INRIA Paris-Rocquencourt – 30 May 2013

application for CR2 in Prosecco team

Formally Verified Privacy-Preserving Distributed Applications

Cătălin Hrițcu

Penn
UNIVERSITY of PENNSYLVANIA

Today's computer systems are insecure

***Formal methods** will play a crucial role in building significantly more secure systems*

Formal methods, broadly

- language design
- rigorous semantics
- specification
- verification
- type systems
- proof assistants
- runtime monitoring
- code generation
- code transformation
- automatic testing
- ...

Formal methods, broadly

- language design
- rigorous semantics
- specification
- verification
- type systems
- proof assistants
- runtime monitoring
- code generation
- code transformation
- automatic testing
- ...

clearly specifying security goals

+ techniques for achieving these goals
(e.g. static analysis or dynamic enforcement)

+ showing that goals were achieved

Formal methods, broadly

- language design
- rigorous semantics
- specification
- verification
- type systems
- proof assistants
- runtime monitoring
- code generation
- code transformation
- automatic testing
- ...

clearly specifying security goals

+ techniques for achieving these goals
(e.g. static analysis or dynamic enforcement)

+ showing that goals were achieved

all these tools are potentially useful

choose the set of tools that
best solves the problem at hand
(cost vs benefit analysis)

Contributions

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012]

Contributions

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012]
2. **Internship & later collaboration @ Microsoft Research Cambridge:**
   - Data processing (Microsoft Excel 2013 “Data Explorer” formula language); refinement types; semantic subtyping using SMT solver; verification condition generator [ICFP 2010, CPP 2011, JFP 2012]

Contributions

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012]
2. **Internship & later collaboration @ Microsoft Research Cambridge:**
   - Data processing (Microsoft Excel 2013 “Data Explorer” formula language); refinement types; semantic subtyping using SMT solver; verification condition generator [ICFP 2010, CPP 2011, JFP 2012]
3. **PostDoc @ UPenn: CRASH/SAFE project** (academia-industry collaboration; clean-slate co-design of secure architecture: hardware+system+language)
   - Robust exception handling for dynamic information flow control [IEEE S&P (Oakland) 2013]
   - Ongoing work: testing and formally verifying information flow machine [ICFP 2013 submission on random testing, working draft on Coq proofs]

🐓 - Coq formalization

Contributions

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis] 🐓🐓
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012] 🐓
2. **Internship & later collaboration @ Microsoft Research Cambridge:**
   - Data processing (Microsoft Excel 2013 “Data Explorer” formula language); refinement types; semantic subtyping using SMT solver; verification condition generator [ICFP 2010, CPP 2011, JFP 2012] 🐓🐓
3. **PostDoc @ UPenn: CRASH/SAFE project** (academia-industry collaboration; clean-slate co-design of secure architecture: hardware+system+language)
   - Robust exception handling for dynamic information flow control [IEEE S&P (Oakland) 2013] 🐓
   - Ongoing work: testing and formally verifying information flow machine [ICFP 2013 submission on random testing, working draft on Coq proofs] 🐓

Contributions

🐓 - Coq formalization

🛠 - tool / software

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis] 🐓🐓🛠🛠
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012] 🐓🛠
2. **Internship & later collaboration @ Microsoft Research Cambridge:**
   - Data processing (Microsoft Excel 2013 “Data Explorer” formula language); refinement types; semantic subtyping using SMT solver; verification condition generator [ICFP 2010, CPP 2011, JFP 2012] 🐓🐓🛠🛠
3. **PostDoc @ UPenn: CRASH/SAFE project** (academia-industry collaboration; clean-slate co-design of secure architecture: hardware+system+language)
   - Robust exception handling for dynamic information flow control [IEEE S&P (Oakland) 2013] 🐓🛠
   - Ongoing work: testing and formally verifying information flow machine [ICFP 2013 submission on random testing, working draft on Coq proofs] 🐓🛠

Contributions

🐓 - Coq formalization

🛠 - tool / software

later

1. **PhD @ Saarland University: Security protocols**
   - Expressive type systems – the first one for zero-knowledge proofs [CCS 2008, CSF 2009, TOSCA 2011, PhD thesis] 🐓🐓🛠🛠
   - Defining security of e-voting protocols; ProVerif (Prosecco) [CSF 2008]
   - Expi2Java: code generator for realistic protocols (TLS) [NFM 2012] 🐓🛠
2. **Internship & later collaboration @ Microsoft Research Cambridge:**
   - Data processing (Microsoft Excel 2013 “Data Explorer” formula language); refinement types; semantic subtyping using SMT solver; verification condition generator [ICFP 2010, CPP 2011, JFP 2012] 🐓🐓🛠🛠
3. **PostDoc @ UPenn: CRASH/SAFE project** (academia-industry collaboration; clean-slate co-design of secure architecture: hardware+system+language)
   - Robust exception handling for dynamic information flow control [IEEE S&P (Oakland) 2013] 🐓🛠
   - Ongoing work: testing and formally verifying information flow machine [ICFP 2013 submission on random testing, working draft on Coq proofs] 🐓🛠

Privacy in the age of “cloud computing”

