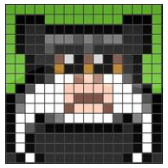
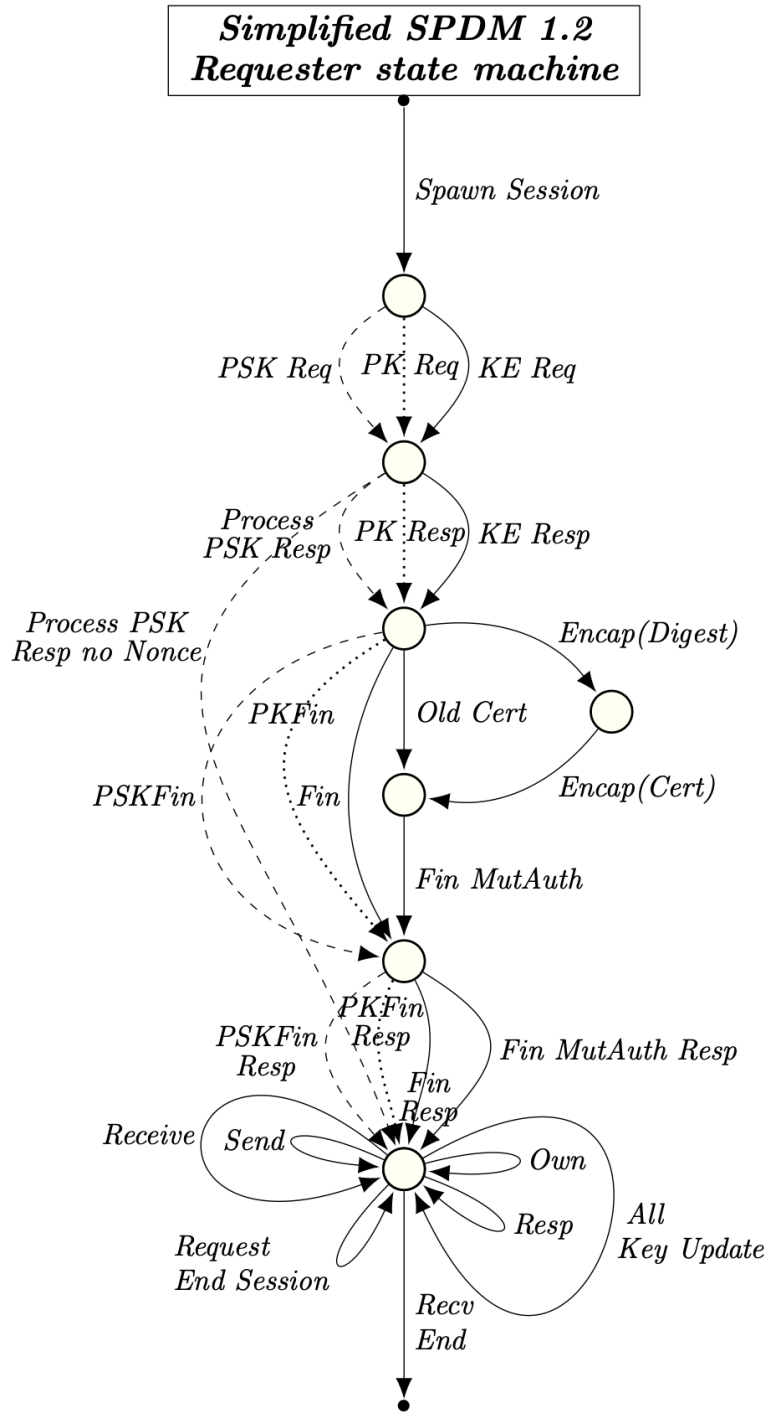
CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY



