_{n+1}$ must satisfy the following requirement:

- either each t_i contains only one variable x
- or the set of variables is $x_1, \ldots, x_k$ and there exists a functional symbol f of arity k such that each t_i is either some x_j or is of the form $t[f(x_1, \ldots, x_k)/y]$ where t is a term that has only one variable y.

The rules for the CCA key-management API do not fit into this class. In particular, the rule Key Import (given above in §2.2 and crucial for modelling the API protocol) does not satisfy these conditions since, for example, the term $\{\!|\, \mathrm{xkek} \,|\!\}_{\mathrm{km}\oplus\mathrm{imp}}$ does not contain all the variables of the rule and is not reduced to a single variable either.

In [13], a variable dependence graph G_r is associated to each rule r. In this graph, nodes represent the terms of the rules, and two terms are adjacent if they share a variable. Each rule r must satisfy the following conditions:

1. each term is linear: no variable occurs twice in a term,
2. G_r is acyclic and adjacent terms share at most one variable.

Again, the rule Key Import does not fulfil the conditions since three distinct terms share the variable xtype, thus creating a cycle in the variable dependence graph.

Conversely, in our class we cannot consider nested encryption or XORed encryption. For example, terms of the form $\{\!|\, \{\!|\, a \,|\!\}_y \,|\!\}_y$ or $y \oplus \{\!|\, a \,|\!\}_y$ are forbidden in our rules. In addition, we do not consider arbitrary cryptographic primitives like in [3] and [13]. In particular, we do not consider the pairing function.

3.3 Proof of Decidability

The key idea of our decidability result is to show that only well-formed terms need to be considered when checking for the deducibility of a (well-formed) term. In particular, there is no need to consider nested encryption. This allows us to consider only a finite number of terms: we have a finite number of atoms in the initial set of rules which can only be combined by encryption and XORing, and XORing identical atoms results in cancellation. At the end of the proof, we comment on the complexity of the resulting decision procedure.

We first prove that whenever an encryption occurs in a deducible term, the encryption is itself deducible.

Proposition 1. *Let $\mathcal{R}$ be a set of well-formed rules. Let S be a set of ground well-formed terms (intuitively the initial knowledge). Let u be a ground term deducible from S and $v_1, \ldots, v_n$ be a proof that $S \vdash u$. If $\{\!|\, u_1 \,|\!\}_{u_2}$ is a subterm of u then $\{\!|\, u_1 \,|\!\}_{u_2} \in S \cup \{v_1, \ldots, v_n\}$.*

The proof is by induction on the number of steps needed to obtain u. Note that $\{\!|\, u_1 \,|\!\}_{u_2}$ must be ground since u is ground.

– if $u \in S$, let $\{\!|\, u_1 \,|\!\}_{u_2}$ be a subterm of u. Since S is a set of ground well-formed terms, we must have $u = \{\!|\, u_1 \,|\!\}_{u_2}$ thus $\{\!|\, u_1 \,|\!\}_{u_2} \in S$.
– if u is one-step deducible from $S \cup \{v_1, \ldots, v_{n-1}\}$. Then there exists a well-formed rule $t_1, \ldots, t_k \to t$ and a ground substitution θ such that $t_i\theta \in S \cup \{v_1, \ldots, v_{n-1}\}$ and $u = t\theta$. Let $\{\!|\, u_1 \,|\!\}_{u_2}$ be a subterm of u.
  - either $u = \{\!|\, u_1 \,|\!\}_{u_2}$ and thus $\{\!|\, u_1 \,|\!\}_{u_2} \in S \cup \{v_1, \ldots, v_n\}$ (remember that $v_n = u$).
  - or $\{\!|\, u_1 \,|\!\}_{u_2}$ is a strict subterm of u. By well-formedness of the rule, $\{\!|\, u_1 \,|\!\}_{u_2}$ must be a subterm of $x\theta$ for some x variable of t. Thus $\{\!|\, u_1 \,|\!\}_{u_2}$ is a subterm of a term in $S \cup \{v_1, \ldots, v_{n-1}\}$. By induction hypothesis, we deduce that $\{\!|\, u_1 \,|\!\}_{u_2} \in S \cup \{v_1, \ldots, v_{n-1}\}$ thus $\{\!|\, u_1 \,|\!\}_{u_2} \in S \cup \{v_1, \ldots, v_n\}$.

