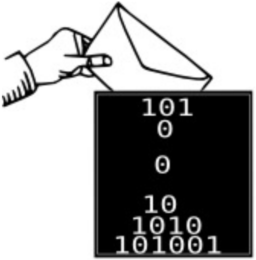
Computer Aided Security: Cryptographic Primitives, Voting protocols, and Wireless Sensor Networks

Pascal Lafourcade

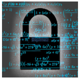
November 6, 2012
Habilitation à Diriger des Recherches

Nowadays Security is Everywhere!



What is cryptography based security?

Cryptography:



- Primitives: RSA, Elgamal, AES, DES, SHA-3 ...
- Protocols: Distributed Algorithms

Properties:

- Secrecy,
- Authentication,
- Privacy ...

Intruders:

- Passive
- Active
- CPA, CCA ...

Designing **secure** cryptographic protocols is **difficult**

Security of Cryptographic Protocols

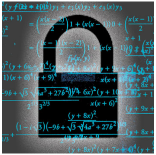
How can we be convinced that a protocols is secure?

- Prove that there is no attack under some assumptions.
  - proving is a difficult task,
  - pencil-and-paper proofs are error-prone.

How can we be convinced that a proof is correct?

Computer-Aided Security.

Formal Verification Approaches



Designer

Attacker

Give a proof

Find a flaw

Back to 1995

     ≥ 17  (Casper/FDR)

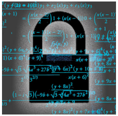
- **Cryptography:** Perfect Encryption hypothesis
- **Property:** Secrecy, Authentication
- **Intruder:**
  - Active
  - Controlling the network
  - Several sessions

Success Story of Symbolic Verification

Tools based on different theories for several properties

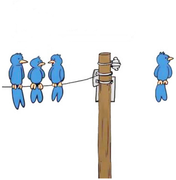
1995	Casper/FRD [Lowe]
2001	Proverif [Blanchet]
2003	Proof of certified email protocol with Proverif [AB]
	OFMC [BMV]
	Hermes [BLP]
	Flaw in Kerberos 5.0 with MSR 3.0 [BCJS]
2004	TA4SP [BHKO]
2005	SATMC [AC]
2006	CL-ATSE [Turuani]
2008	Scyther [Cremers]
	Flaw of Single Sign-On for Google Apps with SAT-MC [ACCCT]
	Proof of TLS using Proverif [BFCZ]
2010	TOOKAN [DDS] using SAT-MC for API
2012	Tamarin [BCM]

Main Contributions:



- Verification techniques for **cryptography**
  - Asymmetric Encryptions
  - Encryption Modes
  - Message Authentication Codes

- **Properties** for E-voting protocols
  - Taxonomy of privacy notions
  - Weighted votes



- **Intruder** models and algorithms for WSN
  - Neighbourhood Discovery Protocols
  - Independent Intruders
  - Routing Algorithms

Related Work

- **CryptoVerif [BP06]:**
  - tool that generates proofs by sequences of games
  - has automatic and manual modes
- **CIL [BDKL10]:** Computational Indistinguishability Logic for proving cryptographic primitives.
- **CertiCrypt [BGZB09] /EasyCrypt [BGHB11]:**
  - Framework for machine-checked cryptographic proofs in Coq
  - Improved by EasyCrypt: generates CertiCrypt proofs from proof sketches

Our Approach

Automatically proving security of cryptographic primitives

1. Defining a language
2. Modeling security properties
3. Building a Hoare Logic for proving the security

Correct but not complete.

- Asymmetric Encryption SchemesAsymmetric Encryption Schemes [CDELL'08,CDELL'10]
- Encryption Modes [GLLS'09]
- Message Authentication Codes (MACs) Submitted [GLL'13]

Examples of Asymmetric Encryptions

- **[BR'93]**: $f(r)||x \oplus G(r)||H(x||r)$
- **[SZ'93]**: $f(r)||G(r) \oplus (x||H(x))$
- **[BR'94] OAEP**: $f(s||r \oplus H(s))$
  where $s = x0^k \oplus G(r)$
- **[Shoup'02] OAEP+**: $f(s||r \oplus H(s))$
  where $s = x \oplus G(r)||H'(r||x)$.
- **[FO'99]**: $\mathcal{E}((x||r); H(x||r))$
  where $\mathcal{E}$ is IND-CPA.

f is a one-way trapdoor permutation, H and G are hash functions and r is a random seed.

