HOW TO GIVE TALKS THAT PEOPLE CAN FOLLOW

Derek Dreyer
MPI for Software Systems

PLMW@POPL 2018
Los Angeles
My job as a researcher

Do research
My job as a researcher

- Do research
- Write papers
- Give talks
My job as a researcher

Do research

Write papers

Give talks
Entertain your audience!

Simon Peyton Jones. *How to give a great research talk.* (MSR Summer School, 2016)
- “Your mission is to **wake them up!**”
- “Your most potent weapon, by far, is **your enthusiasm!**”

John Hughes. *Unaccustomed as I am to public speaking.* (PLMW, 2016)
- “**Put on a show!**”
Entertain your audience!

• Simon Peyton Jones. *How to give a great research talk.* (MSR Summer School, 2016)

  “Your mission is to wake them up!”,
  “Your most potent weapon, by far, is your enthusiasm!”

Good advice, but I don’t know how to teach people to be entertaining...

• John Hughes. *Unaccustomed as I am to public speaking.* (PLMW, 2016)

  “Put on a show!”
A piece of research

Speaker

Audience
By downcasting the pre-axial gaskets, we achieved 47% reduction in XPS latency on the re-uptake bivalve!
By downcasting the pre-axial gaskets, we achieved 47% reduction in XPS latency on the re-uptake bivalve!

OK, but what does it do, and why do I care?
How is a conference talk different from a paper?
Conference talks

On the plus side:

✅ Lots of eyeballs on you and your work!

On the minus side:
Conference talks

On the plus side:

✅ Lots of eyeballs on you and your work!

On the minus side:

❌ You can’t say much.
❌ The audience may or may not care.
❌ Even those who care will easily get lost.
Conference talks

**On the plus side:**

✅ Lots of eyeballs on you and your work!

**On the minus side:**

❌ You can’t say much.

❌ The audience may or may not care.

❌ Even those who care will easily get lost.
A paper structure that works

• Abstract
• Intro
• Key ideas
• Technical meat
• Related work
A paper structure that works

- Abstract
- Intro
- Key ideas
- Technical meat
- Related work
A paper structure that works

- Intro
- Key ideas
- Technical meat
- Related work
A paper structure that works

- Abstract
- Intro
- Key ideas
- Technical meat
- Related work
A paper structure that works

- **Intro** (8 minutes)
- **Key ideas** (11 minutes)
A paper structure that works

- Intro (8 minutes)
- Key ideas (11 minutes)
- What else is in the paper (1 minute)
Key ideas

- Use **concrete illustrative examples** and high-level intuition.

- Do **not** show the general solution! (People can go read your paper for that.)
Conferecne talks

On the plus side:

✅ Lots of eyeballs on you and your work!

On the minus side:

❌ You can’t say much.

❌ The audience may or may not care.

❌ Even those who care will easily get lost.
Conference talks

On the plus side:
✓ Lots of eyeballs on you and your work!

On the minus side:
✗ You can’t say much.
✗ The audience may or may not care.
✗ Even those who care will easily get lost.
BRO

WHAT'S YOUR PROBLEM?!
Stage the motivation

● First, get to a problem.
  - Explain a general version of your problem (but not too general) in the first 2 minutes.

● Then, get to the problem.
  - Motivate and explicitly state your specific problem in the next 4 minutes.
  - Limit discussion of prior work only to what is needed to explain your problem.
WHAT DID YOU DO?!?
Tell them what you did!

- **Proudly state your contributions.**
  
  - After the motivation, the audience eagerly wants to hear what you did. Tell them!

- **Follow immediately with a crisp statement of your key idea(s).**
  
  - It will give audience a take-home message, and give focus to the rest of your talk.
Conference talks

On the plus side:

✓ Lots of eyeballs on you and your work!

On the minus side:

✗ You can’t say much.
✗ The audience may or may not care.
✗ Even those who care will easily get lost.
Conference talks

On the plus side:

✔️ Lots of eyeballs on you and your work!

On the minus side:

❌ You can’t say much.

❌ The audience may or may not care.

❌ Even those who care will easily get lost.
Flow & coherence

Create flow with old to new

Create coherence with one slide, one point
Does this text flow?
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach.

What does this game-playing technique have to do with what came before?
Old to new

- Begin sentences with old info
  - Creates link to earlier text

- End sentences with new info
  - Creates link to the text that follows
  - Also places new info in position of emphasis
Applying old-to-new

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. The game-playing technique, originally proposed by Jones et al., follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games. This is a general design principle for cryptographic proofs to ease their management.
Applying old-to-new

Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. To make it easier to manage such proofs, Jones et al. have proposed a new design principle, called the game-playing technique. This technique follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games.
Security proofs of cryptographic protocols are crucial for the security of everyday electronic communication. However, these proofs tend to be complex and difficult to get right. To make it easier to manage such proofs, Jones et al. have proposed a new design principle, called the game-playing technique. This technique follows a code-based approach where the security properties are formulated in terms of probabilistic programs, called games.
Flow in talks

- **Within** a slide:
  - Script should follow “old to new”

- **Between** slides:
  - Don’t just flip to next slide and say, “So…”
  - Plan something to say **during** the transition
Flow & coherence

Create **flow** with **old to new**

Create **coherence** with **one slide, one point**
Optimization & Concurrency

- Compiler performs several optimizations to generate optimized code.
  - >100 optimizations in GCC, LLVM.