We are now ready to prove our main result, that states that only well-formed terms need to be considered when checking for deducibility.

Proposition 2. *Let $\mathcal{R}$ be a set of well-formed rules and S be a set of ground well-formed terms such that*

– *$\mathcal{R}$ contains the rule $x, y \to x \oplus y$;*
– *S contains 0 (the null element for XOR should always be known to an intruder).*

Let u be a ground well formed term deducible from S. Then there exists a well-formed proof of $S \vdash u$.

Proof: We first introduce some notation. Any term t can by written $t = a_1 \oplus \cdots \oplus a_k \oplus \{\!|\, t_1 \,|\!\}_{t'_1} \oplus \cdots \oplus \{\!|\, t_p \,|\!\}_{t'_p}$ (equality modulo AC) where the a_i are constants or variables. The *atomic part* of t, denoted by $\mathsf{atom}(t)$, is $a_1 \oplus \cdots \oplus a_k$. The set of *external encryptions* of t is $\mathsf{Senc}(t) = \{\{\!|\, t_1 \,|\!\}_{t'_1}, \ldots, \{\!|\, t_p \,|\!\}_{t'_p}\}$. By convention, if the atomic part of t is empty, that it $t = \{\!|\, t_1 \,|\!\}_{t'_1} \oplus \cdots \oplus \{\!|\, t_p \,|\!\}_{t'_p}$, we define $\mathsf{atom}(t) = 0$. The *elements* of a term t is the set $\mathsf{elements}(t) = \{\mathsf{atom}(t)\} \cup \mathsf{Senc}(t)$. For any term t the following equality holds:

$$t = \mathsf{atom}(t) \oplus \bigoplus_{c \in \mathsf{Senc}(t)} c$$

We define the function $\bar{\cdot}$ over terms that replaces any internal encryption term by 0. More formally, $\bar{\cdot}$ is inductively defined as follows:

$$\begin{aligned} \overline{u} &= u \qquad \text{if } u \text{ is a variable or a constant} \\ \overline{t_1 \oplus t_2} &= \overline{t_1} \oplus \overline{t_2} \\ \overline{\{\!|\, t_1 \,|\!\}_{t_2}} &= \{\!|\, \overline{\overline{t_1}} \,|\!\}_{\overline{\overline{t_2}}} \end{aligned}$$

where $\overline{\overline{\cdot}}$ is defined by:

$$\begin{aligned}
\overline{\overline{u}} &= u && \text{if } u \text{ is a variable or a constant}\\
\overline{\overline{t_1 \oplus t_2}} &= \overline{\overline{t_1}} \oplus \overline{\overline{t_2}}\\
\overline{\overline{\{\!|\, t_1 \,|\!\}_{t_2}}} &= 0
\end{aligned}$$

The function $\overline{\cdot}$ is extended to sets of terms as expected.

Consider a proof $u_1, \ldots, u_n$ of $S \vdash u$. We show by induction on n that we can construct well-formed terms $w_1, \ldots, w_p$ such that

- $w_1, \ldots, w_p$ is a well-formed proof of $S \vdash \overline{e}$ for any $e \in \mathsf{elements}(u)$,
- $\bigcup_{i=1}^{n} \overline{\mathsf{elements}(u_i)} \subseteq \{w_1, \ldots, w_p\}$.

This would conclude the proof since due to the well-formedness of u, $\overline{\mathsf{elements}(u)} = \{u\}$.

The base case $u \in S$ is trivial since we have $\overline{\mathsf{elements}(u)} = \{u\}$ by well-formedness of terms in S.