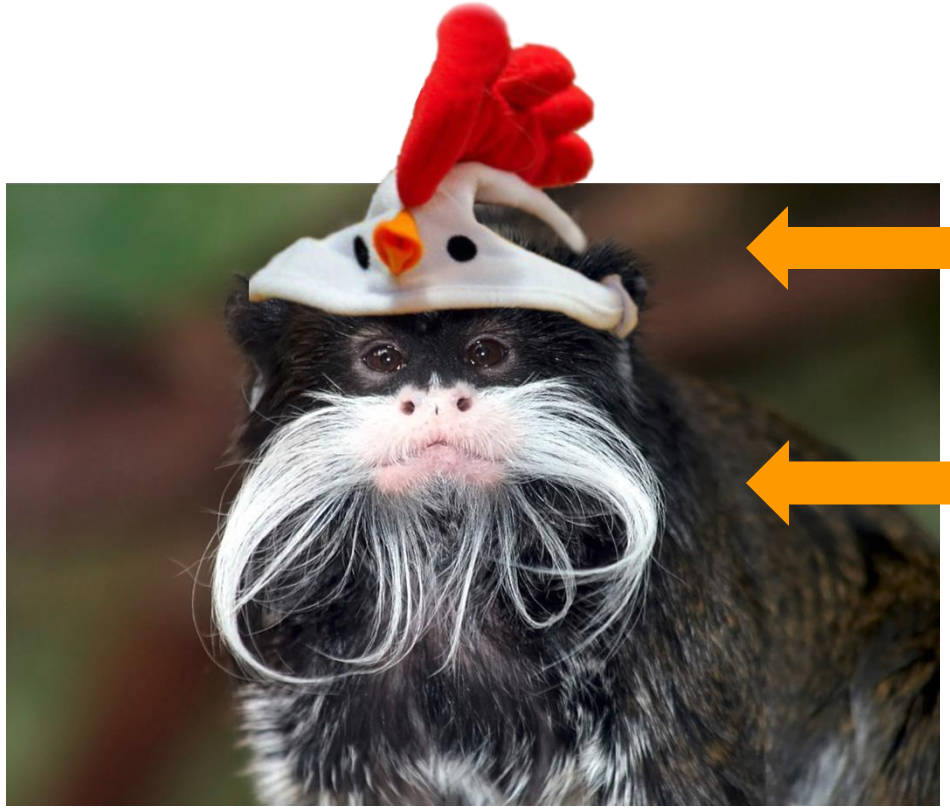
Protocol analysis using Tamarin

Cas Cremers

May 4, CAPS workshop, Eurocrypt 2025



Tamarin prover



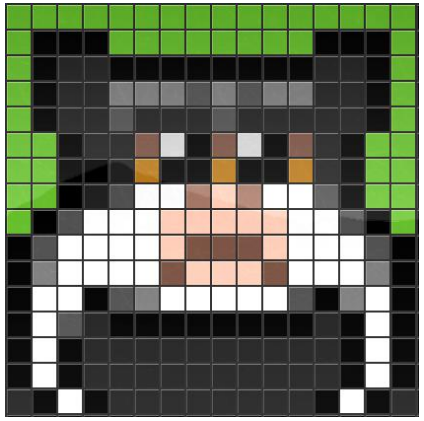
Theorem Prover

Constraint solver

Tamarin prover

Emperor Tamarin

Tamarin prover

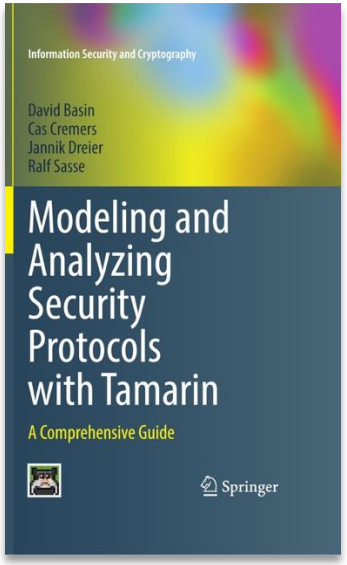


tamarin-prover.com

Development started around 2010 at ETH

Current development core:
- CISPA (Cas Cremers)
- ETH Zurich (David Basin, Ralf Sasse)
- INRIA (Jannik Dreier)

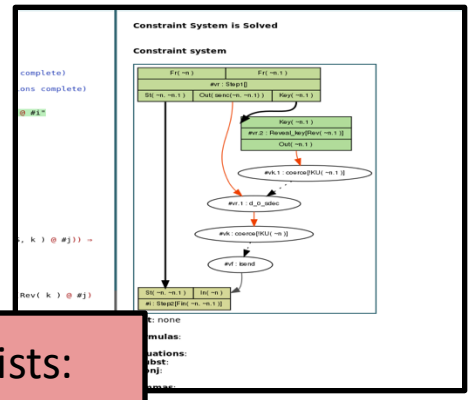
Open-source development, with
- Manual
- Online tutorials
- Active mailing list
- Syntax highlighting (vim, VSC, …)
- Upcoming **book**! *Free for download*
  “Modeling and Analyzing Security Protocols with Tamarin: A Comprehensive Guide”



Attacks Tamarin can find or prove absent

- Large attack scenarios (TLS 1.3 Rev 10: 18 messages)
- Cross-protocol attacks
- Unintended state machine transitions
- Downgrade attacks
- Nonce-reuse attacks (eg WiFi with AES-GCM)
- Invalid Curve Points
- Small order points
- DSKS attacks (Duplicate Signature Key Selection)
- Length extension attacks
- Maliciously generated keys
- Hybrid schemes
- ....

Tamarin prover: workflow



Input file

System S

Property P

Tamarin prover

constraints from S

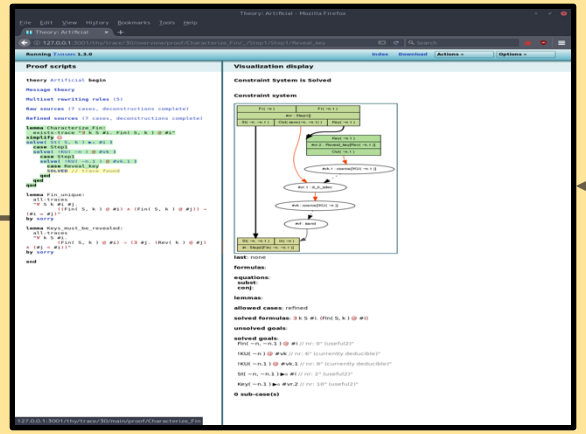
constraints from (not P)

Dedicated constraint solver

Solution exists: ATTACK

No solution exists: PROOF

Run out of time or memory



Interactive mode
(Inspect or guide partial proof)

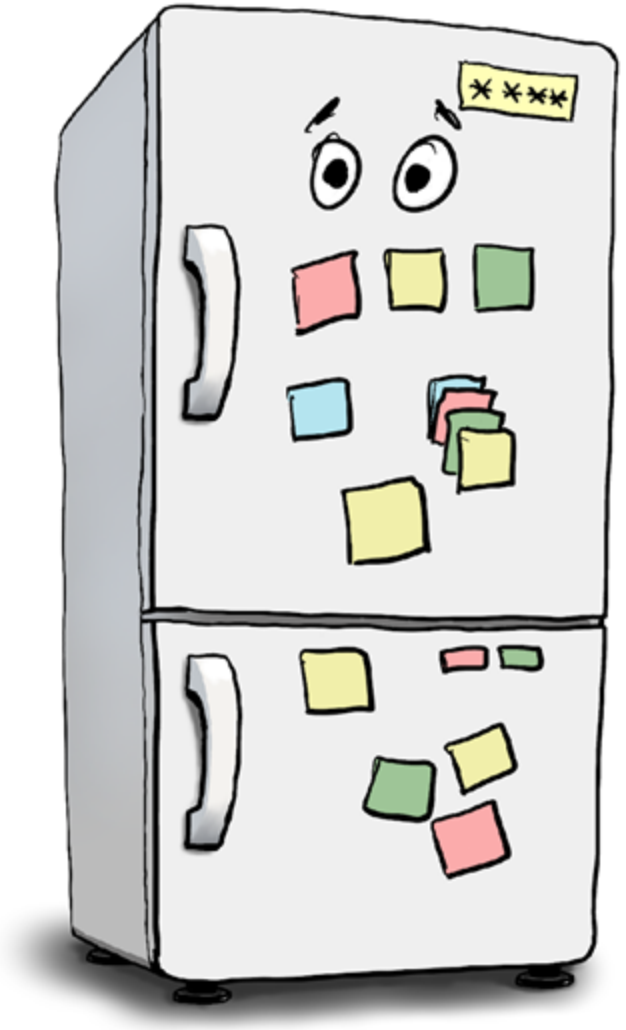
Provide **hints** for the prover (e.g. invariants)