Correct optimizations for sequential programs may be incorrect for shared memory concurrency.

State-of-the-Art:
- Compilers are over-conservative;
  * optimization opportunities are lost.

or
- Buggy optimization
  * “Premature optimization is the root of all evil” ~ Donald Knuth
Talklets

- Break long stretches of talk into talklets.
  - More digestible units of story (2-4 min.)
  - But just having talklets is not enough…

- Use transitions between talklets to remind the audience of the big picture.
  - Summarize the point of the last talklet and how it connects to the next one.
A few words about Slide Design
No sense of style?

Don’t worry
The most important aspects of slide design have **nothing** to do with style
Access control is inadequate, scenario 2: Facebook timeline
- Facebook introduced timeline in 2011 end
- Chronologically order the information on your profile
- Make them easily searchable for other users
- Easier to search, potentially embarrassing older content
- Users were afraid of privacy violation
- Access control was not changed!

Access control is inadequate, scenario 3: Spokes
- Service aggregating information about individuals
- Each individual information is public content
- E.g., your Facebook profile, address
- One can infer non-public information
- Estimating wealth using address and public property records
- Users complain of privacy violation
- Access control was not changed!

Access control inadequate: Summary
- User reaction suggests each of the cases violate privacy
- However, in none of the cases access control is violated
- We propose a new model to reason about privacy

Exposure: Definition
- We define Privacy Exposure at time t as $P(t)$
- $P(t) = U|U$ is aware of $I$ at time $t$
- Then, $E$, exposure of $I$ is:
  
  \[ E = \lim_{t \to \infty} P(t) \]

Modeling user privacy using exposure
- For each content, users have an expected exposure
  - How many other users are likely to access the content
- We can model privacy violation for an information as:
  - Large deviation of actual exposure from expected exposure

Rewriting scenario 1: Facebook newfeed
- Before newfeed was introduced:
  - Expected exposure: Friends who will visit user’s profile
  - Actual exposure was same as expected exposure
- After newfeed was introduced:
  - Actual exposure: All friends to whom the information is pushed
  - Actual exposure is much higher than the expected exposure

Rewriting scenario 2: Facebook timeline
- Before timeline was introduced:
  - Expected exposure for older data: Friends who will search for old content
  - Actual exposure for older data was same as expected exposure
- After timeline was introduced:
  - Actual exposure for old data: All friends who visit the profile
  - Actual exposure is much higher than the expected exposure

Rewriting scenario 3: Spokes
- Before spokes aggregated data:
  - Expected exposure for new inferred data: Users who dig up each individual piece of content from different sources
  - Actual exposure for old data was same as expected exposure
- After spokes aggregated data:
  - Actual exposure for new inferred data: All users who visit public spoke website
  - Actual exposure is much higher than the expected exposure

Proposed model: managing privacy via exposure
- Major deviations from expected exposure can capture the privacy violations not covered by access control

Key challenge: Predicting future exposure
- Huge existing work for predicting growth in content popularity
- E.g., YouTube, Facebook likes, networks
- Use machine learning, regression techniques
- We can leverage advances in those fields to predict exposure
- OSN operators are best-positioned to do the predictions
  - Empirical data on how information disseminates in their sites
  - Facebook or YouTube already provide number of likes or views

Limitations of our model
- Privacy violation by inference using available data
  - It is extremely hard to enumerate all possible inference
- Privacy violation using cross site prediction
  - Prediction across multiple systems
  - E.g., posting a picture taken from Facebook in Twitter
The Protocol

Stack transfer

Our implementation (in C#)

Key idea: Messages are resources
Store in lock-free bags
- parallelized matching
- decreased communication

Is this just STM?

The Problem:
Concurrency libraries are indispensable, but hard to build and extend

This work

Use join patterns for synchronization: [Fournet & Gongther]

Expressive
Write synchronization primitives declaratively and concisely

Scalable
Competitive with industrial libraries; can recover existing algorithms

java.util.concurrent

Synchronization
Reentrant locks
Semaphores
R/W locks
Reentrant R/W locks
Condition variables
Countdown latches
Cyclic barriers
Phasers
Exchangers

Data structures
Queues
Nonblocking
Blocking (array & list)
Synchronous
Priority, nonblocking
Priority, blocking
Deques
Sets
Maps (hash & skiplists)

State of the art?
Leave it to the experts:

Research Literature  Industrial-strength Libraries
java.util.concurrent .NET 4.0 Intel TBB
Key takeaways

- **Avoid PowerPoint-itis**
  - Don’t put lots of text on slides just so they are readable independently of the talk

- **Vary the look of the slides**
  - Some text-only slides are fine, but if there are too many in a row, audience falls asleep
That's all Folks!