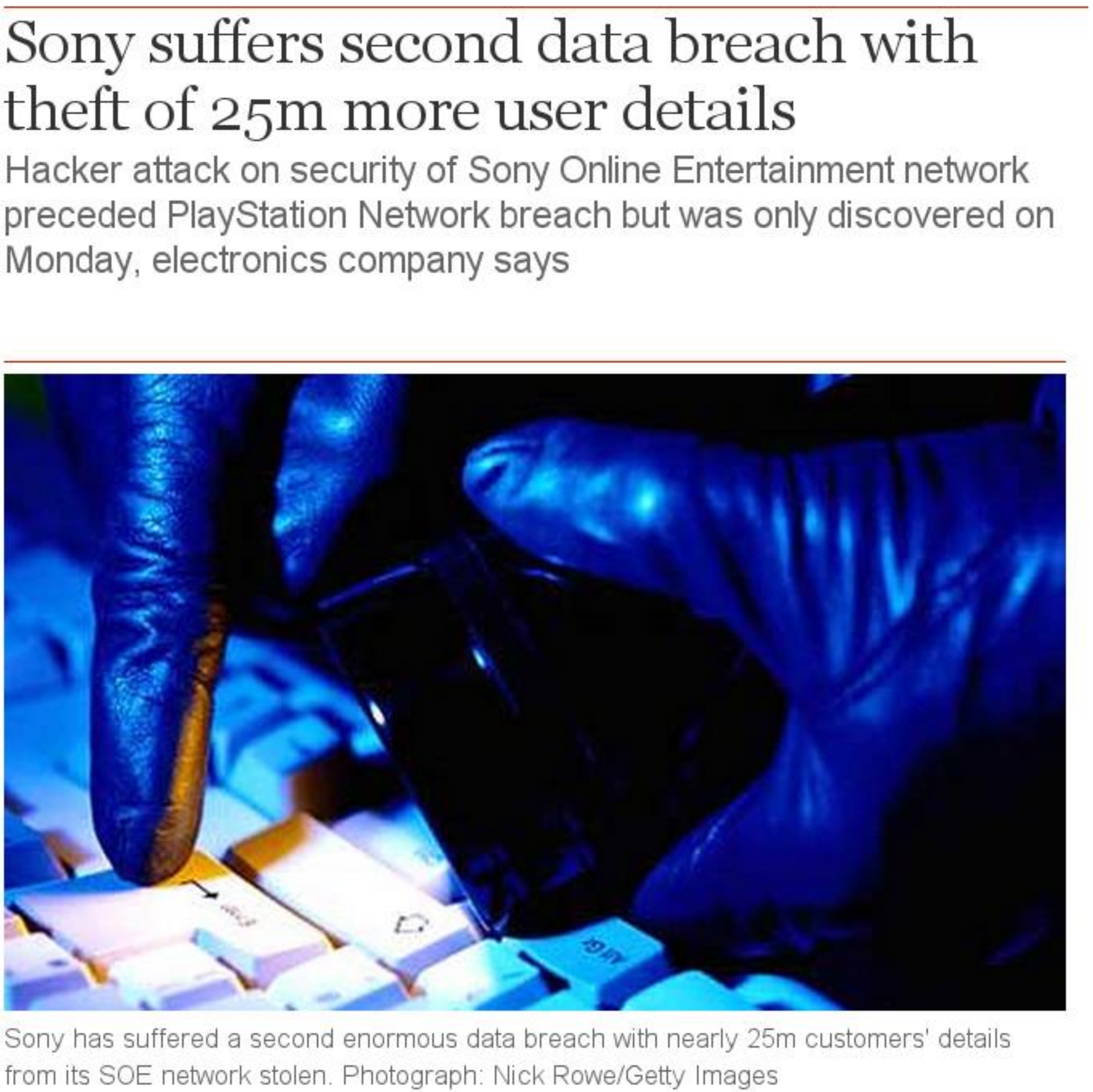
- **lack of online privacy is one of the biggest problems of our time**
  - technology is causing the problem
  - solution not simple and not solely technologic
    - also social, legal, economic, behavioral, philosophical

Privacy in the age of “cloud computing”

- **lack of online privacy is one of the biggest problems of our time**
  - technology is causing the problem
  - solution not simple and not solely technologic
    - also social, legal, economic, behavioral, philosophical
- **“cloud computing” is making this worse**
  - in order to obtain service, users have to entrust private information to 3rd party service providers that gather the data of millions of users

Privacy in the age of "cloud computing"

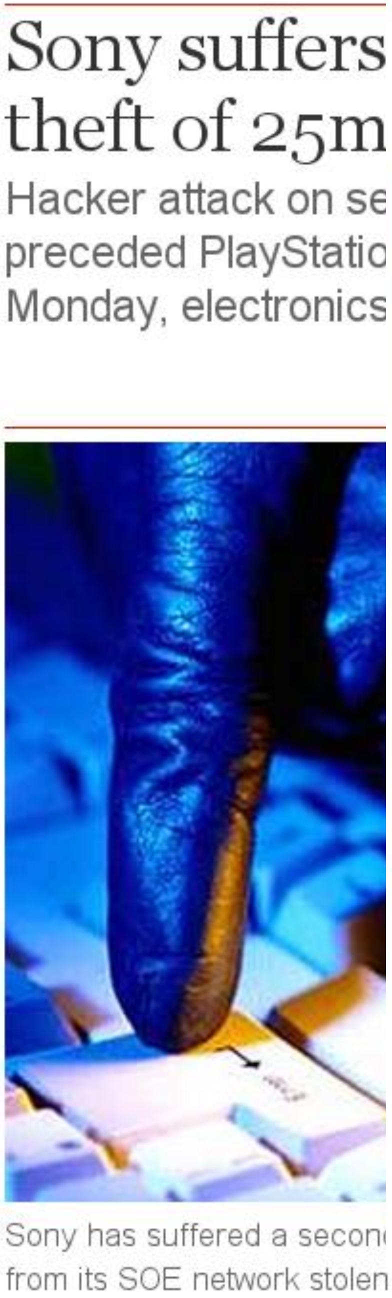
- **lack of online privacy is one of the biggest problems of our time**
  - technology is causing the problem
  - solution not simple and not solely technologic
    - also social, legal, economic, behavioral, philosophical
- **"cloud computing" is making this worse**
  - in order to obtain service, users have to entrust private information to 3rd party service providers that gather the data of millions of users
  - what could possibly go wrong?



Sony suffers second data breach with theft of 25m more user details

Hacker attack on security of Sony Online Entertainment network preceded PlayStation Network breach but was only discovered on Monday, electronics company says