Modeling in Tamarin: transition systems

- Basic ingredients:
  - **Terms** (think “messages”)
  - **Facts** (think “sticky notes on the fridge”)
  - Special facts: **Fr(t)**, **In(t)**, **Out(t)**, **K(t)**
- State of system is a multiset of facts
  - **Initial state** is the empty multiset
  - **Rules** specify the transition rules (“moves”)
- Rules are of the form:

  l --> r

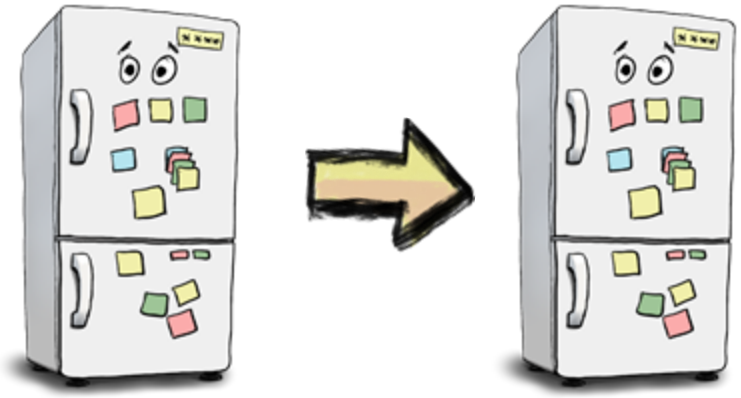
  l --[a]-> r



The model

- **Term algebra**
  - enc(_,_), dec(_,_), h(_,_), _^_, _$^{-1}$, _*_, 1, …
- **Equational theory**
  - dec(enc(m,k),k) $=_E$ m,
  - (x^y)^z $=_E$ x^(y*z),
  - $(x^{-1})^{-1} =_E$ x, …
- **Facts**
  - F(t1,…,tn)
- **Transition system**
  - State: multiset of facts
  - Rules: l –[a]→ r
- **Tamarin-specific**
  - Built-in Dolev-Yao attacker rules:
    In( ),
    Out( ),
    K( )
  - Special **Fresh** rule:
    - [] --[]--> [Fr(**x**)]
      - Constraint on system such that **x** is unique

Semantics



- **Transition relation**

  $S -[a]\rightarrow_R ((S \setminus^{\#} l) \cup^{\#} r)$, where

  - $\notin$ $l -[a]\rightarrow r$ is a ground instance of a rule in R, and
  - $l \subseteq^{\#} S$ w.r.t. the equational theory

- **Executions**

  $-\text{Exec}(R) = \{ [\,] -[a_1]\rightarrow \ldots -[a_n]\rightarrow S_n$
  $\mid \forall n \,.\, \text{Fr}(n)$ appears only once on right-hand side of rule $\}$

- **Traces**

  $-\text{Traces}(R) = \{ [a_1,\ldots,a_n] \mid [\,] -[a_1]\rightarrow \ldots -[a_n]\rightarrow S_n \in \text{Exec}(R) \}$

  –Property specification using first-order logic over traces

Semantics: example 1

‘c’ constant

- **Rules**
  - rule 1: [ ] –[Init()]→ [A('5')]
  - rule 2: [A(x)] –[Step(x)]→ [B(x)]
- **Execution example**
  - [ ]
  - –[Init()]→ [A('5')]
  - –[Init()]→ [A('5'), A('5')]
  - –[Step('5')]→ [A('5'), B('5')]
- **Corresponding trace:** [Init(), Init(), Step('5')]

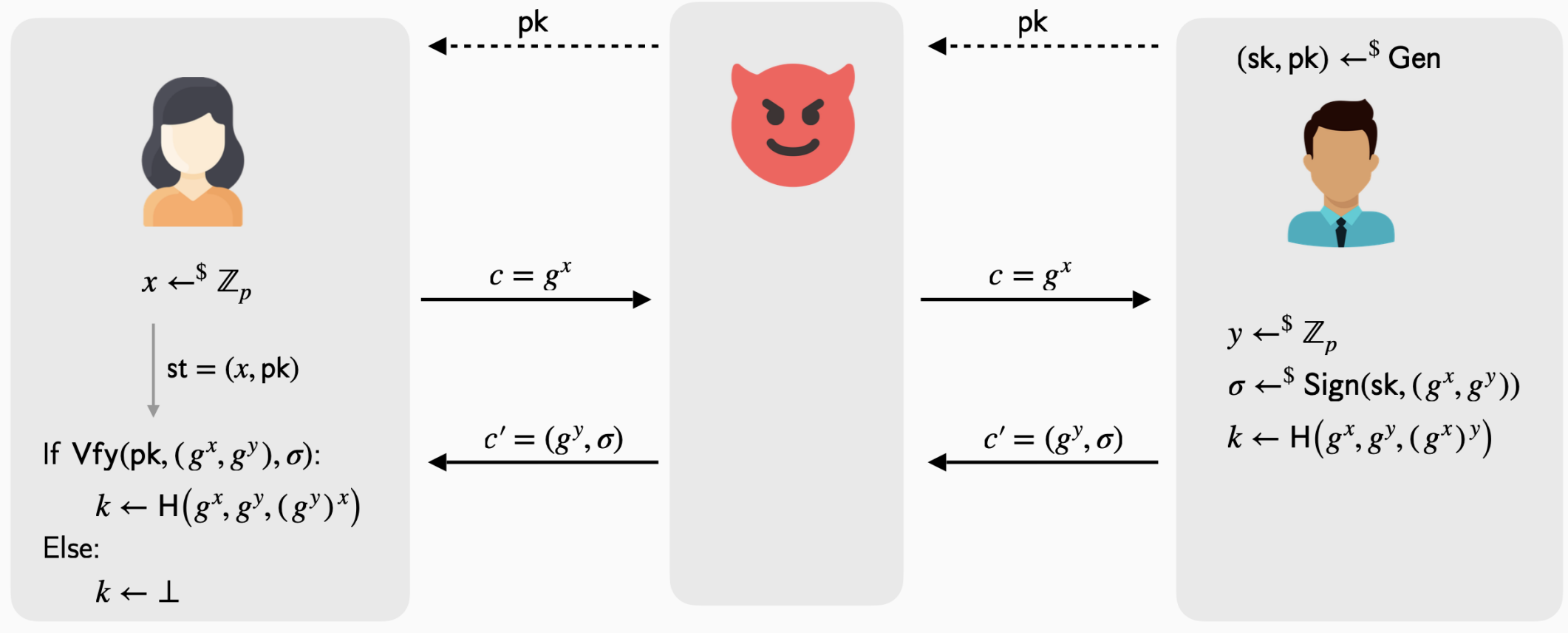
Semantics: example 2 (persistent facts)

