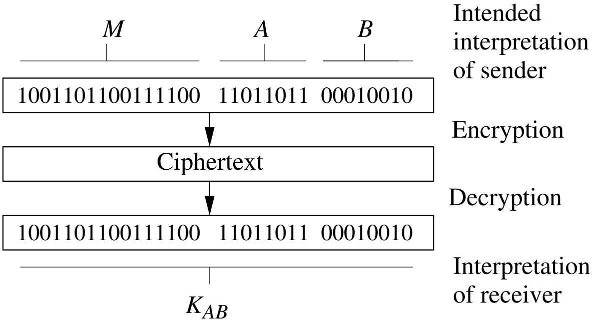
My Own Little Hilbert's Program

Sebastian Mödersheim
Danmarks Tekniske Universitet
samo@imm.dtu.dk

Based on joint work with:
Thomas Groß, University of Newcastle
Luca Viganò, Università di Verona
Hanne Riis Nielson & Flemming Nielson, DTU

Séminaire de Cryptologie, 18.10.2012

Type-Flaw Attacks



- Confusion about the interpretation of messages
- Lower-level/implementation problems
- Often not so interesting
- Get in the way in verification of larger systems
- Can we abstract from them?

"Typed model" in symbolic protocol verification

- Different from standard type systems like Hindley-Milner
- Receivers magically can check correct format/type of messages
- Relative soundness ([Heather et al], [Malladi & Lafourcade], [Arapinis & Duflot])
  "If there is an attack, then there is also a well-typed attack."
- Helpful: complexity/decidability

Protocol Composition

- Disjointness conditions: message parts of the different protocols can be distinguished. [Guttman], [Cortier & Delaune], [Groß & M.]
- Similar requirements & proofs as for typing — protocol types

"Typed" Diffie-Hellman

- Agents magically can check that received half-keys are of the form $\exp(g, X)$
- Soundness result for a certain class of protocols ([M.], [Lynch & Meadows])

Hilbert's Program

"...to recognize our common methods of mathematics in their entirety as consistent."

Some say this program has failed, but in a brilliant way!

My Hilbert's Program

"...to recognize our common models of security in their entirety as typeable."

My Hilbert's Program

- Relate questions of typing, compositional and algebraic reasoning.
- Good engineering practice anyway:
  non-atomic message parts with different meaning must be distinguishable.
- Abstract from lower-level details like parsing.

Challenge: prove this program to fail as well!

Outline

1. **Typeability**
2. **Disjointness in Composition**
3. **Typing for Diffie-Hellman**
4. **Typeable ASLan**
5. **Lazy Mobile Intruders**
6. **Conclusions & Outlook**

Outline

1. **Typeability**
2. Disjointness in Composition
3. Typing for Diffie-Hellman
4. Typeable ASLan
5. Lazy Mobile Intruders
6. Conclusions & Outlook

Message Model

Operations

E.g. crypt, scrypt, exp, $[\cdot]_n$, ...

- Every agent, including intruder, can apply these operations.

Abstracting from parsing problems: n-ary concatenation $[\cdot]_n$.

Mappings

E.g. inv, pk, sk, ...

- Define atomic terms as the closure of all constants and variables under the mapping functions.

Typed Protocol Models

Atomic Types

Typed variables can only be substituted with atomic terms of that type.

Composed Types

Closure of atomic types under operation symbols, e.g.

$$M : \mathrm{scrypt}(\mathit{hash}(\mathit{nonce}, \mathit{nonce}), [\mathit{nonce}, \mathit{nonce}]) \ .$$

Semantics: replace M by appropriate term with fresh vars.

A Typing Result

similar to [Heather et al] and [Arapinis & Duflot]

Well-Designed Protocols

- AVISPA IF protocol P without negative facts and conditions
- Interpreted in free algebra
- Type annotation Γ for all variables
- $MP(P)$ set of all message patterns of honest agents
- $SMP(P)$ non-atomic subterms of $MP(P)$
- P is *well-designed* (w.r.t. Γ): no two elements of $SMP(P)$ that have different composed type can be unified

Theorem

Then if P has an attack, it also has a well-typed attack.

Proof idea: insertion of ill-typed messages never helps the intruder.