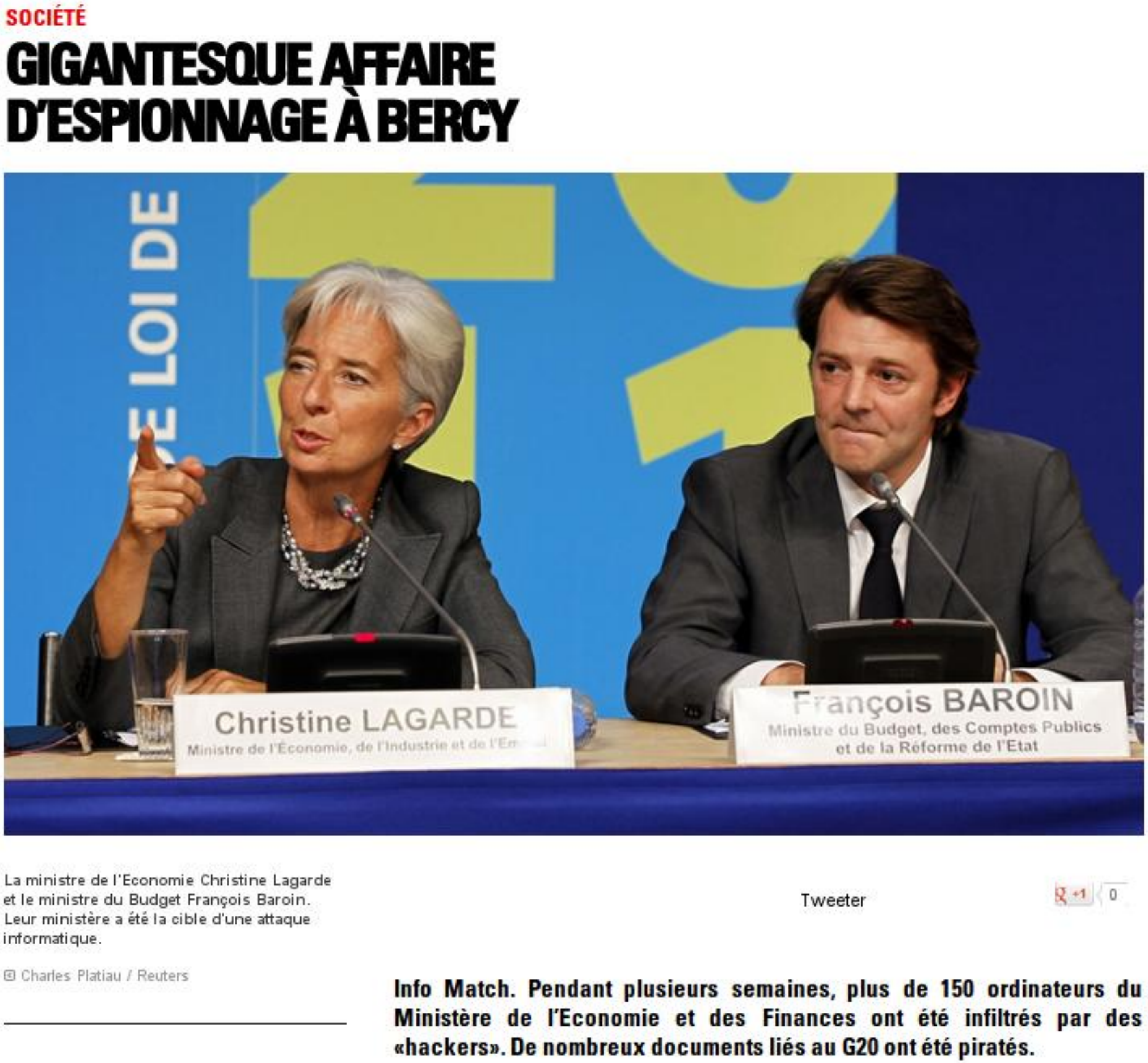
Sony has suffered a second enormous data breach with nearly 25m customers' details from its SOE network stolen. Photograph: Nick Rowe/Getty Images



Sony suffers theft of 25m

Hacker attack on se preceded PlayStatio Monday, electronics

Sony has suffered a secon from its SOE network stolen. Photograph: Nick Rowe/Getty Images



SOCIÉTÉ

GIGANTESQUE AFFAIRE D'ESPIONNAGE À BERCY

E LOI DE
Christine LAGARDE
Ministre de l'Économie, de l'Industrie et de l'Em
François BAROIN
Ministre du Budget, des Comptes Publics et de la Réforme de l'Etat

La ministre de l'Economie Christine Lagarde et le ministre du Budget François Baroin. Leur ministère a été la cible d'une attaque informatique.

© Charles Platiau / Reuters

Tweeter

+1 0

Info Match. Pendant plusieurs semaines, plus de 150 ordinateurs du Ministère de l'Economie et des Finances ont été infiltrés par des «hackers». De nombreux documents liés au G20 ont été piratés.

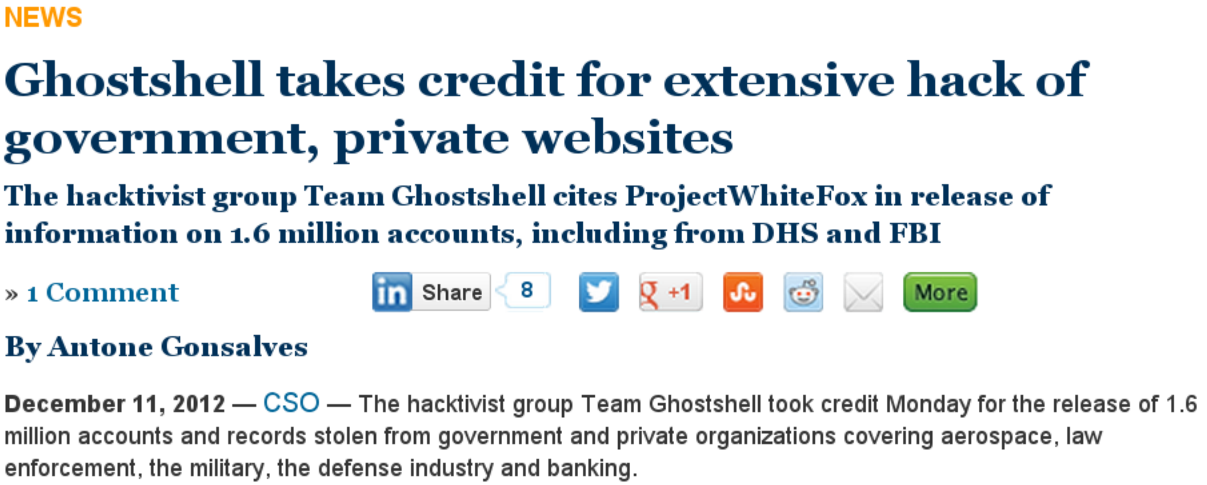
Sony suffers theft of 25m

Hacker attack on se
preceded PlayStatio
Monday, electronics

SOCIÉTÉ

GIGANTESQUE AFFAIRE D'ESPIONNAGE À BERCY

DE LOI DE
Christine LAGARDE
François BAROIN
Ministre du Budget, des Comptes Publics et de la Réforme de l'Etat



NEWS

Ghostshell takes credit for extensive hack of government, private websites

The hacktivist group Team Ghostshell cites ProjectWhiteFox in release of information on 1.6 million accounts, including from DHS and FBI