'c'	constant
!F(...)	persistent fact

- **Rules**
  - rule 1: [ ] –[Init()]→ [!C('ok'), D('1')]
  - rule 2: [!C(x), D(y)] –[Step(x,y)]→ [D(h(y))]
- **Execution example**
  - [ ]
  - –[Init()]→ [!C('ok'), D('1
  - –[Step('ok','1')]→ [!C('ok'), D(h
  - –[Step('ok',h('1'))]→ [!C('ok'), D(h(h('1'
- **Corresponding trace:** [Init(), Step('ok', '1'), Step('ok', h('1'))]

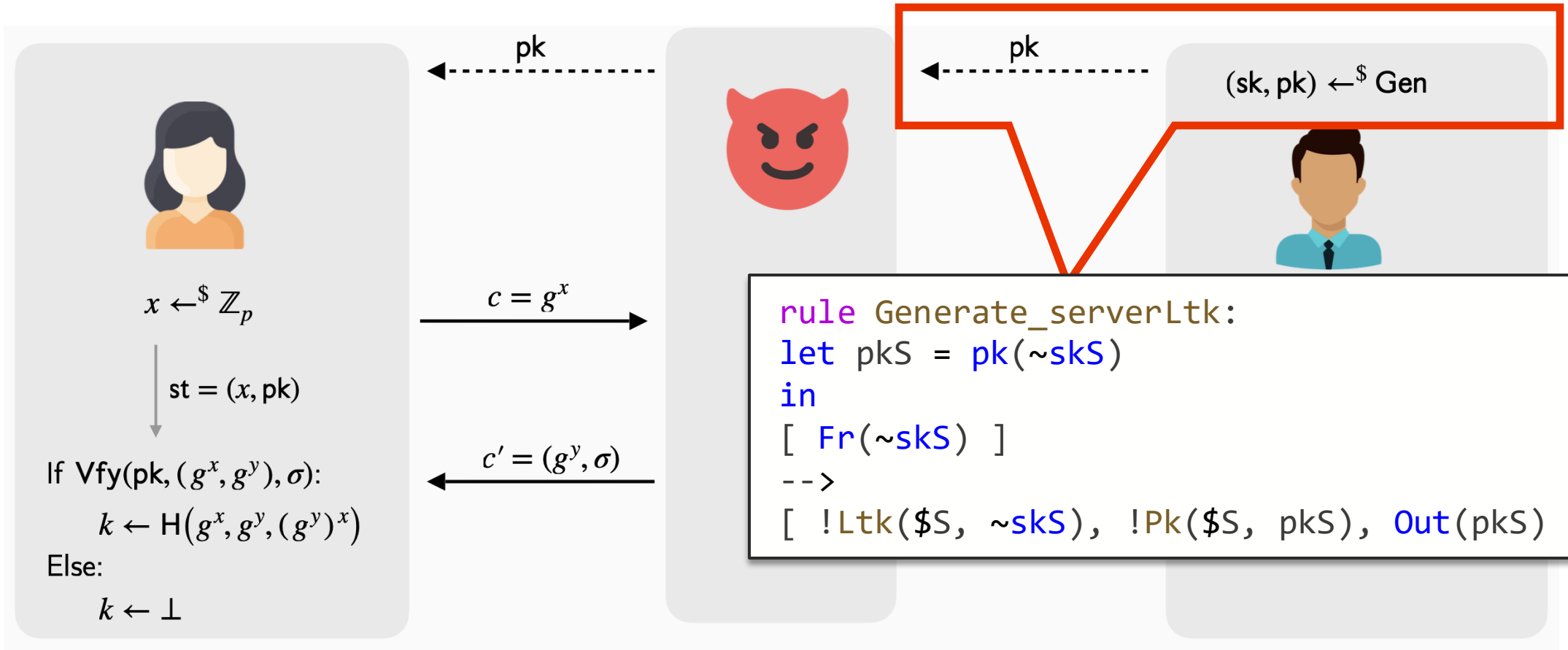
Security properties expressed as first-order logic formulas over traces with quantification over timepoints