The Lazy Intruder

Symbolic/Constraint-based Approach

[Huima], [Amadio & Lugiez], [Millen & Shmatikov], [Chevalier & Vigneron], [Basin & M. & Viganò],[Delaune et al.]...

- Avoid exploration of intruder choices by collecting symbolic constraints $M \vdash m$.
- Correct and terminating reduction calculus for several algebraic theories.

The Lazy Intruder: Example

Example: PKInit Kerberos

$$\begin{array}{rl} C \to Auth: & [C, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_C}, N)] \\ Auth \to C: & \mathrm{crypt}(\mathrm{pk}_C, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_{Auth}}, Ktemp)), \mathrm{scrypt}(Ktemp, \ldots) \end{array}$$

Intruder as dishonest client C' with initial knowledge $K_0 = \{\mathrm{pk}_C, \mathrm{pk}_{Auth}, \mathrm{pk}_{C'}, \mathrm{inv}_{\mathrm{pk}_{C'}}\}$:

$$\begin{array}{lll} K_1 & := & K_0 \cup \{[C, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_C}, N)]\} \\ K_1 & \vdash & [X, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_X}, Y)] \\ K_2 & := & K_1 \cup \{\mathrm{crypt}(\mathrm{pk}_X, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_{Auth}}, Ktemp)), \mathrm{scrypt}(Ktemp, \ldots)\} \\ K_2 & \vdash & \mathrm{crypt}(\mathrm{pk}_C, \mathrm{crypt}(\mathrm{inv}_{\mathrm{pk}_{Auth}}, Z)), \mathrm{scrypt}(Z, \ldots) \end{array}$$

The Lazy Intruder as a Proof Technique

- If a protocol has an attack, then there are well-formed $M \vdash m$ constraints, such that every solution represents an attack.
- The usual constraint reduction is complete
- For a well-designed protocol, the constraint reduction never performs an ill-typed substitution.
  - ★ Substitutions occur only when unifying two terms — and these terms cannot be variables.

Outline

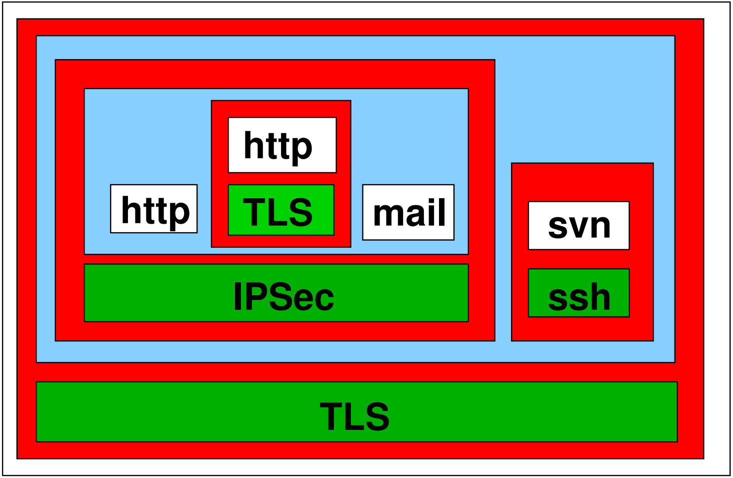
1. Typeability
2. **Disjointness in Composition**
3. Typing for Diffie-Hellman
4. Typeable ASLan
5. Lazy Mobile Intruders
6. Conclusions & Outlook

Typing in Compositional Reasoning

Similar results for parallel composition of protocols:

- Let protocols P_1 and P_2 be are disjoint:
  - ★ No message of $SMP(P_1)$ can be unified with a message of $SMP(P_2)$.
- Protocols have same set of long-term public values
- There are no side-channels such as databases
- Thm: P_1 and P_2 can safely be run in parallel.
  Proof: lazy intruder never unifies P_1 with P_2 terms.
- Seen as typing: messages have either type P_1 or type P_2

Example of Vertical Composition



Is this secure?

An Extended Notion of Disjointness

Definition (Message Patterns Modulo Encryption)

again, everything under appropriate α-renaming

$$
\begin{aligned}
MP(P) &: \text{ message patterns of protocol } P \\
EMP_0(P) &= MP(P) \\
EMP_{n+1}(P) &= \{\text{scrypt}(K_{n+1}, m) \mid m \in EMP_n(P)\} \\
EMP(P) &= \bigcup_{n \in \mathbb{N}} EMP_n(P) \\
EST(P) &: \text{ non-atomic subterms of } EMP(P)
\end{aligned}
$$

Protocols must be disjoint from their own encryptions:

- no unifier between patterns of $EMP_i(P)$ and $EMP_j(P)$ for $i \neq j$.

