Flat and One-Variable Clauses for Single Blind Copying Protocols: the XOR Case

Helmut Seidl Kumar Neeraj Verma

TU München, Germany

RTA 2009

Single blind copying in cryptographic protocols

The Needham-Schroeder public key example:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, N_a\}_{K_b} \\
2. & B \longrightarrow A : \{N_a, N_b\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

For our modeling we consider safe abstractions: unbounded number of sessions, nonces may be non-fresh.

Modeling intruder's knowledge

1.	$A \longrightarrow B : \{A, N_a\}_{K_b}$	$I(\{A, N_a\}_{K_b})$
2.	$B \longrightarrow A : \{N_a, N_b\}_{K_a}$	$\neg I(\{A, x\}_{K_b}) \vee I(\{x, N_b\}_{K_a})$
3.	$A \longrightarrow B : \{N_b\}_{K_b}$	$\neg I(\{N_a, x\}_{K_a}) \vee I(\{x\}_{K_b})$

Secrecy of N_b : $\neg I(N_b)$.

$\Rightarrow$ Clauses with at most one variable.

We abstracted all the nonces to only finitely many.

Less severe (still safe) abstractions are also possible.

$$
\begin{array}{lll}
1. & A \longrightarrow B : \{A, N_a\}_{K_b} & \ldots \\
2. & B \longrightarrow A : \{N_a, N_b\}_{K_a} & \neg I(\{A, x\}_{K_b}) \vee I(\{x, N_b(x)\}_{K_a}) \\
3. & A \longrightarrow B : \{N_b\}_{K_b} & \ldots
\end{array}
$$

The generated nonce is now a function of the received nonce (in the style of [Blanchet01])

This is still a one-variable clause.

Other abilities of the intruder

$I(\mathtt{encrypt}(x, y)) \vee \neg I(x) \vee \neg I(y)$	Intruder can encrypt messages
$I(\mathtt{pair}(x, y)) \vee \neg I(x) \vee \neg I(y)$	Intruder can form pairs
$I(x) \vee \neg I(\mathtt{encrypt}(x, y)) \vee \neg I(y)$	Intruder can decrypt messages
$I(x) \vee \neg I(\mathtt{pair}(x, y))$	Intruder can unpair messages
$I(y) \vee \neg I(\mathtt{pair}(x, y))$	

$\Rightarrow$ Flat clauses

The class $\mathcal{C}$ of Comon-Lundh and Cortier

– Clauses with at most one variable

– Flat clauses: $\bigvee_i \pm_i P_i(f_i(x_i^1, \ldots, x_i^{n_i})) \vee \bigvee_j \pm_j Q_j(x_j)$
for each i, $\{x_i^1, \ldots, x_i^{n_i}\}$ are all the variables in the clause

The class $\mathcal{C}$ of Comon-Lundh and Cortier

– Clauses with at most one variable

– Flat clauses: $\bigvee_i \pm_i P_i(f_i(x_i^1, \ldots, x_i^{n_i})) \vee \bigvee_j \pm_j Q_j(x_j)$
for each i, $\{x_i^1, \ldots, x_i^{n_i}\}$ are all the variables in the clause

Previous upper bound:	3-DEXPTIME
Known lower bounds:	NEXPTIME
	DEXPTIME in Horn case
Our techniques give upper bounds:	NEXPTIME
	DEXPTIME in Horn case

$\Rightarrow$ The secrecy problem is DEXPTIME . . . even DEXPTIME-complete!

Some standard techniques

The binary resolution rule:

$$\frac{C_1 \vee P(s) \quad \neg P(t) \vee C_2}{C_1\sigma \vee C_2\sigma}\ (\sigma = mgu(s,t))$$

Soundness and completeness: the empty clause $\Box$ (false) can be derived iff the given set of clauses is unsatisfiable.

A general technique which decides various fragments of first-order logic, including two-variable fragment, guarded fragment, . . .

Example

Input clauses

$P(a) \qquad P(f(x)) \Leftarrow P(x) \qquad \neg P(f(f(a)))$

Resolution produces

$P(f(a)) \qquad P(f(f(a))) \qquad \Box$

$\Rightarrow$ The set of clauses is unsatisfiable.

