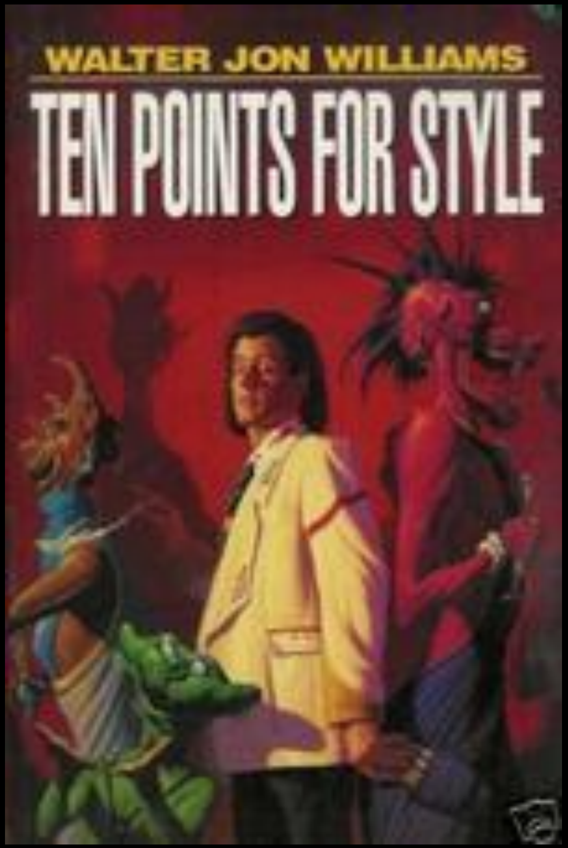
The Spi Calculus

Jamie Floyd

Points for Style

- This is (at least partially) a PL topic
  - But I strive to make it accessible
- Lesson or lecture – designed to teach
- Builds upon itself – will snowball
  - Ask questions!
- But, I will have to go fast…



What is the Spi Calculus?

- An extension of the **Pi Calculus** with cryptographic primitives.
- Pi + Crypto = Spi
- Developed in 1997 by Martin Abadi and Andrew Gordon

Outline

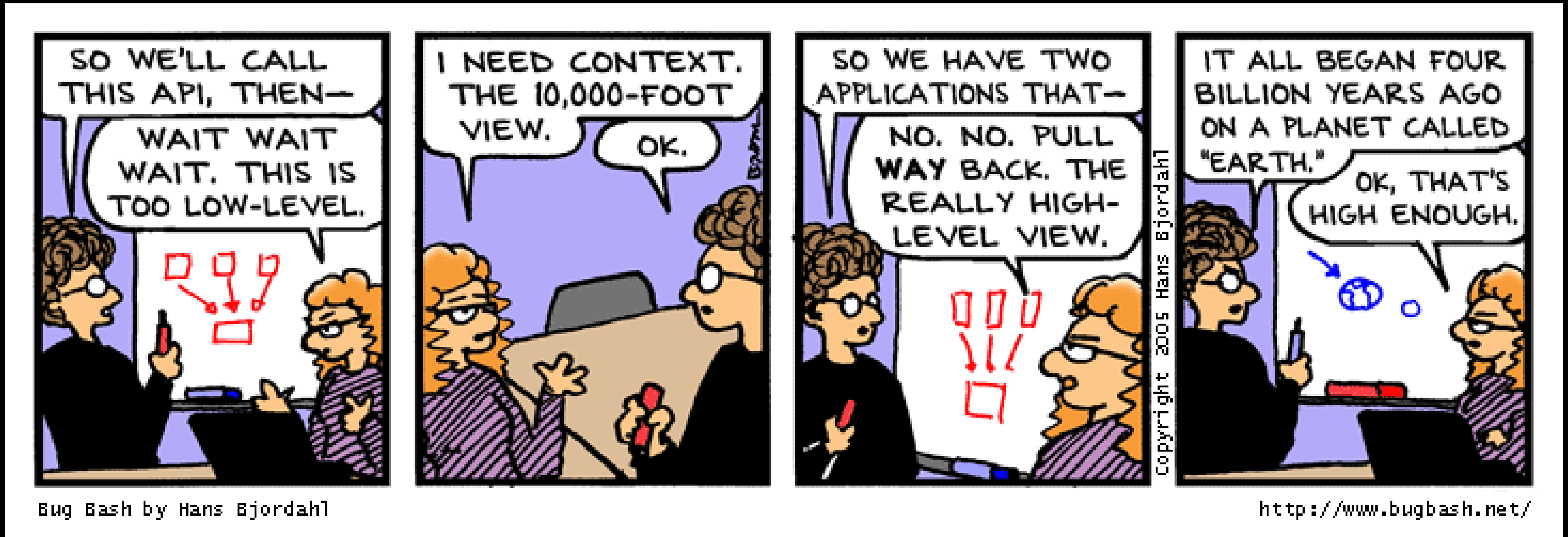
- Understand the **Pi Calculus**
  - Why is it important/useful?
  - Work through a couple simple security protocols using it
- Add in features and move to the **Spi Calculus**
  - Why is it helpful?
  - Extend the previous protocol
- Examine a new, *slightly* more complicated protocol in both the **Pi** and **Spi Calculi**