» 1 Comment

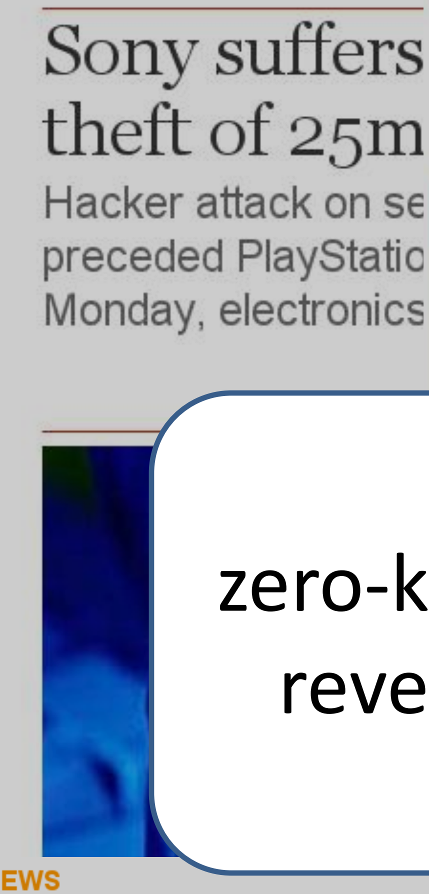
By Antone Gonsalves

December 11, 2012 — CSO — The hacktivist group Team Ghostshell took credit Monday for the release of 1.6 million accounts and records stolen from government and private organizations covering aerospace, law enforcement, the military, the defense industry and banking.

Tweeter

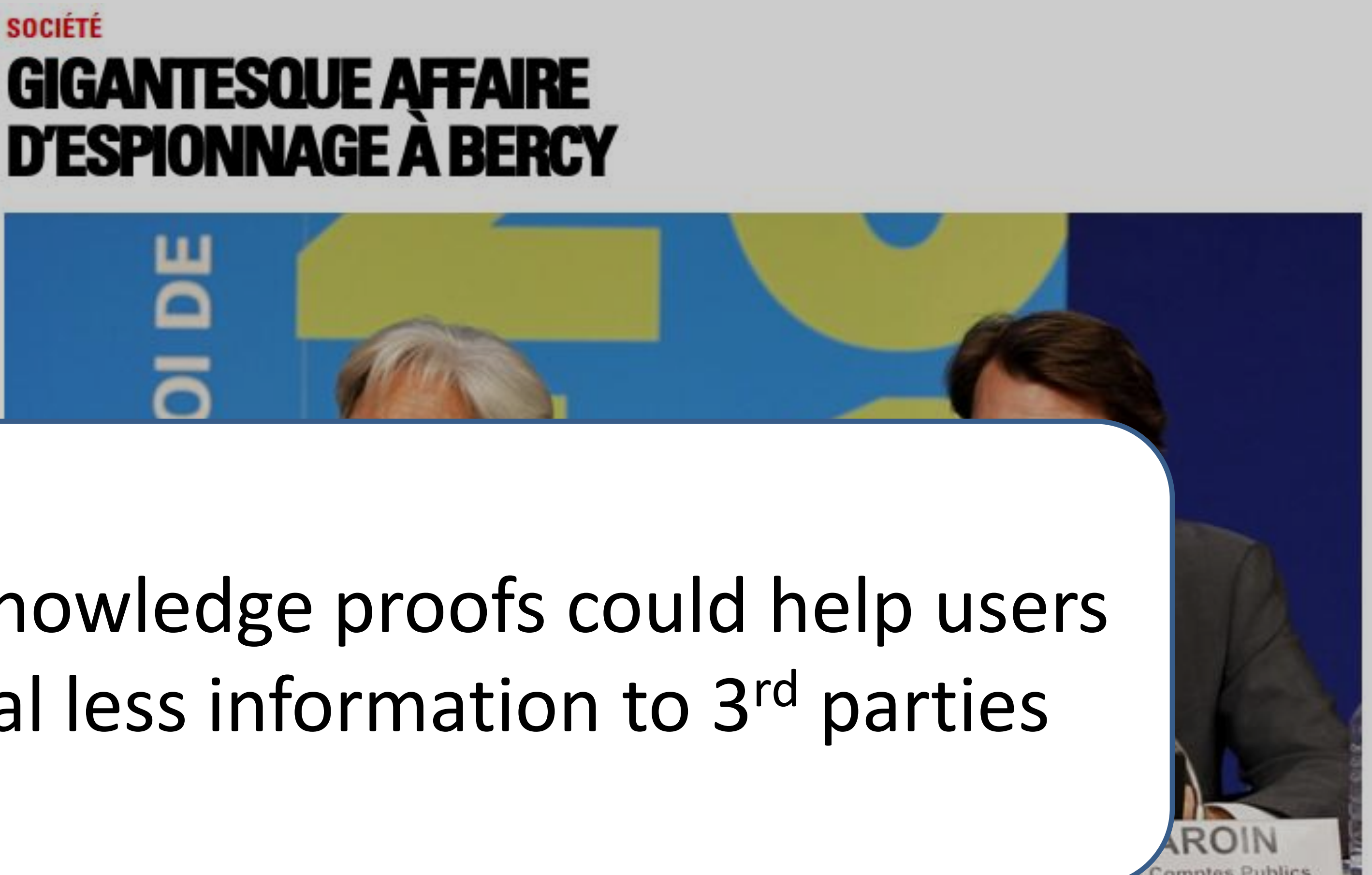
+1 0

t plusieurs semaines, plus de 150 ordinateurs du omie et des Finances ont été infiltrés par des eux documents liés au G20 ont été piratés.



Sony suffers theft of 25m

Hacker attack on se
preceded PlayStatio
Monday, electronics



SOCIÉTÉ

GIGANTESQUE AFFAIRE D'ESPIONNAGE À BERCY

zero-knowledge proofs could help users reveal less information to 3rd parties

NEWS

Ghostshell takes credit for extensive hack of government, private websites

The hacktivist group Team Ghostshell cites ProjectWhiteFox in release of information on 1.6 million accounts, including from DHS and FBI

» 1 Comment

By Antone Gonsalves

December 11, 2012 — CSO — The hacktivist group Team Ghostshell took credit Monday for the release of 1.6 million accounts and records stolen from government and private organizations covering aerospace, law enforcement, the military, the defense industry and banking.