Example

Input clauses

$P(a) \qquad P(f(x)) \Leftarrow P(x) \qquad \neg P(f(f(a)))$

Resolution produces

$P(f(a)) \qquad P(f(f(a))) \qquad \square$

$\Rightarrow$ The set of clauses is unsatisfiable.

Problem: non-termination in case of satisfiability.

Ordered resolution

Select only the maximal literals in a clause during resolution.

Soundness and completeness are preserved.

Input clauses

$P(a) \qquad P(f(x)) \Leftarrow P(x) \qquad \neg P(f(f(a)))$

Resolution produces:

$\neg P(f(a)) \qquad \neg P(a) \qquad \Box$

Ordered resolution, for a suitable ordering, on the class $\mathcal{C}$ leads to a linear bound on the height of terms in produced clauses.

$\Rightarrow$ 3-EXPTIME decision procedure [Comon-Lundh, Cortier]

This analysis is too coarse to obtain optimal complexity.

We are going to use different algorithms, instead of reanalyzing the same algorithm.

One-variable clauses

Generalize alternating pushdown systems on strings:

$$\begin{aligned} &P(a) \\ &P(f_1(f_2(f_3(x)))) && \Leftarrow Q(g_1(g_2(x))) \\ &P(x) && \Leftarrow P_1(x) \wedge P_2(x) \end{aligned}$$

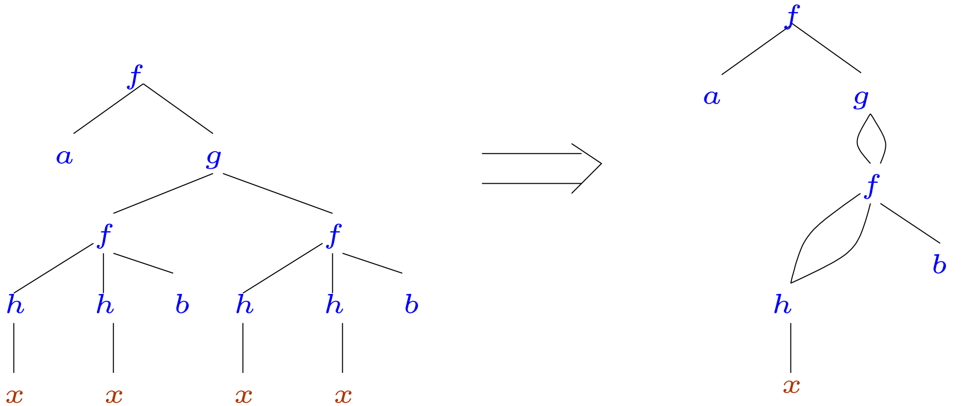
We now allow arbitrary arities and repetition of variables.

$$P(f(x, g(h(a), g(x, x)))) \Leftarrow Q(x) \wedge R(f(x, x))$$

Decomposition of one-variable terms

One variable terms are composed of irreducible terms.

$s[x] = s_1[\ldots[s_n[x]]\ldots]$.



$= f(a, x)[g(x, x)[f(x, x, b)[h(x)]]]$

$\Rightarrow$ One-variable terms behave like strings.

$\Rightarrow$ Satisfiability for one-variable clauses is DEXPTIME-complete.
(As for alternating pushdown systems)

Flat clauses

$$P(f(x,y,x)) \Leftarrow Q(g(y,y,x,y)) \wedge R(x) \wedge S(y) \wedge T(y) \wedge U(h(x,y))$$

Generalize alternating two-way automata, equality constraints between brothers, permutation and repetition of arguments.

NEXPTIME-completeness (DEXPTIME-completeness in the Horn case) is well-known for various restricted cases:
$\star$ either the maximal arity is a constant
$\star$ or the same sequence of variables occurs in all non-trivial atoms in a clause

We show the same complexity for the general case.

