

Access Control and Privacy Policies (7)

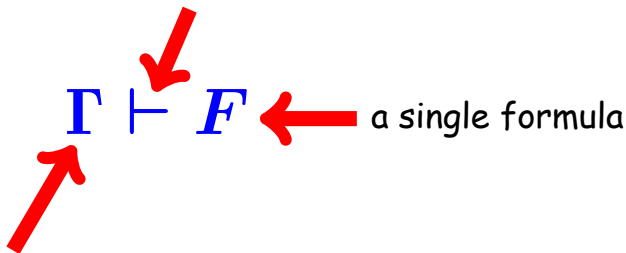
Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)
Slides: KEATS (also homework is there)

Judgements

$$\Gamma \vdash F$$

Judgements

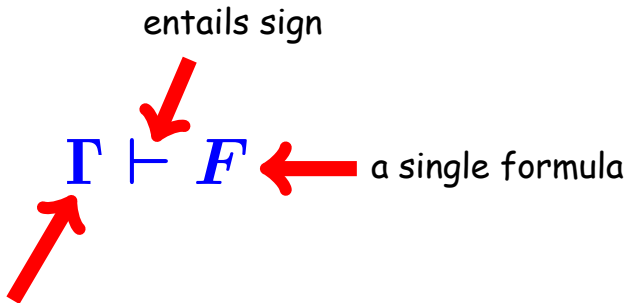
entails sign



Gamma

stands for a collection of formulas
("assumptions")

Judgements



Gamma
stands for a collection of formulas
("assumptions")

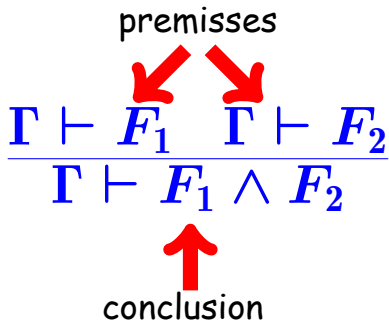
Gimel (Phoenician), Gamma (Greek), C and G (Latin), Gim (Arabic),
?? (Indian), Ge (Cyrillic)

Inference Rules

premisses

$$\frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}$$

conclusion

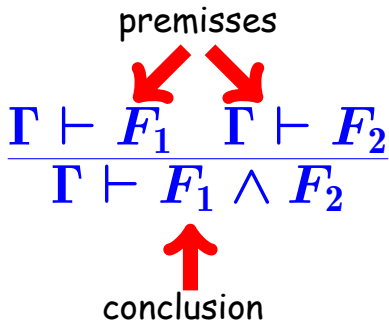


Inference Rules

premisses

$$\frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}$$

conclusion



P says $F \vdash$ *Q* says $F \wedge$ *P* says G

Inference Rules

premisses

$$\frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}$$

conclusion

$$\underbrace{P \text{ says } F}_{\Gamma} \vdash \underbrace{Q \text{ says } F}_{F_1} \wedge \underbrace{P \text{ says } G}_{F_2}$$

$$\frac{\Gamma \vdash F_1 \Rightarrow F_2 \quad \Gamma \vdash F_1}{\Gamma \vdash F_2}$$

$$\frac{\Gamma \vdash F}{\Gamma \vdash P \text{ says } F}$$

We want to prove

$\Gamma \vdash \text{del_file}$

We want to prove

$$\Gamma \vdash \text{del_file}$$

There is an inference rule

$$\frac{\Gamma \vdash F}{\Gamma \vdash P \text{ says } F}$$

We want to prove

$$\Gamma \vdash \text{del_file}$$

There is an inference rule

$$\frac{\Gamma \vdash F}{\Gamma \vdash P \text{ says } F}$$

So we can derive $\Gamma \vdash \text{Alice says del_file}$.

We want to prove

$$\Gamma \vdash \text{del_file}$$

There is an inference rule

$$\frac{\Gamma \vdash F}{\Gamma \vdash P \text{ says } F}$$

So we can derive $\Gamma \vdash \text{Alice says del_file}$.

Γ contains already $\text{Alice says del_file}$.

So we can use the rule

$$\overline{\Gamma, F \vdash F}$$

Done. Qed.

We want to prove

$$\Gamma \vdash \text{del_file}$$

There is an inference rule

$$\frac{\Gamma \vdash F}{\Gamma \vdash P \text{ says } F}$$

So we can derive $\Gamma \vdash \text{Alice says del_file}$.

Γ contains already $\text{Alice says del_file}$.

So we can use the rule

$$\overline{\Gamma, F \vdash F}$$

What is wrong with this?

Done. Qed.