Unilateral signed Diffie-Hellman



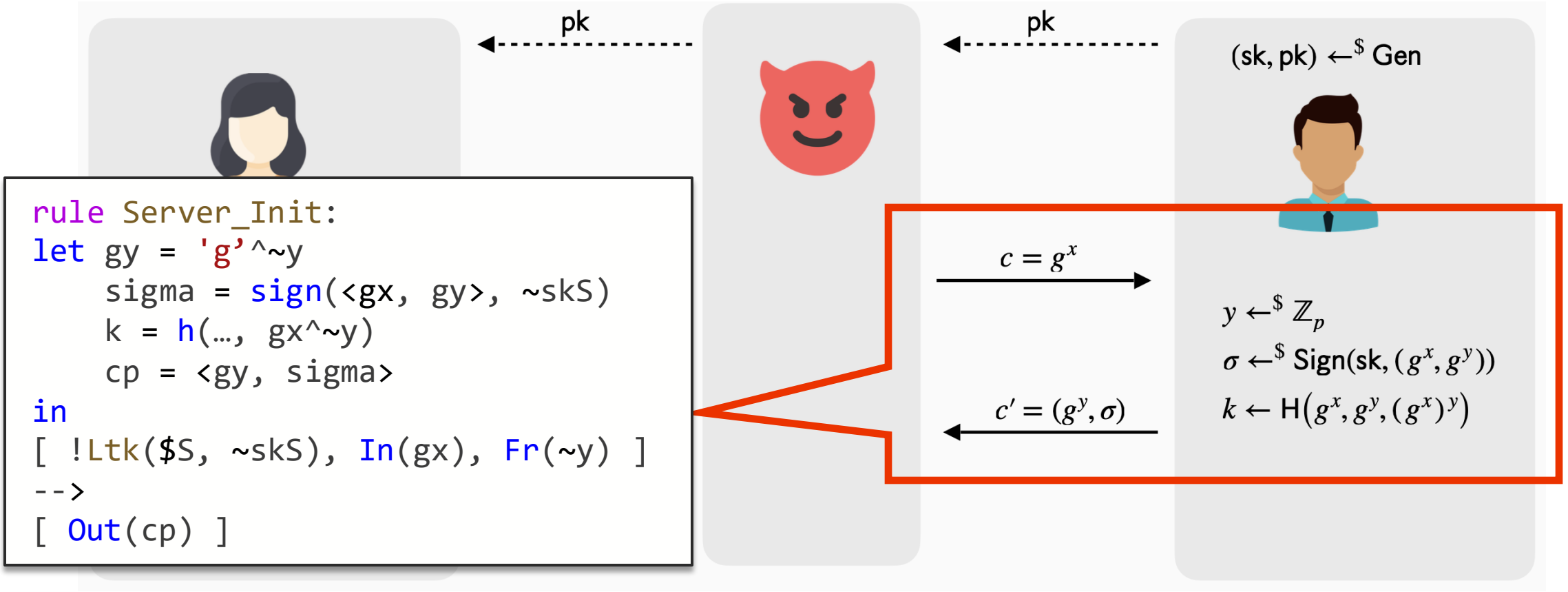
Key setup

‘c’	constant
~x	fresh type
$x	public type
!F(…)	persistent fact



Server

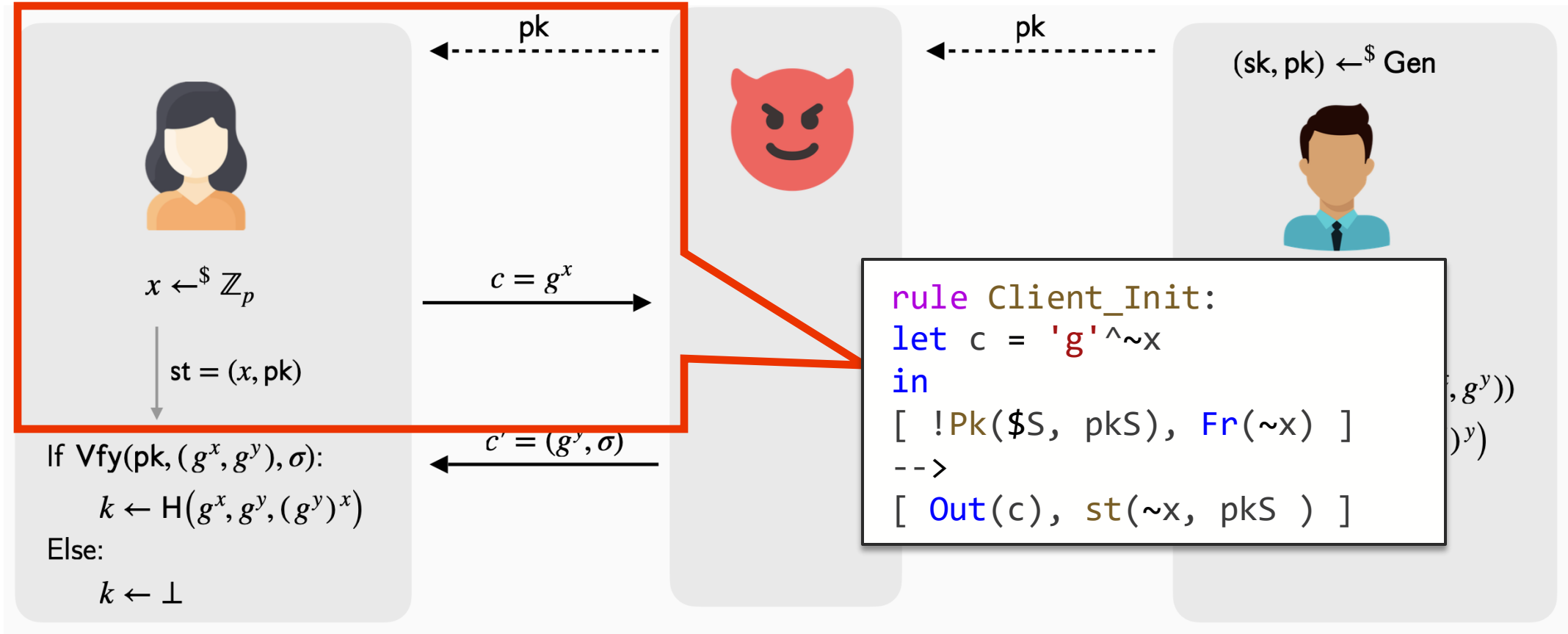
‘c’	constant
~x	fresh type
$x	public type
!F(…)	persistent fact




```
rule Server_Init:
let gy = 'g'^~y
    sigma = sign(<gx, gy>, ~skS)
    k = h(…, gx^~y)
    cp = <gy, sigma>
in
[ !Ltk($S, ~skS), In(gx), Fr(~y) ]
-->
[ Out(cp) ]
```