Protocols must be disjoint from each other modulo encryptions:

- no unifier between patterns of $EST(P)$ and $EST(Q)$ for $P \neq Q$

Vertical Protocol Composition [Groß & M.]

Protocol suite $\mathcal{P}$ that satisfies all conditions so far:

In an arbitrary composition of protocols of $\mathcal{P}$, every non-atomic message part can be attributed to a unique protocol stack.

Gives rise to extended notion of protocol types such as

$$http\langle TLS\rangle\langle IPSec\rangle\langle TLS\rangle$$

Theorem

Every composition of protocols in $\mathcal{P}$ is secure.

Outline

1. Typeability
2. Disjointness in Composition
3. **Typing for Diffie-Hellman**
4. Typeable ASLan
5. Lazy Mobile Intruders
6. Conclusions & Outlook

Diffie-Hellman and Typeability

Unrealistic Diffie-Hellman

$$A \to B : \mathsf{crypt}(\mathsf{inv}_{\mathsf{pk}_A}, [B, \exp(g, X)]_2)$$
$$\ldots$$

B's message pattern for receiving:

$$\mathsf{crypt}(\mathsf{inv}_{\mathsf{pk}_A}, [B, \color{red}{\exp(g, X)}]_2)$$

Actually B cannot check the "format" of the red part.

- This means a significant state space reduction
- Can we justify such a model?

Diffie-Hellman without Difficulty

Requirements for our soundness result

- Already part of typeability:
  half-keys are distinguished from other message parts.
- Fresh exponents for every exchange.
- Exponents only used as
  - ★ Half-key: $\mathrm{exp}(g, X)$
  - ★ Full-key in a symmetric encryption: $\mathrm{scrypt}(\mathrm{exp}(GX, Y), \cdot)$
    where GX received as half-key
- Exponentiation does not occur elsewhere

Captured as additional well-formedness conditions for the lazy intruder.

Diffie-Hellman without Difficulty

Theorem

Said class of Diffie-Hellman protocols is typeable.

Proof:

- Modified notion of simple lazy intruder constraints:

Additionally $\exp(GX, x)$ is considered as simple (will not be reduced) if the constraint introducing GX is already simple.

- Related restriction on unifying terms
- Proof that resulting lazy intruder is still complete
- . . . and does not perform ill-typed substitutions.

Millen's Theory

Abstract Diffie-Hellman to primitives for half-key (kap) and full-key (kas):

$$kas(kap(X), Y) \approx kas(kap(Y), X)$$

- Typeability implies that this model is sound relative to our original Diffie-Hellman model.
- This allows for encoding into free algebra by case distinction. (Similar to [Küsters & Truderung])
- Case distinction can even be avoided: ensure type initiator and type responder in half-key exchange!

Outline

1. Typeability
2. Disjointness in Composition
3. Typing for Diffie-Hellman
4. **Typeable ASLan**
5. Lazy Mobile Intruders
6. Conclusions & Outlook

ASLan: AVANTSSAR specification language

- Combination of two concepts:
  - ★ AVISPA IF: infinite-state transition system, based on set-rewriting
  - ★ Horn clauses: immediate evaluations at every state
- Useful for modeling of complex systems (and properties):
  - ★ Classic example: intruder deduction in a protocol.
  - ★ Integration of workflow with policies
  - ★ Participants maintain databases and make internal computations in these databases.
  - ★ Dynamic virtualized infrastructure with information flow evaluations.
- Undecidability: Horn clauses alone allow for logic programming

Example

Declarations:

$mem : \text{pred}\ (agent, gid)$

$deputy : \text{pred}\ (agent, agent)$

$attack : \text{pred}\ ()$

$G, G1, G2, g1, g2 : gid$

$own : \text{pred}\ (gid, fid)$

$xs : \text{pred}\ (agent, fid)$

$A, B, a, b : agent$

$F, F1, F2, f1, f2 : fid$

Initial State:

$$mem(a, g1) \wedge mem(b, g2) \wedge own(g1, f1) \wedge own(g2, f2)$$

Transition Rules:

$$mem(A, G1) \wedge \neg\exists B : deputy(A, B) =\!\![G2]\!\!\Rightarrow mem(A, G2)$$

$$xs(A, F1) \wedge xs(A, F2) \wedge own(G1, F1) \wedge own(G2, F2) \wedge G1 \neq G2 \Rightarrow attack()$$