Digression: Proofs in CS

Formal proofs in CS sound like science fiction?
Completely irrelevant! Lecturers gone mad!

Digression: Proofs in CS

Formal proofs in CS sound like science fiction?
Completely irrelevant! Lecturers gone mad!

- in 2008, verification of a small C-compiler
 - "if my input program has a certain behaviour, then the compiled machine code has the same behaviour"
 - is as good as `gcc -O1`, but less buggy
- in 2010, verification of a micro-kernel operating system (approximately 8700 loc)
 - 200k loc of proof
 - 25 - 30 person years
 - found 160 bugs in the C code (144 by the proof)



Bob Harper
(CMU)



Frank Pfenning
(CMU)

published a proof about a
specification in a journal
(2005), ~31pages



Bob Harper
(CMU)



Frank Pfenning
(CMU)

published a proof about a
specification in a journal
(2005), ~31pages

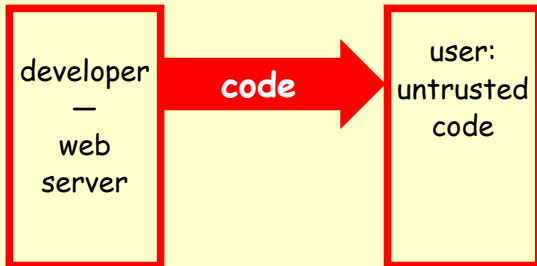


Andrew Appel
(Princeton)

relied on their proof in a
security critical application

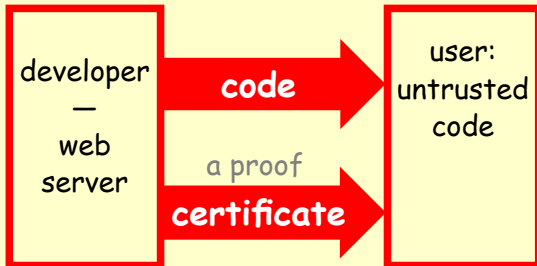
Proof-Carrying Code

Idea:



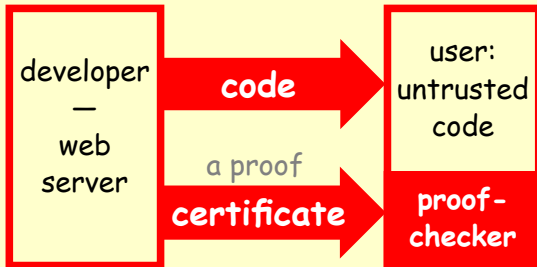
Proof-Carrying Code

Idea:



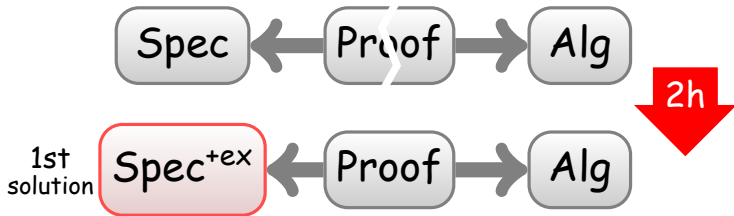
Proof-Carrying Code

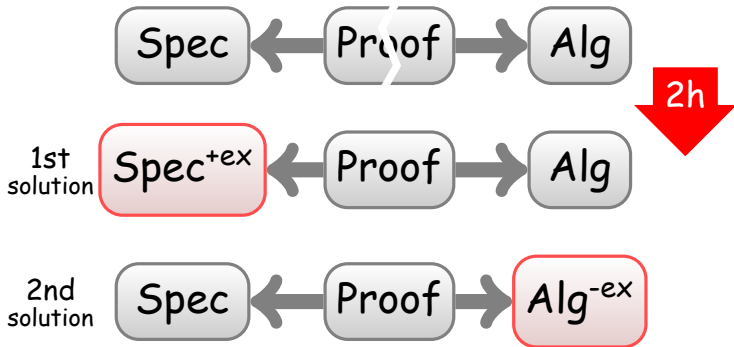
Idea:

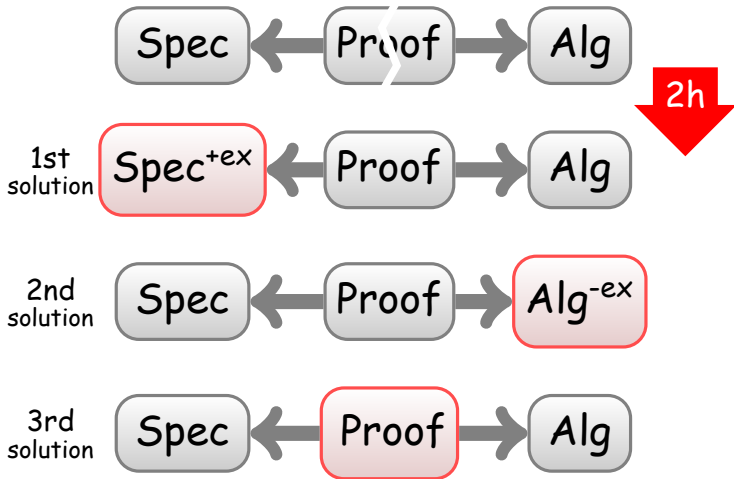




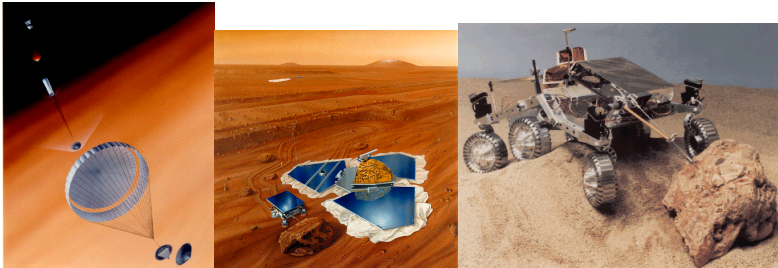








Mars Pathfinder Mission 1997



- despite NASA's famous testing procedure, the lander crashed frequently on Mars
- problem was an algorithm not used in the OS

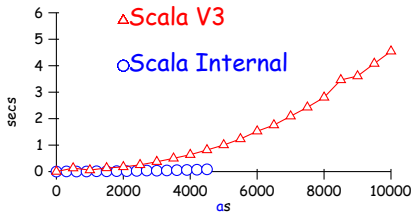
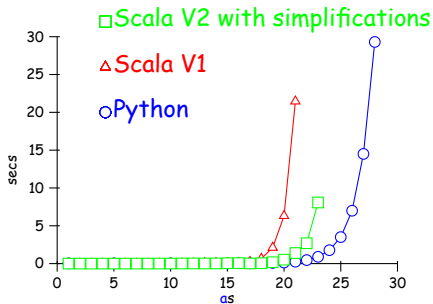
Priority Inheritance Protocol

- ...a scheduling algorithm that is widely used in real-time operating systems
- has been “proved” correct by hand in a paper in 1983
- ...but the first algorithm turned out to be incorrect, despite its “proof”

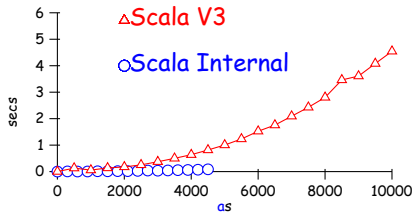
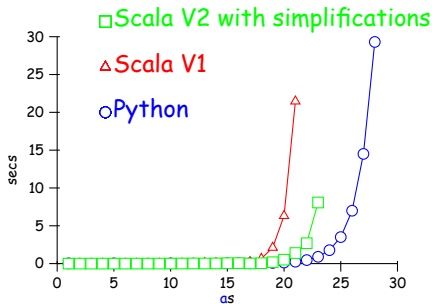
Priority Inheritance Protocol

- ...a scheduling algorithm that is widely used in real-time operating systems
- has been “proved” correct by hand in a paper in 1983
- ...but the first algorithm turned out to be incorrect, despite its “proof”
- we specified the algorithm and then proved that the specification makes “sense”
- we implemented our specification in C on top of PINTOS (used for teaching at Stanford)
- our implementation was much more efficient than their reference implementation

Regular Expression Matching

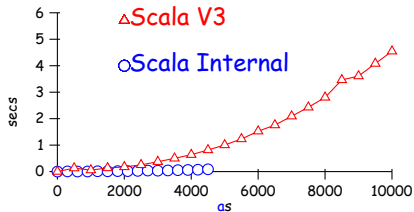
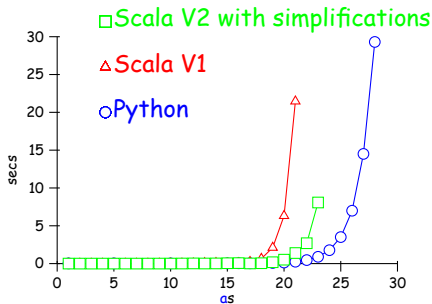


Regular Expression Matching



- I needed a proof in order to make sure my program is correct

Regular Expression Matching



- I needed a proof in order to make sure my program is correct

End Digression.