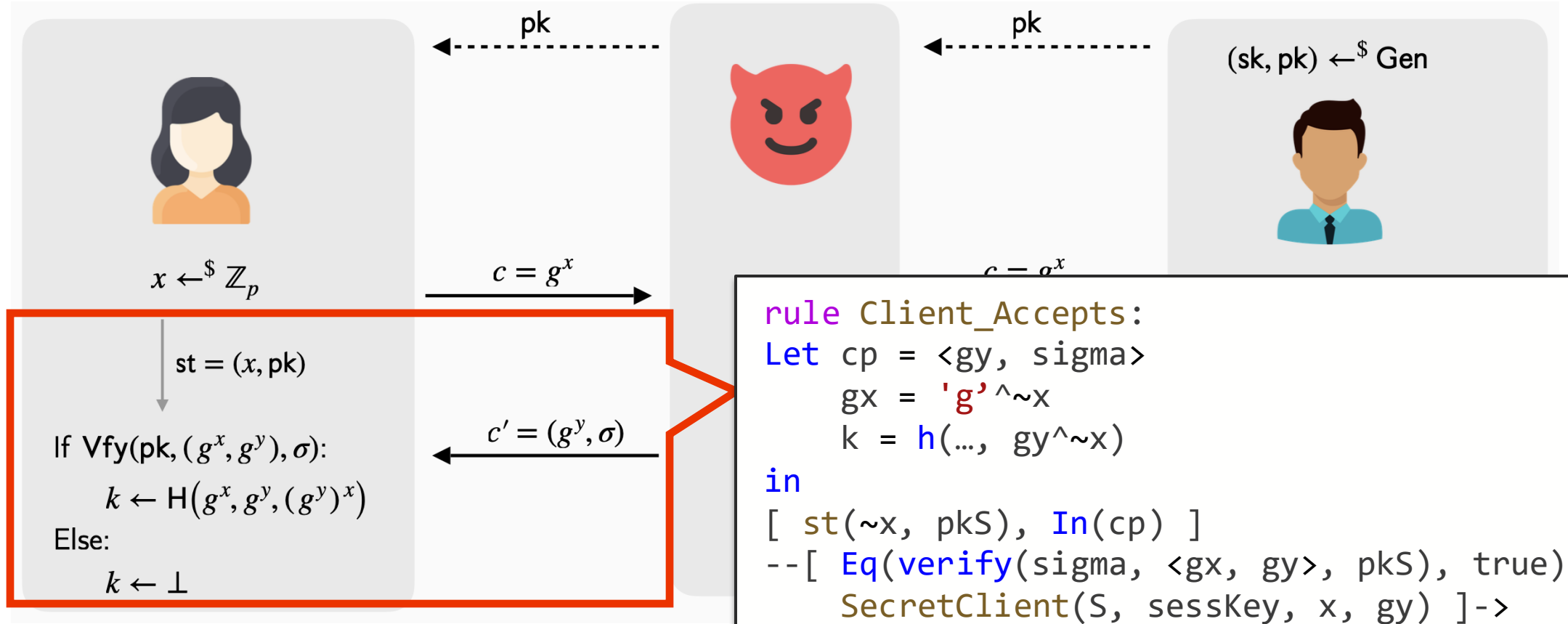
Client (1/2)

‘c’	constant
~x	fresh type
$x	public type
!F(…)	persistent fact



Client (2/2)

‘c’	constant
~x	fresh type
$x	public type
!F(…)	persistent fact

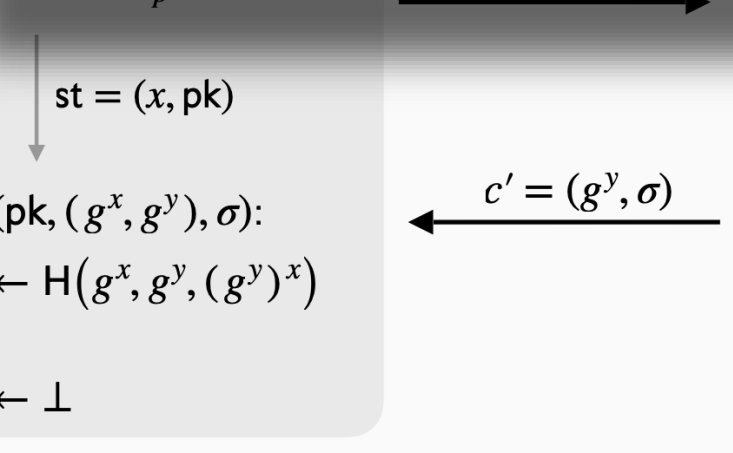



```
rule Client_Accepts:
Let cp = <gy, sigma>
    gx = 'g'^~x
    k = h(…, gy^~x)
in
[ st(~x, pkS), In(cp) ]
--[ Eq(verify(sigma, <gx, gy>, pkS), true),
    SecretClient(S, sessKey, x, gy) ]->
[ ]
```

Security properties

‘c’	constant
~x	fresh type
$x	public type
!F(…)	persistent fact

```
lemma SessionKey_Secrecy:
"All pkS sessKey x gy #i. SecretClient(pkS, sessKey, x, gy) @ #i
==>
not(Ex #j      . K(sessKey) @ #j)
| (Ex skS #j   . CompromiseLtk(pkS, skS)@ #j)
| (Ex gx #j    . CompromiseEphemeralKey(x, gx)@ #j)
| (Ex y  #j    . CompromiseEphemeralKey(y, gy)@ #j)"
```




```
rule Client_Accepts:
Let cp = <gy, sigma>
    gx = 'g'^~x
    k = h(…, gy^~x)
in
[ st(~x, pkS), In(cp) ]
--[ Eq(verify(sigma, <gx, gy>, pkS), true),
    SecretClient(S, sessKey, x, gy) ]->
[ ]
```