Horn clauses:

$$mem(A, G) \wedge own(G, F) \rightarrow xs(A, F)$$

$$deputy(A, B) \wedge xs(B, F) \rightarrow xs(A, F)$$

Example

mem(a,g1)
mem(b,g2)
own(g1,f1)
own(g2,f2)

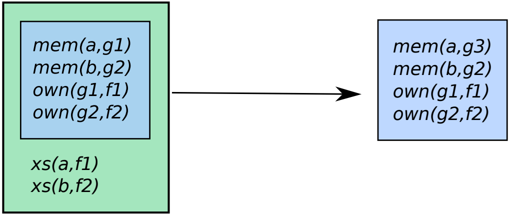
Example

mem(a,g1)
mem(b,g2)
own(g1,f1)
own(g2,f2)

xs(a,f1)
xs(b,f2)

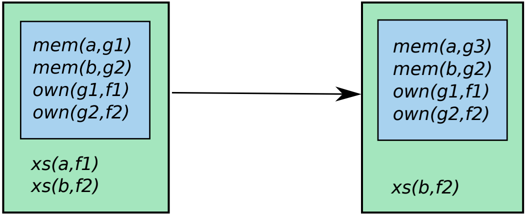
$$mem(A, G1) \land \neg\exists B : deputy(A, B) =\![G2]\!\!\Rightarrow mem(A, G2)$$

Example



$$mem(A, G1) \wedge \neg \exists B : deputy(A, B) =\![G2]\!\Rightarrow mem(A, G2)$$

Example



$$mem(A, G1) \land \neg\exists B : deputy(A, B) =\!\!\![G2]\!\!\Rightarrow mem(A, G2)$$

Example

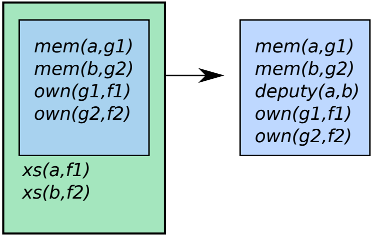
mem(a,g1)
mem(b,g2)
own(g1,f1)
own(g2,f2)

Example

mem(a,g1)
mem(b,g2)
own(g1,f1)
own(g2,f2)

xs(a,f1)
xs(b,f2)

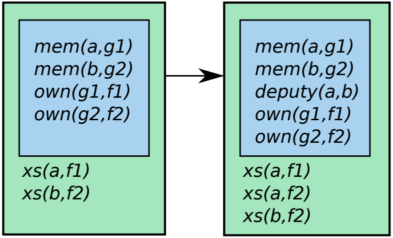
Example



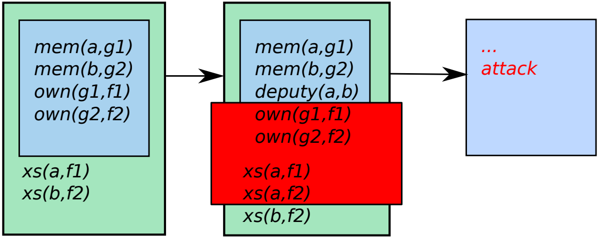
Bob assigns a deputy (simplified):

$mem(A, G1) \wedge mem(B, G2) \Rightarrow$
$mem(A, G1) \wedge mem(B, G2) \wedge deputy(A, B)$

Example


mem(a,g1)
mem(b,g2)
own(g1,f1)
own(g2,f2)
xs(a,f1)
xs(b,f2)
mem(a,g1)
mem(b,g2)
deputy(a,b)
own(g1,f1)
own(g2,f2)
xs(a,f1)
xs(a,f2)
xs(b,f2)

Example



$$xs(A, F1) \wedge xs(A, F2) \wedge own(G1, F1) \wedge own(G2, F2) \wedge G1 \neq G2 \Rightarrow attack()$$

A Symbolic Representation

Symbolic state:

$$\begin{array}{lll} \phi & ::= & \text{Symbolic state} \\ & P & \quad\text{Predicate} \\ & S \vdash P & \quad\text{Deduction constraint} \\ & \neg\exists \vec{X} : s_1 = t_1 \wedge \ldots \wedge s_n = t_n & \quad\text{Negated substitution} \\ & X = t & \quad\text{Substitution} \\ & \phi \wedge \psi & \quad\text{Conjunction} \end{array}$$

Lemma

This allows for a finitely branching symbolic transition relation.