(Our small proof is 0.0005% of the OS-proof.)

One More Thing

- I arrived at King's last year
- Maxime Crochemore told me about a string algorithm (suffix sorting) that appeared at a conference in 2007 (ICALP)
- "horribly incomprehensible", no implementation, but claims to be the best $O(n + k)$ algorithm

One More Thing

- I arrived at King's last year
- Maxime Crochemore told me about a string algorithm (suffix sorting) that appeared at a conference in 2007 (ICALP)
- "horribly incomprehensible", no implementation, but claims to be the best $O(n + k)$ algorithm
- Jian Jiang found 1 error and 1 superfluous step in this algorithm
- he received 88% for the project and won the prize for the best 7CCSMPRJ project in the department
- no proof ... yet

Trusted Third Party

Simple protocol for establishing a secure connection via a mutually trusted 3rd party (server):

Message 1 $A \rightarrow S : A, B$

Message 2 $S \rightarrow A : \{K_{AB}\}_{K_{AS}}$ and $\{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}}$

Message 3 $A \rightarrow B : \{K_{AB}\}_{K_{BS}}$

Message 4 $A \rightarrow B : \{m\}_{K_{AB}}$

Encrypted Messages

- Alice sends a message m

Alice says m

Encrypted Messages

- Alice sends a message m

Alice says m

- Alice sends an encrypted message m
(with key K)

Alice says $\{m\}_K$

Encrypted Messages

- Alice sends a message m

Alice says m

- Alice sends an encrypted message m
(with key K)

Alice says $\{m\}_K$

- Decryption of Alice's message

$$\frac{\Gamma \vdash \text{Alice says } \{m\}_K \quad \Gamma \vdash \text{Alice says } K}{\Gamma \vdash \text{Alice says } m}$$

Encryption

- Encryption of a message

$$\frac{\Gamma \vdash \text{Alice says } m \quad \Gamma \vdash \text{Alice says } K}{\Gamma \vdash \text{Alice says } \{m\}_K}$$

Trusted Third Party

- Alice calls Sam for a key to communicate with Bob
- Sam responds with a key that Alice can read and a key Bob can read (pre-shared)
- Alice sends the message encrypted with the key and the second key it received

A sends *S* : $\text{Connect}(A, B)$

S sends *A* : $\{K_{AB}\}_{K_{AS}}$ and $\{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}}$

A sends *B* : $\{K_{AB}\}_{K_{BS}}$

A sends *B* : $\{m\}_{K_{AB}}$

Sending Rule

$$\frac{\Gamma \vdash P \text{ says } F \quad \Gamma \vdash P \text{ sends } Q : F}{\Gamma \vdash Q \text{ says } F}$$

Sending Rule

$$\frac{\Gamma \vdash P \text{ says } F \quad \Gamma \vdash P \text{ sends } Q : F}{\Gamma \vdash Q \text{ says } F}$$

$$P \text{ sends } Q : F \stackrel{\text{def}}{=} (P \text{ says } F) \Rightarrow (Q \text{ says } F)$$

Trusted Third Party

A sends *S* : $\text{Connect}(A, B)$

S says $(\text{Connect}(A, B) \Rightarrow$

$$\{K_{AB}\}_{K_{AS}} \wedge \{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}})$$

S sends *A* : $\{K_{AB}\}_{K_{AS}} \wedge \{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}}$

A sends *B* : $\{K_{AB}\}_{K_{BS}}$

A sends *B* : $\{m\}_{K_{AB}}$

Trusted Third Party

A sends S : $\text{Connect}(A, B)$

S says $(\text{Connect}(A, B) \Rightarrow$

$$\{K_{AB}\}_{K_{AS}} \wedge \{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}})$$

S sends A : $\{K_{AB}\}_{K_{AS}} \wedge \{\{K_{AB}\}_{K_{BS}}\}_{K_{AS}}$

A sends B : $\{K_{AB}\}_{K_{BS}}$

A sends B : $\{m\}_{K_{AB}}$

$\Gamma \vdash B$ says m ?