DEMO (command line)

DEMO (GUI)

But Tamarin can do much more…

- Modern models high double-digits number of rules, covering all modes of complex protocols in one go

Some examples:

- IETF TLS 1.3
- SPDM 1.2
- 5G-AKA
- EMV (Chip and pin)
- Noise protocol suite
- Apple iMessage PQ3

Tamarin found 18-messageattack on draft 10+ that breaks post-handshake authentication

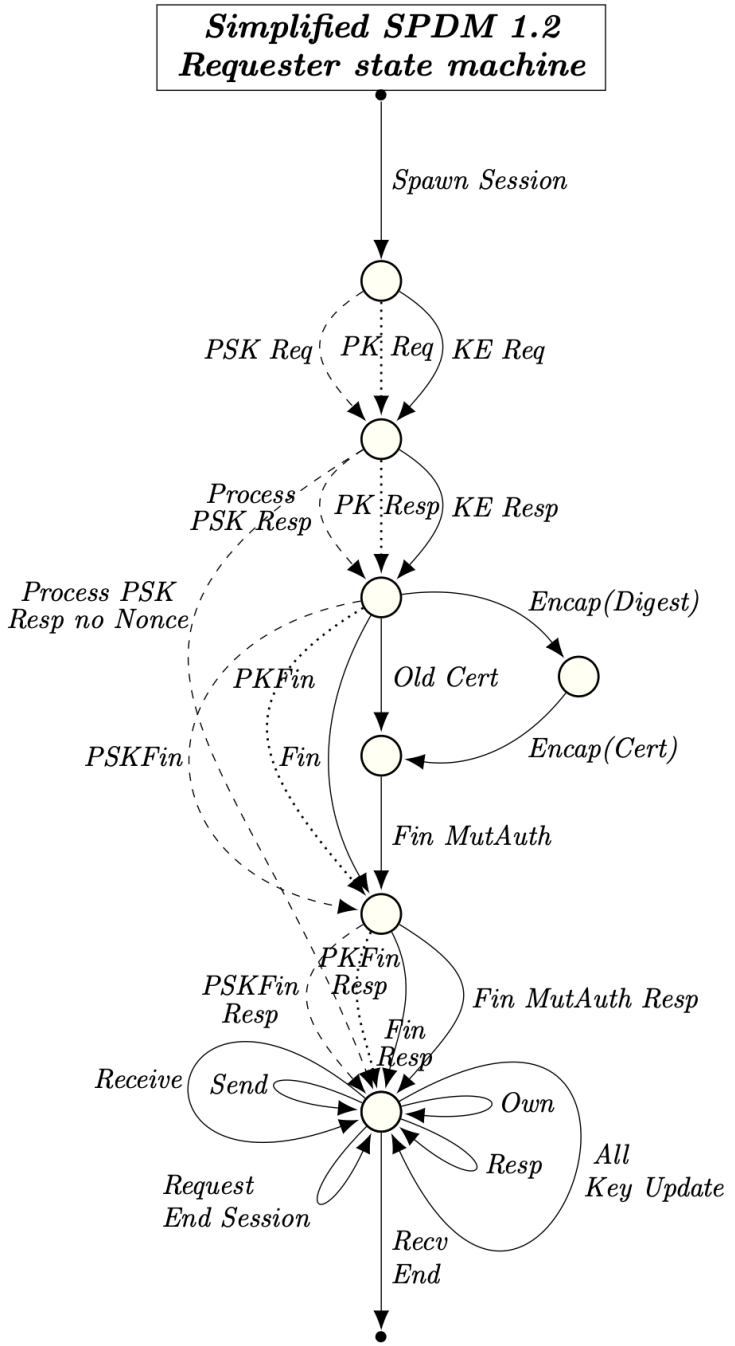
Tamarin found cross-protocol attack that breaks psk authenticaton

We can automate “the strongest property” that holds for each noise pattern

Complexity example: SPDM 1.2

- Simplified picture:
- Actual model nearly 70 rules

- Tamarin finds a cross-protocol attack that breaks PSK authentication
  - Works on reference implementation and Intel's rust implementation
  - CVE with score 9.0 (critical)
- Prove fixed version



Recent developments: Blurring the lines

- We are moving beyond traditional symbolic/Dolev-Yao features

- *"Perfect cryptography"*
  and
  "one symbolic model for each cryptographic primitive"

- New: a range of symbolic models for
  - Signatures
  - AEADs
  - Hash functions
  - DH/EC
  - KEMs

- *"Symbolic models explicitly specify the possible adversary operations"*

- We can avoid this by trace restrictions:
  Instead of explicitly specifying allowed transitions, generalize and restrict

A good starting point is:
https://ia.cr/2019/779

Reality, models, and gaps

Reality

Computational

Symbolic

Reality, models, and gaps

Reality

Computational

Symbolic

“SoK: Computer-Aided Cryptography”, BBBBCLP IEEE S&P 2021. IACR eprint 2019/1393

The need to create breathing room

Multiple approaches for provable security



In practice, the results are incomparable

- *We should not fight this but embrace it*
  - *Study* multiple approaches, don't discard
  - Need space and time for new ideas to come to fruition
    - **Reviewing**
      - "you didn't do everything"
      - "use other method instead"
    - **Writing**:
      - Framing of results:
      - "formally *verified*" "*provably secure*" unhelpful

