

Spi2F # Code Generator

Generating secure code

Outline

- Motivation
- Sample
- My Bachelor thesis
- Demo

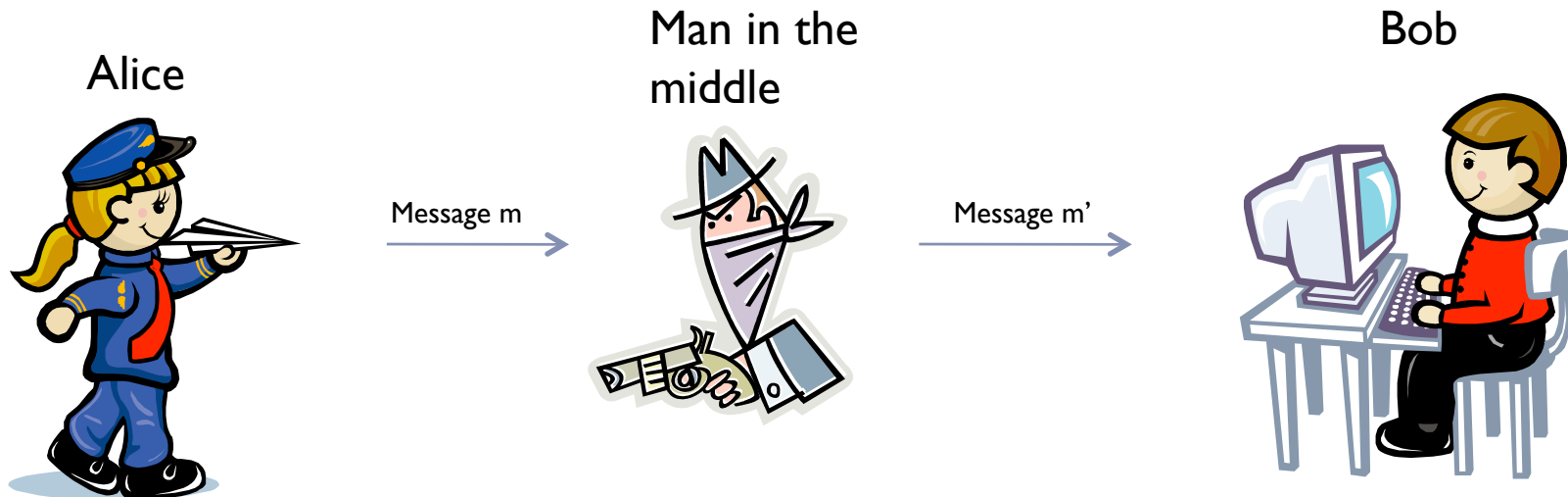
Sample protocol

- ▶ Message exchange

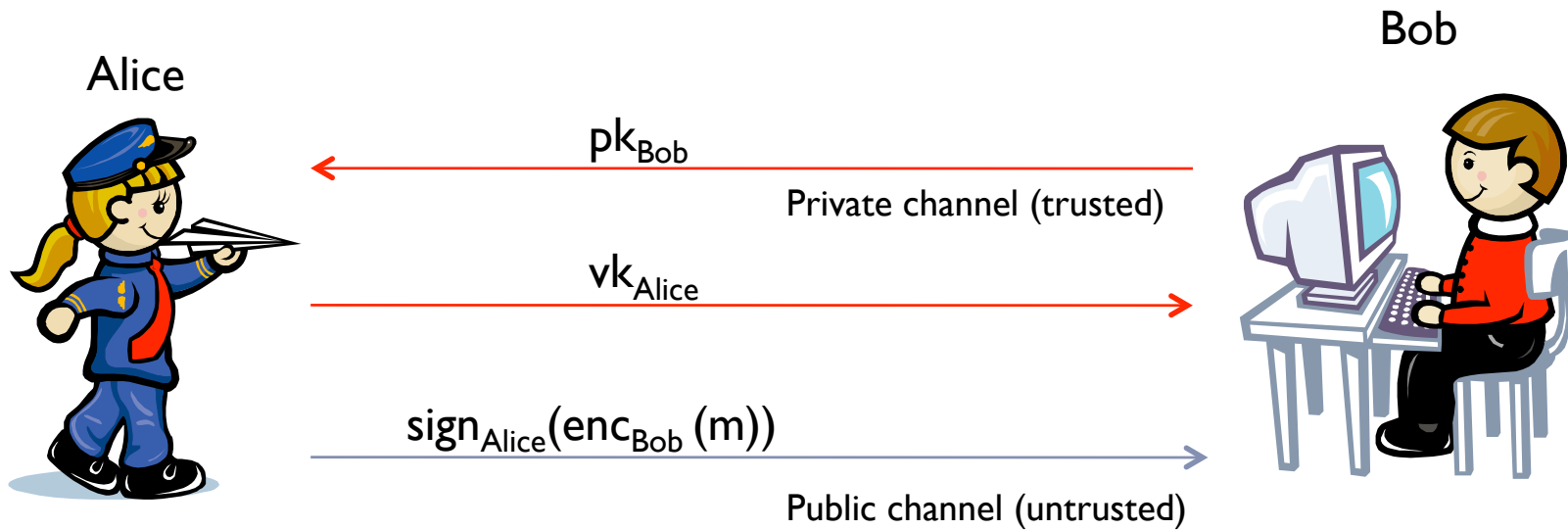


Attack

- ▶ Man in the middle could change message easily (no authentication)
- ▶ Confidentiality can also not be guaranteed



Countermeasures



Formalisation

- ▶ **Needed because informal protocols can contain flaws**
 - ▶ Recent MITM attack on public-key Kerberos (2005)
- ▶ **Formal protocols can be proven to be secure**
 - ▶ In our case this can be done through a typechecker