AROIN
Comptes Publics
et de la Réforme de l'Etat

Tweeter

+1 0

t plusieurs semaines, plus de 150 ordinateurs du
omie et des Finances ont été infiltrés par des
eux documents liés au G20 ont été piratés.

Applications of zero-knowledge proofs have skyrocketed in recent years

anonymous authentication

electronic voting

privacy-preserving
digital identity management

e-cash

electronic auctions

security despite
compromise

privacy-friendly
smart metering

anonymous trust
and reputation

decentralized
social networks

risk assurance
for hedge funds

anonymous credentials

biometric authentication

anonymous electronic ticketing
for public transportation

Achieving privacy with zero-knowledge

Alice proves to online store that she is over 18, without revealing her age

protecting personal information

+

digital credentials (authorization)

→

IBM
idemix
PrimeLife
Microsoft
U-Prove
ABC4TRUST

privacy-preserving digital identity management

Achieving privacy with zero-knowledge

Alice proves to online store that she is over 18, without revealing her age

protecting personal information

+

digital credentials (authorization)

→

IBM idemix

PrimeLife

Microsoft U-Prove

ABC4TRUST

privacy-preserving digital identity management

vote privacy coercion resistance

[CSF 2008]

+

verifiability (software independence)

→

remote electronic voting

Achieving privacy with zero-knowledge

Alice proves to online store that she is over 18, without revealing her age

protecting personal information

+

digital credentials (authorization)

→

privacy-preserving digital identity management

vote privacy coercion resistance

[CSF 2008]

+

verifiability (software independence)

→

remote electronic voting

Zero-knowledge proofs, by example

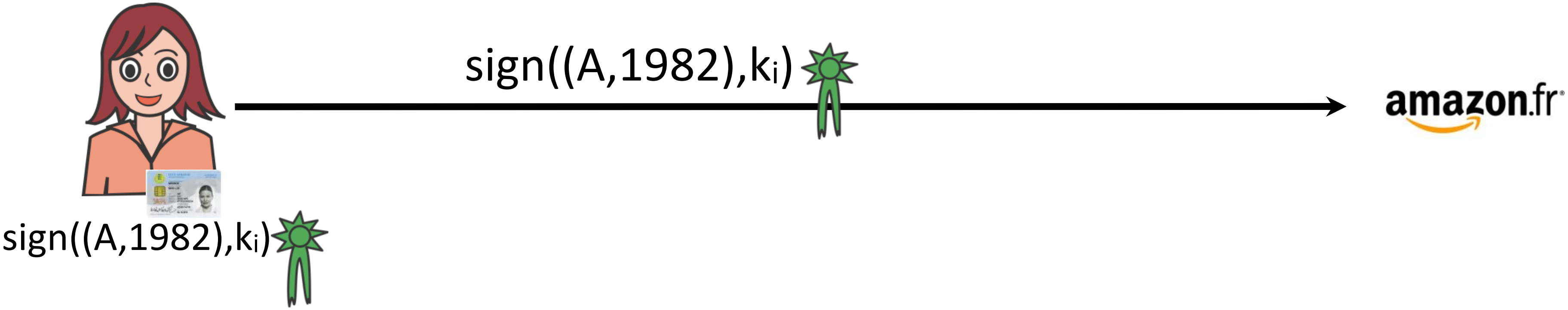
Alice proves to online store that she is over 18

sign((A,1982),k_i)

amazon.fr

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18

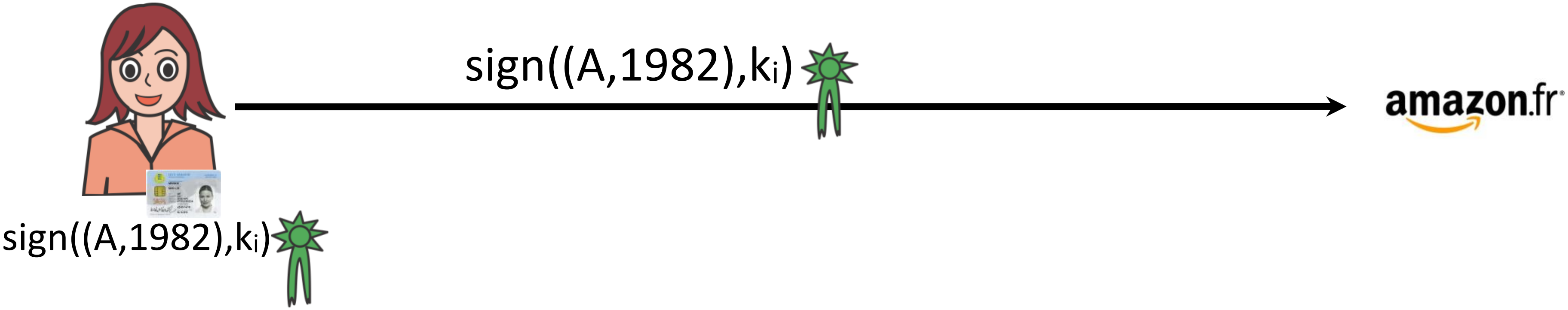


sign((A,1982),k_i)

sign((A,1982),k_i)

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age

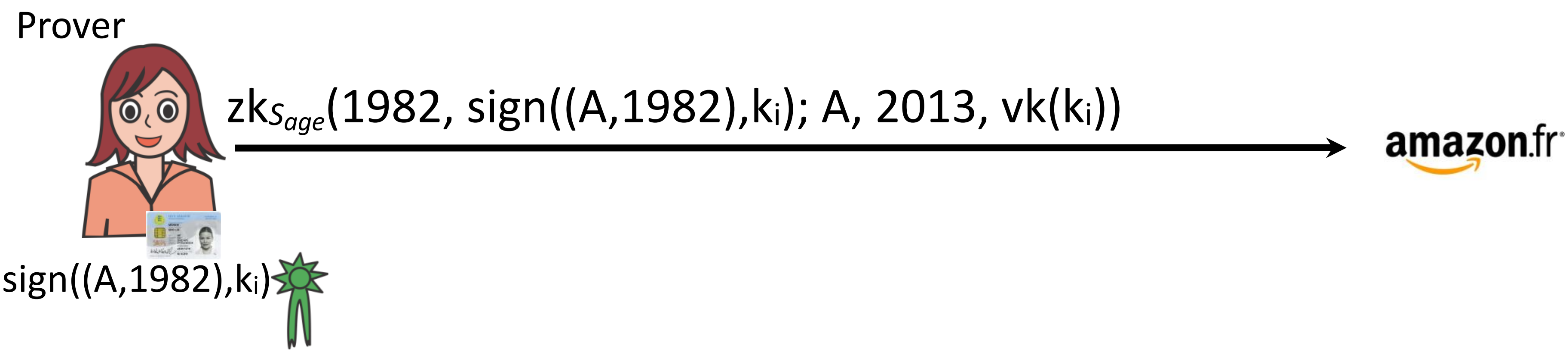


sign((A,1982),k_i)

sign((A,1982),k_i)