Assume now that there are well-formed terms $w_1, \ldots, w_p$, such that $w_1, \ldots, w_p$ is a well-formed proof of $S \vdash \overline{e}$ for any $e \in \mathsf{elements}(u_j)$, $1 \leq j \leq i$, such that $\bigcup_{j=1}^{i} \overline{\mathsf{elements}(u_j)} \subseteq \{w_1, \ldots, w_p\}$. Let us show that we can construct a well-formed term w_{p+1} such that $\bigcup_{j=1}^{i+1} \overline{\mathsf{elements}(u_j)} \subseteq \{w_1, \ldots, w_p, w_{p+1}\}$ and $w_1, \ldots, w_p, \ldots, w_{p+1}$ is a well-formed proof of $S \vdash \overline{e}$ for any $e \in \mathsf{elements}(u_{i+1})$. The term u_{i+1} is one-step deducible from $S \cup \{u_1, \ldots, u_i\}$ thus there exists a well-formed rule $t_1, \ldots, t_k \to t$ and a ground substitution θ such that for all $1 \leq j \leq k$ $t_j\theta \in S \cup \{u_1, \ldots, u_i\}$ and $u_{i+1} = t\theta$. Let θ' be the substitution obtained from θ by replacing each external encryption by 0. More precisely, $x\theta' = \mathsf{atom}(x\theta)$ for any $x \in dom(\theta)$.

It is easy to verify that for any well-formed term v, we have $v\theta' \in \overline{\mathsf{elements}(v\theta)}$:

1. either v is an encryption term and $\overline{\mathsf{elements}(v\theta)} = \{\overline{v\theta}\} = \{v\theta'\}$
2. or v is an XOR term and $v\theta' = \mathsf{atom}(v\theta)$.

Thus, for each t_j, since $t_j\theta = u_l$ for some $1 \leq l \leq i$, we have $t_j\theta' \in \bigcup_{l=1}^{i} \overline{\mathsf{elements}(u_l)}$ thus $t_j\theta' \in \{w_1, \ldots, w_p\}$. We deduce that $t\theta'$ is one-step deducible from $\{w_1, \ldots, w_p\}$ and $\{w_1, \ldots, w_p, w_{p+1}\}$ is a well-formed proof of $S \vdash t\theta'$ where $w_{p+1} = t\theta'$ (note that $t\theta'$ is a well-formed term).

- If t is an encryption term, then $\overline{\mathsf{elements}(t\theta)} = \{t\theta'\}$, which concludes the proof.
- If t is a XOR term, then $\mathsf{elements}(t\theta) = \{\mathsf{atom}(t\theta)\} \cup \mathsf{Senc}(t\theta)$. We have $\mathsf{atom}(t\theta) = t\theta' = w_{p+1}$ thus $\mathsf{atom}(t\theta) \in \{w_1, \ldots, w_p, w_{p+1}\}$. Now, for any $c \in \mathsf{Senc}(t\theta)$ external encryption of $t\theta$, since t is a XOR term, c

must be a subterm of some x variable of t thus c already occurs as subterm of some $t_j\theta$. By Proposition 1, we have that $c \in \overline{S \cup \{u_1, \ldots, u_i\}}$ thus $\mathsf{elements}(c) = \{\overline{c}\} \subseteq \{w_1, \ldots, w_p\}$. We deduce that $\overline{\mathsf{elements}(t\theta)} \subseteq \{w_1, \ldots, w_p, w_{p+1}\}$ $\square$.

Using Proposition 2, we can now easily conclude the decidability of deducibility.

Theorem 1. *The following problem*

- *Given a finite set of well-formed rules $\mathcal{R}$ containing the rule $x, y \to x \oplus y$, a finite set S of ground well-formed terms containing 0 and a ground well-formed term u,*
- *Does $S \vdash_{\mathcal{R}} u$?*

is decidable in exponential time in the size of $\mathcal{R}$, S and u.