Challenge-Response Protocol

- an engine E and a transponder T share a key K
- E sends out a **nonce** N (random number) to T
- T responds with $\{N\}_K$
- if E receives $\{N\}_K$ from T , it starts engine

Challenge-Response Protocol

E says N (start)

E sends $T : N$ (challenge)

$(T \text{ says } N) \Rightarrow (T \text{ sends } E : \{N\}_K \wedge$
 $T \text{ sends } E : \text{Id}(T))$ (response)

T says K (key)

T says $\text{Id}(T)$ (identity)

$(E \text{ says } \{N\}_K \wedge E \text{ says } \text{Id}(T)) \Rightarrow$
 $\text{start_engine}(T)$ (engine)

$\Gamma \vdash \text{start_engine}(T)?$

Exchange of a Fresh Key

A and B share a ("super-secret") key K_{AB} and want to share another key

- assumption K_{AB} is only known to A and B
- A sends B : $A, \{N_A\}_{K_{AB}}$
- B sends A : $\{N_A + 1, N_B\}_{K_{AB}}$
- A sends B : $\{N_B + 1\}_{K_{AB}}$
- B sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$

Assume K_{AB}^{new} is compromised by I

Exchange of a Fresh Key

A and B share a ("super-secret") key K_{AB} and want to share another key

- assumption K_{AB} is only known to A and B
- A sends B : $A, \{N_A\}_{K_{AB}}$
- B sends A : $\{N_A + 1, N_B\}_{K_{AB}}$
- A sends B : $\{N_B + 1\}_{K_{AB}}$
- B sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$
- A sends B : $\{msg\}_{K_{AB}^{new}}$

Assume K_{AB}^{new} is compromised by I

The Attack

An intruder I convinces A to accept the compromised key K_{AB}^{new}

- A sends B : $A, \{N_A\}_{K_{AB}}$
- B sends A : $\{N_A + 1, N_B\}_{K_{AB}}$
- A sends B : $\{N_B + 1\}_{K_{AB}}$
- B sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$ recorded by I

The Attack

An intruder I convinces A to accept the compromised key K_{AB}^{new}

- A sends B : $A, \{N_A\}_{K_{AB}}$
- B sends A : $\{N_A + 1, N_B\}_{K_{AB}}$
- A sends B : $\{N_B + 1\}_{K_{AB}}$
- B sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$ recorded by I
- A sends B : $A, \{M_A\}_{K_{AB}}$
- B sends A : $\{M_A + 1, M_B\}_{K_{AB}}$
- A sends B : $\{M_B + 1\}_{K_{AB}}$
- B sends I : $\{K_{AB}^{newer}, N_B^{newer}\}_{K_{AB}}$ intercepted by I
- I sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$

The Attack

An intruder I convinces A to accept the compromised key K_{AB}^{new}

- A sends B : $A, \{N_A\}_{K_{AB}}$
- B sends A : $\{N_A + 1, N_B\}_{K_{AB}}$
- A sends B : $\{N_B + 1\}_{K_{AB}}$
- B sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$ recorded by I
- A sends B : $A, \{M_A\}_{K_{AB}}$
- B sends A : $\{M_A + 1, M_B\}_{K_{AB}}$
- A sends B : $\{M_B + 1\}_{K_{AB}}$
- B sends I : $\{K_{AB}^{newer}, N_B^{newer}\}_{K_{AB}}$ intercepted by I
- I sends A : $\{K_{AB}^{new}, N_B^{new}\}_{K_{AB}}$
- A sends B : $\{msg\}_{K_{AB}^{new}}$ I can read it also

Another Challenge-Response

A and B share the key K_{AB} and want to identify each other

- A sends B : A, N_A
- B sends A : $\{N_A, K'_{AB}\}_{K_{AB}}$
- A sends B : $\{N_A\}_{K'_{AB}}$

Another Challenge-Response

A and B share the key K_{AB} and want to identify each other

- A sends B : A, N_A
- B sends A : $\{N_A, K'_{AB}\}_{K_{AB}}$
- A sends B : $\{N_A\}_{K'_{AB}}$

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

I sends A : B, N_A

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

I sends A : B, N_A

A sends I : $\{N_A, K'_{AB}\}_{K_{AB}}$

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

I sends A : B, N_A

I sends A : $\{N_A, K'_{AB}\}_{K_{AB}}$

A sends I : $\{N_A, K'_{AB}\}_{K_{AB}}$

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

I sends A : B, N_A

I sends A : $\{N_A, K'_{AB}\}_{K_{AB}}$

A sends I : $\{N_A, K'_{AB}\}_{K_{AB}}$

A sends I : $\{N_A\}_{K'_{AB}}$

Another Challenge-Response

Unfortunately, an intruder I can impersonate B .

A sends I : A, N_A

I sends A : $\{N_A, K'_{AB}\}_{K_{AB}}$

A sends I : $\{N_A\}_{K'_{AB}}$

I sends A : B, N_A

A sends I : $\{N_A, K'_{AB}\}_{K_{AB}}$

I sends A : $\{N_A\}_{K'_{AB}}$