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age



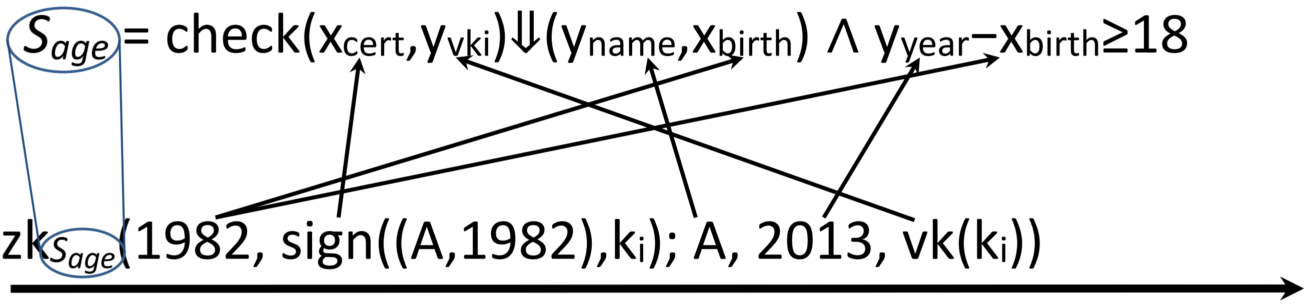
Prover

$zk_{S_{age}}(1982, sign((A,1982),k_i); A, 2013, vk(k_i))$

$sign((A,1982),k_i)$

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age



$S_{age} = \mathrm{check}(x_{cert}, y_{vki}) \Downarrow (y_{name}, x_{birth}) \wedge y_{year} - x_{birth} \geq 18$

$zk_{S_{age}}(1982, \mathrm{sign}((A,1982),k_i); A, 2013, \mathrm{vk}(k_i))$

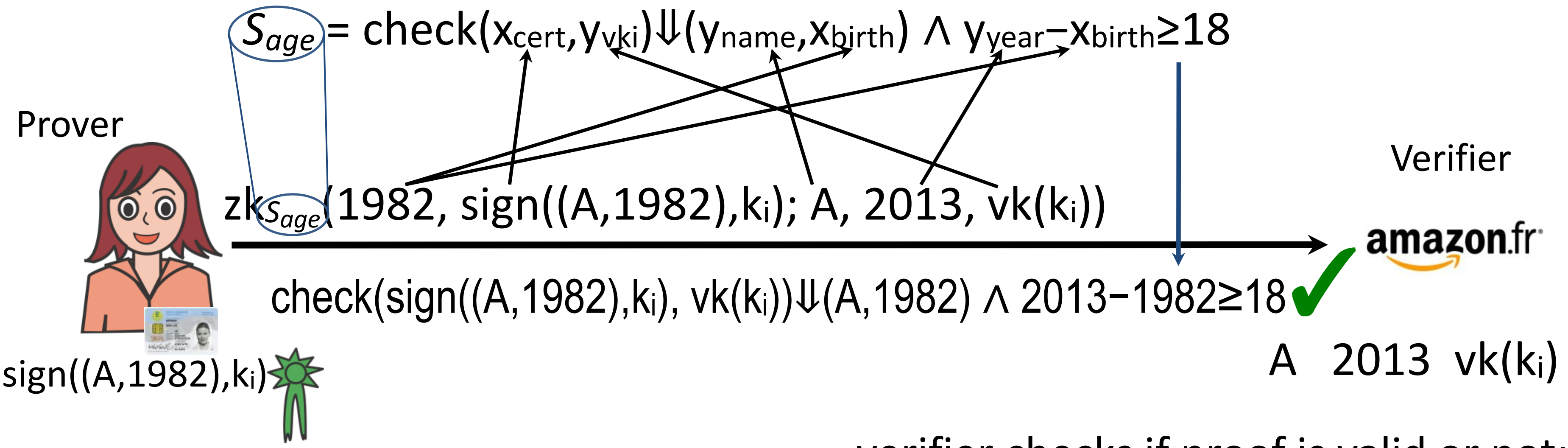
Prover

amazon.fr

$\mathrm{sign}((A,1982),k_i)$

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age



S_{age} = $\mathrm{check}(x_{cert}, y_{vki}) \Downarrow (y_{name}, x_{birth}) \wedge y_{year} - x_{birth} \geq 18$

Prover

Verifier

$\mathrm{zk}_{S_{age}}(1982, \mathrm{sign}((A,1982),k_i); A, 2013, \mathrm{vk}(k_i))$

$\mathrm{check}(\mathrm{sign}((A,1982),k_i), \mathrm{vk}(k_i)) \Downarrow (A,1982) \wedge 2013 - 1982 \geq 18$ ✓