Let $a_1, \ldots, a_n$ be the constants that occur in $\mathcal{R}$, S or u. Let k be the maximal number of terms in the left-hand side of a rule in $\mathcal{R}$. For any $t_1, \ldots, t_l \to t \in \mathcal{R}$, we have $l \leq k$. We show that $S \vdash_{\mathcal{R}} u$ can be decided in $\mathcal{O}(2^{2kn})$.

The decision procedure is as follows: we saturate S by adding any well-formed deducible terms. We obtain a set S^*. By Proposition 2, $S \vdash_{\mathcal{R}} u$ if and only if $u \in S^*$. In S^* there are at most

- 2^n XOR terms
- and $2^n \times 2^n = 2^{2n}$ encryption terms

thus $|S^*| \leq 2^{2n+1}$. Note that we consider here terms modulo AC which means that we only consider one concrete representation for each class of terms equal modulo AC. This can be done for example by fixing an arbitrary order on the constants and use it to normalise terms.

Now, at each iteration, we need to consider any tuple of terms $(u_1, \ldots, u_l)$ with u_i in the set that is being saturated and check whether it is an instance of the left-hand side of a rule of $\mathcal{R}$. We consider at most $|S^*|^k \leq 2^{k(2n+1)}$ tuples at each iteration. All together, we need at most $\mathcal{O}(2^{(k+1)(2n+1)})$ operations to compute S^*.

4 Further Work

First, we are currently implementing our procedure in order to (hopefully) obtain a proof of safety of the new version of the IBM 4758 CCA API protocol. Since our algorithm is exponential, its implementation requires further optimisations and modifications. In particular, it should not be necessary to construct *all* deducible messages. A backward search should allow us to consider only the messages which can lead to the target (the secret). In addition, we have proved an exponential complexity upper bound but we do not know whether it is EXPTIME-hard. We plan to investigate more precisely the complexity class of the problem of deciding deducibility for WFX protocols.

Our second area of future work is to extend our XOR decidability result to more cryptographic primitives and relaxed hypotheses. Regarding cryptographic primitives, a first primitive to consider in order to capture more security protocols, is the concatenation function. Regarding the hypotheses, one key element of our result is that any encrypted term that occurs in a deducible term is itself deducible. For this property to hold, we do not actually require the terms in the premise of a rule to be well-formed. We plan to investigate larger classes of protocols where the notion of well-formedness is relaxed. Additionally, it should be possible to extend the proof that nested encryptions may be safely ignored to protocols with unbounded message length, e.g. group protocols. This result alone would not be sufficient to show decidability, but would still provide a powerful heuristic for use in automated tools.

Third, we plan to study further properties of the encryption scheme used by the IBM 4758, in order to get closer to the implementation. In particular, decryption is of course an explicit operation, publicly available. It could be represented by a function symbol dec and its associated cancellation rule $\mathsf{dec}(\{\!|\, x \,|\!\}_y, y) = x$. In the 4758 setting, it is also likely that re-encrypting a message right after decrypting it leads to the same message. Thus the following equality should also be considered: $\{\!|\, \mathsf{dec}(x, y) \,|\!\}_y = x$. This equation enables more transitions. For example, consider the protocol rule $\{\!|\, x \,|\!\}_k \rightarrow \mathit{secret}$: if an agent receives an encrypted message with k, he sends out the secret. If no message with the key k has been published, this protocol rule is secure in our setting while it is not using the new equation. Indeed, by sending some public value a, the intruder can learn the secret since $a = \{\!|\, \mathsf{dec}(a, k) \,|\!\}_k$. This example shows that adding the two equations can impact strongly the security of a protocol. Our goal is to prove (or disprove) the safety of the CCA API protocol in the presence of these two equations. This problem seems related to the result of Millen [9], where it is shown that if a symmetric-key protocol specifies no encryption of free, untyped variables, then it may safely be analysed in the free algebra with only implicit decryption, i.e. without considering the two equations above. Our protocol is clearly outside this class. However, we suspect that there may be a variation of our well-formedness property for API-class protocols that is also sufficient to allow a protocol to be analysed with only implicit encryption. This would form a distinct but overlapping class with Millen's 'EV-free' class. In order to prove this, our equivalent of Millen's 'star substitution', which converts derivations with explicit decryption to derivations without, would have to be a little more complex. Without EV-freedom, we cannot be sure that substitutions do not give rise to reductions via the cancellation equations. However, without nested encryptions, these reductions must be at the top level of a term, in which case there should be an equivalent derivation step using implicit decryption available. Part of our future work will be to investigate this, and attempt a proof.