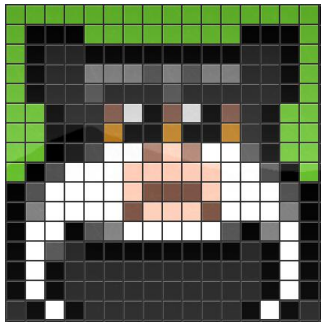
Conclusions

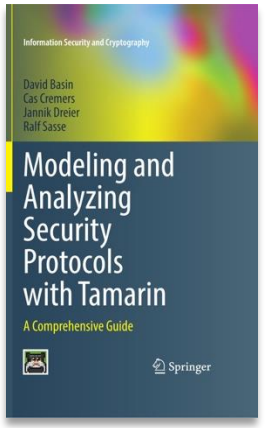
Conclusions: Tamarin

- Tamarin is a mature tool
- capable of dealing with highly complex models
- Offers state-of-the art features
- Proofs or attack-finding
- GUI enables guiding/inspecting partial proofs
- SAPIC+: applied-Pi processing
- Active development, user community, tutorials, book (soon: lecture slides based on book)

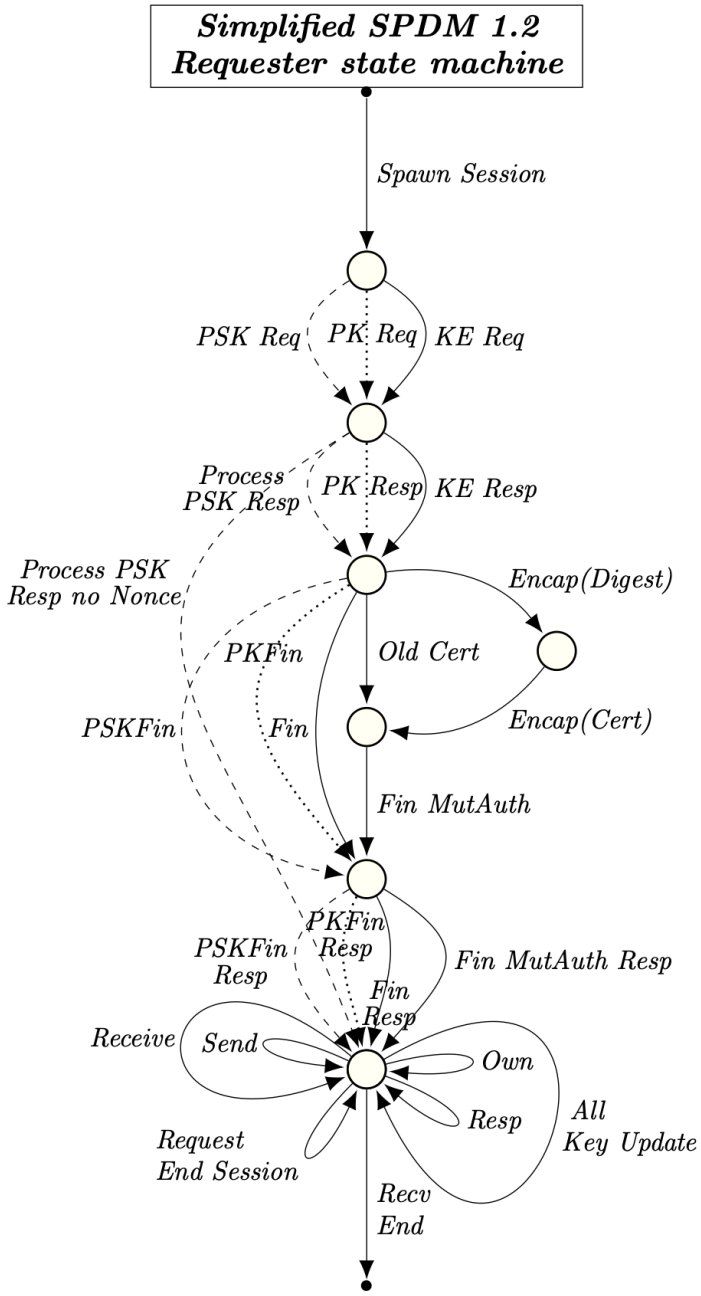
Cas Cremers – cremers@cispa.de

tamarin-prover.com





- IETF TLS 1.3
- SPDM 1.2
- 5G-AKA
- EMV (Chip and pin)
- Noise protocol suite
- Apple iMessage PQ3



References

- The Tamarin Team: **The Tamarin Prover**. https://tamarin-prover.com
- Simon Meier, Benedikt Schmidt, Cas Cremers, David A. Basin: **The TAMARIN Prover for the Symbolic Analysis of Security Protocols**. CAV 2013
- David A. Basin, Ralf Sasse, Jorge Toro-Pozo: **The EMV Standard: Break, Fix, Verify**. IEEE S&P 2021
- David A. Basin, Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, Ralf Sasse, Vincent Stettler: **A Formal Analysis of 5G Authentication**. CCS 2018
- Cas Cremers, Martin Dehnel-Wild: **Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion**. NDSS 2019
- Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, David A. Basin: **A Spectral Analysis of Noise: A Comprehensive, Automated, Formal Analysis of Diffie-Hellman Protocols**. USENIX Security Symposium 2020

References

- Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, Ralf Sasse: **Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures**. CCS 2019
- Cas Cremers, Dennis Jackson: **Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman**. CSF 2019
- Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, Bryan Parno: **SoK: Computer-Aided Cryptography**. IEEE S&P 2021
- Cas Cremers, Alexander Dax, Charlie Jacomme, Mang Zhao: **Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD Differences can impact Protocol Security**. USENIX Security Symposium 2023
- Vincent Cheval, Cas Cremers, Alexander Dax, Lucca Hirschi, Charlie Jacomme, Steve Kremer: **Hash Gone Bad: Automated discovery of protocol attacks that exploit hash function weaknesses**. USENIX Security Symposium 2023

References

- Felix Linker, Ralf Sasse, David A. Basin: **A Formal Analysis of Apple's iMessage PQ3 Protocol**. IACR Cryptol. ePrint Arch. 2024: 1395
- Cas Cremers, Alexander Dax, Aurora Naska: **Breaking and Provably Restoring Authentication: A Formal Analysis of SPDM 1.2 including Cross-Protocol Attacks**. CCS 2025
- Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe: **Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication**. IEEE Symposium on Security and Privacy 2016