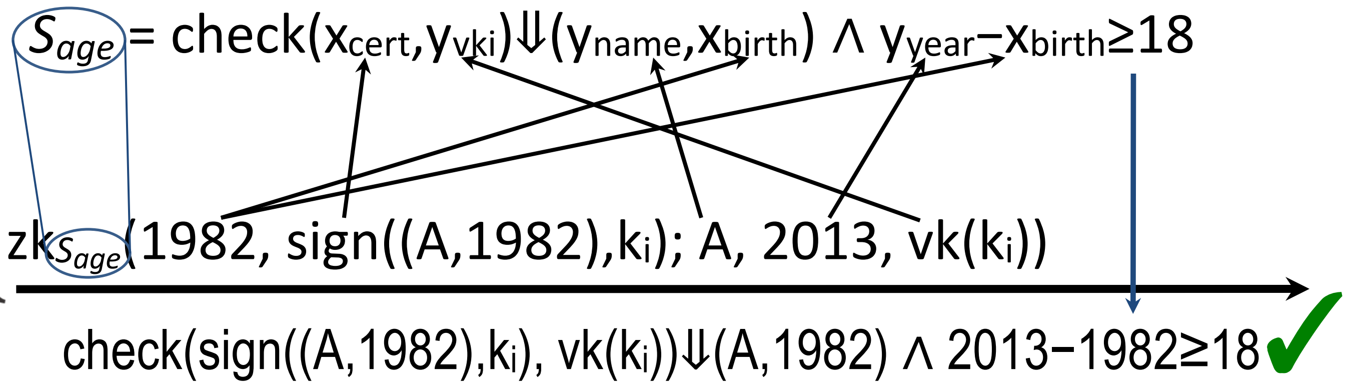
$\mathrm{sign}((A,1982),k_i)$

$A \quad 2013 \quad \mathrm{vk}(k_i)$

verifier checks if proof is valid or not;
finds out public arguments (ys)

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age



S_{age} = check(x_{cert}, y_{vki})⇓(y_{name}, x_{birth}) ∧ $y_{year} - x_{birth} \geq 18$

$zk_{S_{age}}$(1982, sign((A,1982),k_i); A, 2013, vk(k_i))

check(sign((A,1982),k_i), vk(k_i))⇓(A,1982) ∧ 2013−1982≥18 ✓

Prover

sign((A,1982),k_i)

Verifier

amazon.fr

A 2013 vk(k_i)

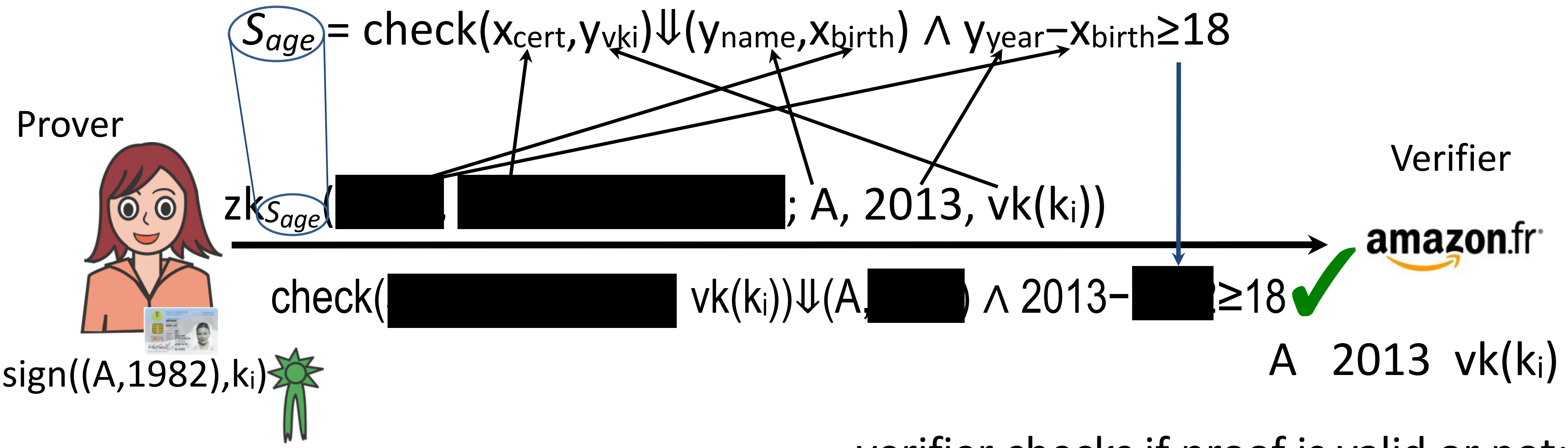
verifier checks if proof is valid or not;
finds out public arguments (ys)

zero-knowledge:

no information is revealed about witnesses (xs) beyond the validity of the statement

Zero-knowledge proofs, by example

Alice proves to online store that she is over 18, without revealing her age



S_{age} = check(x_{cert},y_{vki})$\Downarrow$(y_{name},x_{birth}) $\wedge$ y_{year}−x_{birth}≥18

Prover

zkS_{age}([redacted], [redacted]; A, 2013, vk(k_i))

Verifier

check([redacted] vk(k_i))$\Downarrow$(A,[redacted]) $\wedge$ 2013−[redacted]≥18 ✓

A 2013 vk(k_i)

sign((A,1982),k_i)

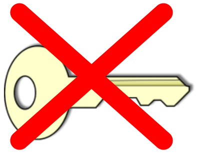
verifier checks if proof is valid or not;
finds out public arguments (ys)

zero-knowledge:
no information is revealed about witnesses (xs)
beyond the validity of the statement

Challenges of type-checking zero-knowledge

[CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]

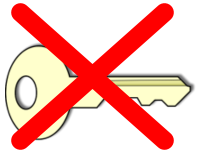
- **Zk-proofs don’t depend on crypto keys**
  - previous type systems rely on assigning types to keys
  - ***solution**: assign types to each zk-statement*
    - refinement type “$T_{Sage}=\{y_{name}:\mathrm{Un},\ldots \mid \exists x_{birth}.\ \mathrm{Send}(y_{name},x_{birth})\}$”
    - type-checker enforces this strong type on **honest** provers



Challenges of type-checking zero-knowledge

[CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]

- **Zk-proofs don't depend on crypto keys**
  - previous type systems rely on assigning types to keys
  - ***solution*: *assign types to each zk-statement***
    - refinement type "$T_{S_{age}}$={y_{name}:Un,...|∃x_{birth}. Send(y_{name},x_{birth})}"
    - type-checker enforces this strong type on **honest** provers

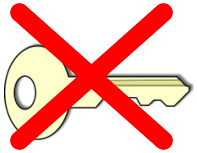


- **Attacker can also produce valid zk-proofs**
  - successfully verifying zk-proof
    - guarantees "$\exists x_{birth},x_{cert}.\ \overbrace{\text{check}(x_{cert},y_{vki})\Downarrow(y_{name},x_{birth})\wedge y_{year}-x_{birth}\geq 18}^{S_{age}}$"
    - guarantees $T_{S_{age}}$ **only if** type-checker can **infer** that the verified zk-proof was produced by honest prover (i.e. type-checked)
  - ***solution*: *statement-based inference + intersection types*** (∧)
    + reasoning about type disjointness (Un∧Private=∅)