Idea: resolution modulo propositional reasoning

The resolution step

$$\frac{P(x) \Leftarrow Q(f(x,x)) \qquad Q(f(x,y)) \Leftarrow R(y)}{P(x) \Leftarrow R(x)}$$

can be broken into an instantiation step

$$\frac{Q(f(x,y)) \Leftarrow R(y)}{Q(f(x,x)) \Leftarrow R(x)}$$

and a propositional implication generation step

$$\frac{D_1 \vee L \qquad \neg L \vee D_2}{D_1 \vee D_2}$$

$\Rightarrow$ Generate interesting propositional implications, and avoid intermediate clauses.

Use the fact that propositional satisfiability is in NP.

Optimal complexity results for several classes:

	General case	Horn case
One-variable	DEXPTIME-complete	DEXPTIME-complete
Flat clauses	NEXPTIME-complete	DEXPTIME-complete
Combination	NEXPTIME-complete	DEXPTIME-complete

Secrecy of cryptographic protocols with single blind copying is DEXPTIME-complete.

Extension: adding the XOR theory

Algebraic properties of cryptographic primitives often need to be considered for a precise analysis.

Frequently occurring properties include those of associativity and commutativity, properties of modular exponentiation, XOR,...

We consider the XOR theory:

$$
\begin{aligned}
x+(y+z) &= (x+y)+z \\
x+y &= y+x \\
x+0 &= x \\
x+x &= 0
\end{aligned}
$$

We generalize our clauses.

- Arbitrary one-variable clauses, possibly containing the XOR symbol
- Flat clauses, without the XOR symbol
- One intruder clause $I(x+y) \Leftarrow I(x) \wedge I(y)$

We generalize our clauses.

- Arbitrary one-variable clauses, possibly containing the XOR symbol
- Flat clauses, without the XOR symbol
- One intruder clause $I(x+y) \Leftarrow I(x) \wedge I(y)$

Allowing arbitrary many clauses of the form

$$P_1(x+y) \Leftarrow P_2(x) \wedge P_3(y)$$

leads to undecidability.

Problem 1: no stable ordering

Usual orderings useful for ordered resolution don't work in the XOR case.

With the subterm ordering we have

$$x < f(x+f(f(0)))$$

But applying the substitution $x \mapsto f(f(0))$, we must have:

$$f(f(0)) < f(0)$$

Solution: the substitution $x \mapsto f(f(0))$ involves only ground subterms from the input set. Do all such problematic substitutions separately, and then do ordered resolution.

Problem 2: The intruder clause resolves with itself to give larger and larger clauses.

$$\frac{I(x+y) \Leftarrow I(x) \wedge I(y) \quad I(x'+y') \Leftarrow I(x') \wedge I(y')}{I(x+x'+y') \Leftarrow I(x) \wedge I(x') \wedge I(y')}$$

Problem 2: The intruder clause resolves with itself to give larger and larger clauses.

$$\frac{I(x+y) \Leftarrow I(x) \wedge I(y) \quad I(x'+y') \Leftarrow I(x') \wedge I(y')}{I(x+x'+y') \Leftarrow I(x) \wedge I(x') \wedge I(y')}$$

Solution: replace this with special deduction rules, e.g.:

$$\frac{P(x) \Leftarrow I(f(x)+g(x)) \quad I(f(x)+h(x)) \Leftarrow Q(x)}{P(x) \Leftarrow I(g(x)+h(x)) \wedge Q(x)}$$

Problem 2: The intruder clause resolves with itself to give larger and larger clauses.

$$\frac{I(x+y) \Leftarrow I(x) \wedge I(y) \quad I(x'+y') \Leftarrow I(x') \wedge I(y')}{I(x+x'+y') \Leftarrow I(x) \wedge I(x') \wedge I(y')}$$

Solution: replace this with special deduction rules, e.g.:

$$\frac{P(x) \Leftarrow I(f(x)+g(x)) \quad I(f(x)+h(x)) \Leftarrow Q(x)}{P(x) \Leftarrow I(g(x)+h(x)) \wedge Q(x)}$$

Elementary decision procedure :-))

Conclusion

- General techniques from automata theory and automated deduction help in verification of cryptographic protocols.
- Precise complexity for our XOR class not yet known.