Process Calculi – Motivation and History

- Computer networks changed our model of computation
  - λ-calculus only works for sequential programs
- C.A.R. Hoare, 1978 – CSP (Communicating Sequential Processes)
- Robin Milner, 1980 – CCS (Calculus of Communicating Systems)
  - Stronger and more flexible than CSP
  - One major flaw!
- The answer: **Pi Calculus** (Milner et al., 1989)
  - "all that and a bag of chips"
  - Simple but expressive

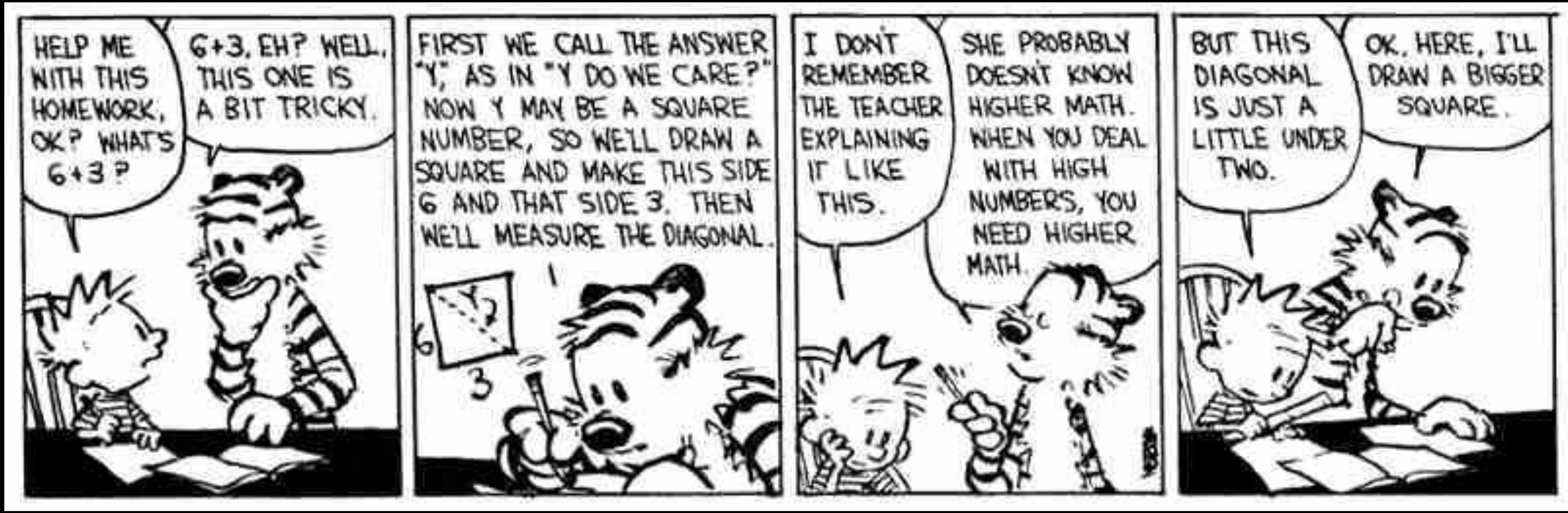
The Pi Calculus – Definition

- Multiple versions - presenting a simplified version for clarity
  - Even simpler than the first **Pi Calculus** paper
  - It will be enough to make it through simple protocols



The Pi Calculus – Definition

- The **Pi Calculus** is a very high level 'programming language'
  - Similar to the concept of the λ-calculus, but for concurrent programs
  - Everything in the **Pi Calculus** is a *process*
  - Processes communicate with each other using channels