Challenges of type-checking zero-knowledge

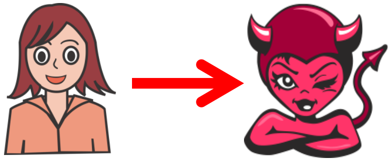
[CCS 2008, CSF 2009, TOSCA 2011, PhD thesis]

- **Zk-proofs don't depend on crypto keys**
  - previous type systems rely on assigning types to keys
  - ***solution**: assign types to each zk-statement*
    - refinement type "$T_{S_{age}}=\{y_{name}{:}Un, \ldots \mid \exists x_{birth}.\ Send(y_{name}, x_{birth})\}$"
    - type-checker enforces this strong type on **honest** provers



- **Attacker can also produce valid zk-proofs**
  - successfully verifying zk-proof
    - guarantees "$\exists x_{birth}, x_{cert}.\ \overbrace{check(x_{cert}, y_{vki}) \Downarrow (y_{name}, x_{birth}) \wedge y_{year} - x_{birth} \geq 18}^{S_{age}}$"
    - guarantees $T_{S_{age}}$ **only if** type-checker can **infer** that the verified zk-proof was produced by honest prover (i.e. type-checked)
  - ***solution**: statement-based inference + intersection types* (∧) ***+ reasoning about type disjointness*** (Un∧Private=∅)

- **Participants can be dynamically compromised**
  - inferred types conditioned on participants' honesty
  - ***solution**: union types* {Private|¬Bad(A)} ∨ {Un|Bad(A)} ***+ logical subtyping***
  - automatically strengthened protocols [CSF 2009]



Type-checking zero-knowledge

- first type systems to analyze zk-protocols
  [CCS 2008, TOSCA 2011, PhD thesis]
- same ideas for protocol models (π)
  & simple implementations (λ)
- formalized, implemented, experimented
  - type-checkers **used independently in other projects**

Why isn’t this enough?

- many real zk-applications are **beyond current state of the art** in automatic protocol analysis; my previous type systems:
  - largest example: simplified DAA ~250 lines of λ-calculus (RCF)
  - only authorization (robust safety), not “privacy”
  - only non-interactive zero-knowledge
  - crypto assumed perfect (symbolic model)

Goals of this proposal

- remove these limitations
- make the design, analysis, and correct implementation of zk-applications practical
- verify implementations of real zk-applications
- produce better generally-useful tools

Goals of this proposal

- remove these limitations
- make the design, analysis, and correct implementation of zk-applications practical
- verify implementations of real zk-applications ←
- produce better generally-useful tools

Verifying real zk-applications

- privacy-preserving digital identity management (e.g. idemix, UProve), e-voting (e.g. Civitas/CaveatCoercitor), and e-cash (e.g. ZeroCoin)

Verifying real zk-applications

- privacy-preserving digital identity management (e.g. idemix, UProve), e-voting (e.g. Civitas/CaveatCoercitor), and e-cash (e.g. ZeroCoin)
- formalize **end-to-end security properties** (i.e. “privacy”)
  - relational, quantitative, probabilistic

Verifying real zk-applications

- privacy-preserving digital identity management (e.g. idemix, UProve), e-voting (e.g. Civitas/CaveatCoercitor), and e-cash (e.g. ZeroCoin)
- formalize **end-to-end security properties** (i.e. “privacy”)
  - relational, quantitative, probabilistic
- devise **sound automated verification tools** that
  - work for **real code** (not abstract models)
  - provide **strong guarantees** (computational crypto)
  - support non-interactive + **interactive** zero-knowledge

Verifying real zk-applications

- privacy-preserving digital identity management (e.g. idemix, UProve), e-voting (e.g. Civitas/CaveatCoercitor), and e-cash (e.g. ZeroCoin)
- formalize **end-to-end security properties** (i.e. “privacy”)
  - relational, quantitative, probabilistic
- devise **sound automated verification tools** that
  - work for **real code** (not abstract models)
  - provide **strong guarantees** (computational crypto)
  - support non-interactive + **interactive** zero-knowledge

2 ways to approach this; capitalize previous experience (mine + Prosecco)

Short term objectives (1/2)

1. **reimplement applications in OCaml/F#**
   and use new, very expressive type-checker
   - combine the strengths of existing type systems
     - F5: non-interactive zero-knowledge [TOSCA 2011, PhD thesis]
     - F7: computational guarantees (Prosecco)
     - F*: relational properties (Prosecco)
   - *challenge*: devise this super expressive type system
   - *challenge*: interactive zero-knowledge proofs
     - fixed interaction pattern (e.g. to Σ-protocols)

Short term objectives (2/2)

2. **generate code from verified abstract models**
   - extend CryptoVerif (Prosecco) to zero-knowledge proofs
     - add indistinguishability axioms (e.g. zero-knowledge property)
     - *challenges*: existentials (Skolemize?), guarded rewriting
   - code generator targeting mainstream language like C
     - experience: Expi2Java [NFM 2012], CryptoVerif2OCaml (Prosecco)
     - zero-knowledge implementation is statement dependent
       - use existing cryptographic compiler – e.g. ZKCrypt (IMDEA)
     - *challenge*: security of translation wrt. formal semantics of C

More speculative ideas

- tools aiding **design** of privacy-preserving applications
  - **automated synthesis** from high-level specifications
  - **privacy-enhancing transformations**
- studying **other general privacy-enhancing techniques**
  - secure multi-party computation
  - (fully) homomorphic encryption

Cătălin Hrițcu

- Publications:

  best conferences in security

  - *conferences (8)*: IEEE S&P, ACM CCS, 2 x IEEE CSF, ACM ICFP, ...
  - *journals (2)*; *textbook (1)*; *workshops (6); under review (2)*
- Software: >67.6k lines of code
  - 13.2k OCaml/F#, 9.1k Haskell, 16k Java,
    20.1k Breeze, 5.3k π-calculus, 3.9k λ-calculus (RCF)
- Machine-checked formalizations: >57k lines of Coq
- MSc + PhD Fellowships from Microsoft Research & MPI (IMPRS)
- Günter Hotz Medal for “outstanding CS graduates” @ Saarland Univ.
- Best course award:“Practical Aspects of Security” (TA+guest lecturer)
- Advised 3 MSc + 2 BSc theses; 4 of them on my own

  resulted in 3 conference publications