Symbolic State Satisfiability

Constraint type	*Satisfiability*
Intruder deduction constraints $\mathrm{ik}(t_1) \wedge \ldots \wedge \mathrm{ik}(t_n) \vdash \mathrm{ik}(t)$	Lazy Intruder (NP-complete)
Other deduction constraints $S \vdash P$	Undecidable in general
Negated substitutions $\neg\exists\vec{X} : s_1 = t_1 \wedge \ldots \wedge s_n = t_n$	Unification after substituting bound variables with fresh values
Substitutions $X = t$	(Just bookkeeping)

TASLan

SMP: non-atomic subterms of $\mathrm{ik}(t)$ of transition rules.

Definition

TASLan is ASLan with following requirements/modifications:

- Classical disjointness: If $t_1, t_2 \in SMP$ have a unifier, then t_1 and t_2 must have the same type.
- Type untyped occurs only in intruder deduction.

For instance you cannot have:

$$\begin{array}{l} p(z) \\ \forall N.p(N) \to p(s(N)) \end{array}$$

For many specifications these restrictions are fine!

Results

Theorem

For TASLan the following problems are decidable:

- *Satisfiability of symbolic states*
- *Reachability of an attack state in l steps (NEXPTIME-complete)*

If a symbolic state is satisfiable, then it is satisfiable under a well-typed interpretation.

Proof.

In TASLan:

- Applying Horn clauses does not introduce ill-typed substitutions.
- The lazy intruder never introduces ill-typed substitutions.
- This allows for a convergent evaluation relation for the $S \vdash P$.

□

Outline

1. Typeability
2. Disjointness in Composition
3. Typing for Diffie-Hellman
4. Typeable ASLan
5. **Lazy Mobile Intruders**
6. Conclusions & Outlook

Lazy Mobile Intruders

Problem

Given:

- Platform $C[\cdot]$ that executes potentially malicious code
- Initial intruder knowledge K_0
- Attack predicate $attack(S)$

Question: exist code P and state S such that

- $K_0 \vdash P$
  The intruder can generate the code from his initial knowledge
- $attack(S)$
- $C[P] \to^* S$
  The platform can reach an attack state when executing P.

Obviously we cannot search the space of all programs $K_0 \vdash P$!

Lazy Mobile Intruders

Idea: [M.& Nielson & Nielson]

- Generate the code in a demand-driven, lazy way
- Code is initially $\boxed{\mathsf{K}}$ for knowledge K.
- Determine code step by step by what transitions are possible
- When reading message: augment K
- When writing message: $K \vdash x$
- When two intruder processes meet: pool knowledge
- Analyze what locations the intruder code can reach and with what knowledge
- Decision procedure for mobile ambient calculus without replication:
  - ★ Platform can perform only bounded number of steps
  - ★ Intruder code not bounded

Outline

1. Typeability
2. Disjointness in Composition
3. Typing for Diffie-Hellman
4. Typeable ASLan
5. Lazy Mobile Intruders
6. **Conclusions & Outlook**

Conclusions & Outlook

The Lazy Intruder is a versatile idea

- Protocol Verification
  - ★ Bounded sessions, several algebraic theories
  - ★ Now even unbounded case... [Guttman et al]
  - ★ Typeable ASLan: transition system + Horn clauses
- Relative soundness results:
  - ★ Different notions of typing: messages, protocols, channels
  - ★ Reduce complex verification problems to smaller ones in algebraic and compositional reasoning
  - ★ Proofs are similar, abusing the lazy intruder as an argument
  - ★ Exploiting what is good engineering practice anyway!
- Lazy invention of intruder-generated code

A lot left to do:

- Less restrictive assumptions, more algebraic theories
- Broader scope of channel properties for compositional reasoning
- Privacy/unlinkeability