Sample.spi

```
predicate Authentic(*1*).

free c1 (*: Ch PubKey(<x1:Private>[*#Authentic(x1)*]) *).
free c2 (*: Ch VerKey(PubEnc(<x1:Private>[*#Authentic(x1)*])) *).
free cm (*: Ch Un *).

let alice =
  new sigA (*: SigKey(PubEnc(<x1:Private>[*#Authentic(x1)*])) *);
  in(c1, pkB);
  out(c2, vk(sigA));
  new m (*: Private *);
  (assume (*#Authentic(m)*)
   | out(cm,sign(enc(<m(*=x*)>[*#Authentic(x)*],pkB),sigA))).

let bob =
  new privB (*: PrivKey(<x1:Private>[*#Authentic(x1)*]) *);
  out(c1, pk(privB));
  in(c2, vkA);
  in(cm, m);
  let m1 = check(m,vkA) in
  let m2 = dec(m1,privB) in
  let <m3> = m2 in
  assert (*#Authentic(m3)*).

process
  (bob | alice)
```

Code generation

- ▶ Manually implementing the protocol is a weak link
- ▶ Code generators can generate implementations in a more reliable manner
- ▶ Code generators can hardly be proven to be correct
- ▶ → Prove generated code to be secure
 - ▶ Hardly possible in Java

Why F#

- ▶ Functional programming language developed by MS Research
 - ▶ Based on Ocaml
 - ▶ Uses .net libraries
- ▶ MS Research conducted research into verifying the security of F# code
 - ▶ First try: Extract model from F code
 - ▶ Not always decidable [Bhargavan et al., MSR-TR-2006-46]
 - ▶ Second try: use types to verify RCF code
 - ▶ Like spi types [Bengtson et al., CSF 2008]

Library implementations

Symbolic

- ▶ Programme works, but it is an abstraction
- ▶ Eg send only stores value in a local variable, but does not actually send anything
- ▶ Can be verified by F7 checker
- ▶ Useful for debugging and prototyping

Concrete

- ▶ Uses .NET functions to actually execute the operations
- ▶ Eg send actually sends a package over the network
- ▶ We have to trust the implementation of the .NET functions to be correct

- ▶ Implementations have to be in sync

My bachelor thesis

1. Write a code generator from Spi calculus to F#
2. Define a translation between Spi types and RCF types
3. Prove this translation to be correct, that is, we want to show that typing is preserved

1. Write a code generator

- ▶ Spi syntax is close to RCF syntax
- ▶ Type inference needed for implementation types
 - ▶ What kind of message to expect on a channel?
 - ▶ What data is contained in a message of type Un or Private?

Demo

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Shows the file `types.fs` with the following F# code:

```
1 // Turn on the lightweight syntax
2 #light
3
4 // open some standard namespaces
5 open System
6 open Syntax
7
8 type Types = Bytes | Un | PairT | PrivKey | PubKey | Str
9
10 type private Binding = Bind of string * Types
11
12 exception private Found of Types
13
14 type Environment() =
15     let env = ref []
16     member e.add n t = env := Bind(n, t) :: !env
17     member e.get u =
18         try
19             (List.iter (fun a ->
20                 match a with
21                     | Bind(x, t) -> if String.compare x u = 0 then raise (Found t) else ()
22                     | _ -> () !env; raise (InvalidArgument ("cannot find " ^ u ^ " in env"))
23                 )
24             with
25                 Found t -> t
26
27     let env = new Environment()
28
29
30 let TypeTy p =
31     match p with
32     | TUn -> Un
33     | TPrivate -> Un
34     | TPubKey _ -> PubKey
35     | TPrivKey _ -> PrivKey
36     | TVerKey _ -> PubKey
37     | TSigKey _ -> PrivKey
38
39 let Type d =
```
- Output Window:** Shows the following error messages:

```
Show output from: Debug
A first chance exception of type 'System.Collections.Generic.KeyNotFoundException' occurred in FSharp.Core.dll
A first chance exception of type 'Types.FoundException' occurred in output.exe
A first chance exception of type 'Types.FoundException' occurred in output.exe
A first chance exception of type 'Types.FoundException' occurred in output.exe
The program '[624] output.exe: Managed' has exited with code 0 (0x0).
```
- Solution Explorer:** Shows a solution named 'Prototype' containing projects 'types.fs', 'main.fs', and 'sample.spi'.
- Properties Window:** Shows the 'Properties' view for the selected project.
- Status Bar:** Shows 'Ready' on the left and 'Ln 12 Col 33 Ch 33 INS' on the right.

Advanced generation problems

- ▶ **What kind of code to generate**
 - ▶ Most likely library is generated that is used by a host application
- ▶ **How to specify encryption / signing parameters**
 - ▶ Separate parameter in constructor that specifies a config set
 - ▶ Parameters to generated functions
 - ▶ Callback functions
- ▶ **How to deal with multithreading**
 - ▶ Identify callback thread

2./3. Define a translation between Spi type system and RCF type system

- ▶ **Goal:**All programmes that typecheck in Spi will typecheck in RCF
- ▶ **Semantics of types is somewhat different**
 - ▶ Eg Private is not in RCF type system

Summary

- ▶ Code generator will make it easier to implement a protocol
- ▶ F7 will be able to prove security properties
- ▶ Limitation because security depends on concrete library and host application
- ▶ Questions?

References

- ▶ Verified Interoperable Implementations of Security Protocols, Bhargavan et al., MSR-TR-2006-46
- ▶ Refinement Types for Secure Implementations, Bengtson et al., CSF 2008
- ▶ Analysing Security Protocols with Secrecy Types and Logic Programs, Abadi et al.