References

1. M. Bond and R. Anderson. API level attacks on embedded systems. *IEEE Computer Magazine*, pages 67–75, October 2001.
2. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, 2003.
3. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164, Valencia, Spain, June 2003. Springer-Verlag.
4. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. of the 12th European Symposium On Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 99–113, Warsaw, Poland, April 2003. Springer Verlag.
5. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 271–280, 2003.
6. J. Courant and J.-F. Monin. Defending the bank with a proof assistant. In *Proceedings of Workshop on Issues in the Theory of Security (WITS '06)*, Vienna, March 2006.
7. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, Trento, Italia, 1999.
8. V. Ganapathy, S. A. Seshia, S. Jha, T. W. Reps, and R. E. Bryant. Automatic discovery of API-level exploits. In *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, pages 312–321, New York, NY, USA, May 2005. ACM Press.
9. J. K. Millen. On the freedom of decryption. *Inf. Process. Lett.*, 86(6):329–333, 2003.
10. R. Nieuwenhuis, editor. *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*, volume 3632 of *Lecture Notes in Computer Science*. Springer, 2005.
11. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.
12. G. Steel. Deduction with XOR constraints in security API modelling. In Nieuwenhuis [10], pages 322–336.
13. K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational Horn clauses. In Nieuwenhuis [10], pages 337–352.
14. P. Youn, B. Adida, M. Bond, J. Clulow, J. Herzog, A. Lin, R. Rivest, and R. Anderson. Robbing the bank with a theorem prover. Technical Report UCAM-CL-TR-644, University of Cambridge, August 2005.

Appendix - Protocol Description

IBM CCA Symmetric Key Management Transaction Set

$$\mathrm{xk1}, \mathrm{xtype} \rightarrow \{\!|\, \mathrm{xk1} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{xtype}}$$ **(Key Part Import (1))**

$$\{\!|\, \mathrm{xk1} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{xtype}}, \mathrm{xk2}, \mathrm{xtype} \rightarrow \{\!|\, \mathrm{xk1} \oplus \mathrm{xk2} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{xtype}}$$ **(Key Part Import (2))**

$$\{\!|\, \mathrm{xk1} \oplus \mathrm{xk2} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{xtype}}, \mathrm{xk3}, \mathrm{xtype} \rightarrow \{\!|\, \mathrm{xk1} \oplus \mathrm{xk2} \oplus \mathrm{xk3} \,|\!\}_{\mathrm{km} \oplus \mathrm{xtype}}$$ **(Key Part Import (3))**

$$\{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{xkek} \oplus \mathrm{xtype}}, \mathrm{xtype}, \{\!|\, \mathrm{xkek} \,|\!\}_{\mathrm{km} \oplus \mathrm{imp}} \rightarrow \{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{km} \oplus \mathrm{xtype}}$$ **(Key Import)**

$$\{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{km} \oplus \mathrm{xtype}}, \mathrm{xtype}, \mathrm{xtype}, \{\!|\, \mathrm{xkek} \,|\!\}_{\mathrm{km} \oplus \mathrm{exp}} \rightarrow \{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{xkek} \oplus \mathrm{xtype}}$$ **(Key Export)**

$$\mathrm{x}, \{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{km} \oplus \mathrm{data}} \rightarrow \{\!|\, \mathrm{x} \,|\!\}_{\mathrm{xkey1}}$$ **(Encrypt Data)**