The Pi Calculus – Definition

Everything in the **Pi Calculus** can be expressed in the grammar:

c<M>.P **output** message *M* on channel *c*, then do *P*

c(x).P **receive** message on channel *c* and binds it to *x*, then do *P*

P | Q **composition** – run *P* and *Q* in parallel

(μc)P **restriction** – create a new private name in *P* (called *c*)

There's more (replication, case matching, nil processes, etc.)
but we won't need them

The Pi Calculus – Definition

Just one more term we need:

- P ≈ Q means the *behaviors* of processes *P* and *Q* are indistinguishable
- They can have different internal structure
- A third process *R* cannot tell the difference between:
  - R | P
  - R | Q

The Pi Calculus – Example 1

Speaker = air<M>	sends M over the air
Phone = air(x).wire<x>	copies M from air to wire
AT&T = wire(x).fiber<x>	copies M from wire to fiber
System = Speaker | Phone | AT&T	the whole system

System -> fiber<M>

example adapted from W. Weimer

The Pi Calculus – Example 1

It’s easy to snoop on this system!

Introduce another process:

WireTap = wire(x).wire<x>.NSA<x>

The Pi Calculus – Example 1

Speaker = air<M>

Phone = air(x).wire<x>

WireTap = wire(x).wire<x>.NSA<x>

AT&T = wire(x).fiber<x>

System’ = Speaker | Phone | WireTap | AT&T

System’ -> fiber<M>

The Pi Calculus – Example 1

The problem:

System = Speaker | Phone | AT&T

System’ = Speaker | Phone | WireTap | AT&T

As defined, System ≈ System’.

We can’t tell if the NSA is tapping our phone calls in this system.

The Pi Calculus – Example 2

- Simplest (secure) system possible: *A* sends message *M* to *B* over c_{AB}

$A = c_{AB}$<M>

$B = c_{AB}(x)$

$System = (\mu c_{AB})(A \mid B)$

This time, the channel c_{AB} is restricted.

The Pi Calculus – Example 2

- Simplest (secure) system possible: *A* sends message *M* to *B* over c_{AB}

$A = c_{AB}\langle M \rangle$

$B = c_{AB}(x).F(x)$

$System = (\mu c_{AB})(A \mid B)$

This time, the channel c_{AB} is restricted.

The Pi Calculus – Example 2

Using this protocol, we can define two important cryptographic properties:

1. **Authenticity** or **integrity**: *B* always applies *F* to the message *M* that *A* sends. An attacker cannot cause *B* to apply *F* to some other message.
2. **Secrecy**: the message *M* cannot be read in transit from *A* to *B*. If *F* does not reveal *M*, the whole process does not reveal *M*.

The Pi Calculus – Example 2

- Secrecy: if $F(M) \approx F(M')$ for any M, M' then $System(M) \approx System(M')$.
- Authenticity: Create a new process, $System_{spec}$.

$$A = c_{AB}\langle M \rangle$$

$$B_{spec} = c_{AB}(x).F(M)$$

$$System_{spec} = (\mu c_{AB})(A \mid B_{spec})$$

The protocol has the authenticity property if $System_{spec} \approx System$

Finally… The Spi Calculus

What's wrong with this protocol as expressed in the Pi Calculus?

- The private channel c_{AB}(had to be created in System, but with no mention of:
  - how A and B can access it
  - why and attacker cannot

The notion of a completely private channel isn't realistic.

New idea: use public channels, but only send encrypted data

The Spi Calculus – Definition

Two new terms to define:

$\{M\}_N$ the **ciphertext** resulting from encrypting message M with key N

$[C, x]_N.P$ attempts to decrypt C with key N. If $C = \{M\}_N$ then process $P[M/x]$ runs. If decryption fails, it does nothing.

Note: these terms don't talk about *how* encryption/decryption occurs.

Thus, we are still dependent on security of any underlying algorithms.

The Spi Calculus – Example 1

Return to our example: A sends message M to B over c_{AB}.

This time, c_{AB} is public.

$A = c_{AB}\langle\{M\}_{K_{AB}}\rangle$

$B = c_{AB}(C) \,.\, [C, x]_{K_{AB}} \,.\, F(x)$

$System = (\mu K_{AB})(A \mid B)$

Secrecy and Authenticity can be defined the same as before.