$$\{\!|\, \mathrm{x} \,|\!\}_{\mathrm{xkey1}}, \{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{km} \oplus \mathrm{data}} \rightarrow \mathrm{x}$$ **(Decrypt Data)**

$$\{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{xkek1} \oplus \mathrm{xtype}}, \mathrm{xtype}, \{\!|\, \mathrm{xkek1} \,|\!\}_{\mathrm{km} \oplus \mathrm{imp}} \{\!|\, \mathrm{xkek2} \,|\!\}_{\mathrm{km} \oplus \mathrm{exp}} \rightarrow \{\!|\, \mathrm{xkey1} \,|\!\}_{\mathrm{xkek2} \oplus \mathrm{xtype}}$$ **(Translate Key)**

Let R_{API1} be the set of rules listed above. It models the 4748 CCA API initial protocol. Let R_{API2} be the set of the rules in R_{API1} minus the Key Import command. Let R_{API3} be the set of rules in R_{API1} minus all the Key Part Import commands (1–3). To model the second fix proposed by IBM, we have to check that both R_{API2} and R_{API3} are secure.

Intruder Abilities

$$\begin{array}{rl} x, y \rightarrow \{\!|\, x \,|\!\}_y & \text{encryption} \\ \{\!|\, x \,|\!\}_y, y \rightarrow x & \text{decryption} \\ x, y \rightarrow x \oplus y & \text{xoring} \end{array}$$

Let R_I be the set of rules listed above. They model the intruder capabilities.

Initial Knowledge of intruder

Control vectors:

$$\text{kp, imp, exp, data, pin}$$

PIN key encrypted for transfer

$$\{\!|\, \mathrm{pdk} \,|\!\}_{\mathrm{kek} \oplus \mathrm{pin}}$$

An account number

$$\mathrm{pan}$$

For the original API (R_{API1}), where an intruder potentially has access to the Key Part Import (3) command as well and Key Import, the initial knowledge includes the following:
A key part:

$$\mathrm{k3}$$

A partially completed key (result of Key Part Import (2)):

$$\{\!|\, \mathrm{kek} \oplus \mathrm{k3} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{imp}}$$

Let S_1 be the set of terms listed above. It corresponds to the initial knowledge of the intruder in the first version of the protocol.

To verify IBM's second fix, we must consider two models, with differing initial knowledge. In the first (R_{API2}) the user (and hence the intruder) has the same initial knowledge as the original intruder, i.e. $S_2 = S_1$. In the second model (R_{API3}), the intruder does not have the partially completed key, but he does know the completed key. So let $S_3 = (S_1 - \{\{\!|\, \mathrm{kek} \oplus \mathrm{k3} \,|\!\}_{\mathrm{km} \oplus \mathrm{kp} \oplus \mathrm{imp}}\}) \cup \{\{\!|\, \mathrm{kek} \,|\!\}_{\mathrm{km} \oplus \mathrm{imp}}\}$.

The Secrecy Property

The property is the security of customer PINs, which are obtained by encrypting their account numbers under the PIN derivation keys. So the term that must remain a secret is $\{\!|\, \mathrm{pan} \,|\!\}_{\mathrm{pdk}}$.

For the initial (flawed) protocol, the secrecy property can be stated as:

$$S_1 \overset{?}{\vdash}_{R_{\mathsf{API1}} \cup R_I} \{\!|\, \mathrm{pan} \,|\!\}_{\mathrm{pdk}}$$

For the corrected protocol, there are two secrecy properties:

$$S_2 \overset{?}{\vdash}_{R_{\mathsf{API2}} \cup R_I} \{\!|\, \mathrm{pan} \,|\!\}_{\mathrm{pdk}}$$

$$S_3 \overset{?}{\vdash}_{R_{\mathsf{API3}} \cup R_I} \{\!|\, \mathrm{pan} \,|\!\}_{\mathrm{pdk}}$$