The Spi Calculus – Example 1

- This protocol is more realistic than the **Pi Calculus** version
- Where old protocol had to establish a private channel, new protocol has to establish a private key
  - Both are ‘hand-waivy’
  - We can describe a protocol for each of these!

Wide-Mouthed Frog

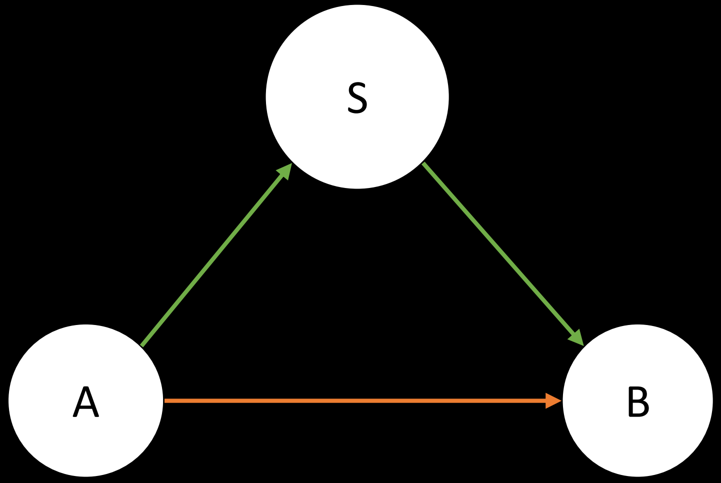
- In **Pi Calculus**, a protocol for channel establishment
- Uses a trusted third party

$A = (\mu c_{AB})c_{AS}\langle c_{AB}\rangle \,.\, c_{AB}\langle M\rangle$

$S = c_{AS}(x) \,.\, c_{BS}\langle x\rangle$

$B = c_{BS}(x) \,.\, x(y) \,.\, F(y)$

$\text{System} = (\mu c_{AS})(\mu c_{BS})(\ A \mid S \mid B\ )$



Wide-Mouthed Frog

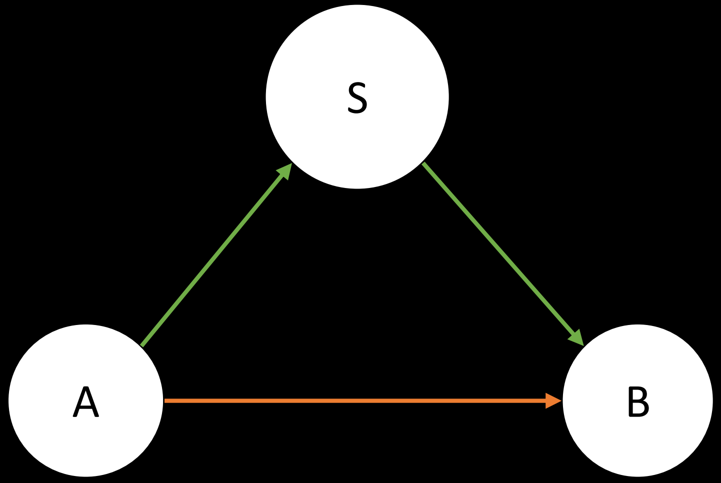
- In **Spi Calculus**, a protocol for key agreement
- Uses a trusted third party

$$A = (\mu k_{AB})c_{AS}<\{k_{AB}\}_{k_{AS}}> . c_{AB}<\{M\}_{k_{AB}}>$$

$$S = c_{AS}(x) . [x, y]_{k_{AS}} . c_{BS}<\{y\}_{k_{BS}}>$$

$$B = c_{BS}(x) . [x, y]_{k_{BS}} . c_{AB}(z_1) . [z_1, z_2]_y . F(z_2)$$

$$System = (\mu k_{AS})(\mu k_{BS})(A \mid S \mid B)$$



The Spi Calculus Today

- The **Spi Calculus** is used as a specification language for security protocols.
  - It is possible to automatically detect some security defects in a protocol
- There exists some work in converting **Spi Calculus** specifications to a Java implementation of the protocol
  - Also requires specification of underlying encryption
  - Comes with some correctness/security guarantees

Conclusions

- The **Pi Calculus** is a process calculus capable of describing a system of dynamic channel creation and manipulation.
  - It is possible to describe security protocols in the **Pi Calculus** alone.
- The **Spi Calculus** is an extension of the **Pi Calculus** with cryptographic primitives.
  - It enables us to model more realistic systems in which processes must communicate over public channels.

Questions?