Security Property: Indistinguishability

Indis(x; V_1; V_2): seeing V_1 and $f(V_2)$.

Modelling: Generic Encryption Scheme

Grammar for Generic Encryption

$$\begin{aligned}\text{cmd} \quad ::= \quad & x \leftarrow \mathcal{U} \mid x := f(y) \mid x := H(y) \mid \\ & x := y \oplus z \mid x := y || z \mid \text{cmd}; \text{cmd}\end{aligned}$$

A Generic Encryption Scheme

$$\mathcal{E}(\text{in}_e, \text{out}_e) =$$
$$c_1;$$
$$c_2;$$
$$\vdots$$
$$c_n;$$

Bellare & Rogaway'93:

$f(r) || \text{in}_e \oplus G(r) || H(\text{in}_e || r)$

$\mathcal{E}_{BR93}(\text{in}_e, \text{out}_e) =$
$r \stackrel{r}{\leftarrow} \mathcal{U};$
$a := f(r);$
$g := G(r);$
$b := \text{in}_e \oplus g;$
$t := \text{in}_e || r;$
$c := H(t);$
$\text{out}_e := a || b || c$

Only Three Predicates in the ROM

Predicates

$$\begin{array}{lcl} \psi & ::= & \mathsf{H}(G,e) \mid \mathsf{WS}(x;V) \mid \mathsf{Indis}(x;V_1;V_2) \\ \varphi & ::= & \mathsf{true} \mid \psi \mid \varphi \wedge \varphi \end{array}$$

- **H**(G, e): Not-Hashed-Yet

  $\Pr[S \stackrel{r}{\leftarrow} X : S(e) \in S(\mathcal{T}_H).dom]$ is negligible.

- **WS**$(x; V)$: cannot to compute some “hidden” value.

  $\Pr[S \stackrel{r}{\leftarrow} X : A(S) = S(x)]$ is negligible.

- **Indis**$(x; V_1; V_2)$: seeing V_1 and $f(V_2)$.

But more than 30 rules

Verification Technique: Hoare Logic

Set of rules (R_i) : $\{P\}$ cmd $\{Q\}$

$(R_5)\{P_0\}\ c_1\ \{Q_0\}$
$(R_2)\{P_1\}\ c_2\ \{Q_2\}$, where $P_1 \subseteq Q_0$
$\vdots$
$(R_8)\{P_n\}\ c_n\ \{Indis(out_e)\}$?

Examples of rules:

(X2): $\{\mathrm{Indis}(w; V_1, y, z; V_2)\}\ \ x := y \oplus z\ \ \{\mathrm{Indis}(w; V_1, x, y, z; V_2)\}$

(H6): $\{\mathrm{WS}(y; V_1; V_2, y) \wedge \mathrm{H}(H, y)\}\ \ x := H(y)\ \ \{\mathrm{WS}(y; V_1, x; V_2, y)\}$

Example : Bellare & Rogaway's 1993

$r \stackrel{r}{\leftarrow} \{0,1\}^{n_0}$	–	$\mathrm{Indis}(r) \wedge \mathrm{H}(G, r) \wedge \mathrm{H}(H, h\|\|r)$
$a := f(r)$	–	$\mathrm{Indis}(a; \mathrm{Var} - r) \wedge \mathrm{WS}(r; \mathrm{Var} - r) \wedge$
	–	$\mathrm{H}(H, h\|\|r)$
$g := G(r)$	–	$\mathrm{Indis}(a; \mathrm{Var} - r) \wedge \mathrm{Indis}(g; \mathrm{Var} - r) \wedge$
	–	$\mathrm{WS}(r; \mathrm{Var} - r) \wedge \mathrm{H}(H, h\|\|r)$
$e := h \oplus g$	–	$\mathrm{Indis}(a; \mathrm{Var} - r) \wedge \mathrm{Indis}(e) \wedge$
	–	$\wedge \mathrm{WS}(r; \mathrm{Var} - r) \wedge \mathrm{H}(H, h\|\|r)$
$d := h\|\|r$	–	$\mathrm{Indis}(a) \wedge \mathrm{Indis}(e) \wedge$
	–	$\mathrm{WS}(r; \mathrm{Var} - r) \wedge$
	–	$\mathrm{H}(H, d) \wedge \mathrm{WS}(d)$
$c := H(d)$	–	$\mathrm{Indis}(a) \wedge \mathrm{Indis}(e)$
	–	$\wedge \mathrm{Indis}(c)$
$\mathrm{out}_e := a\|\|e\|\|c$	–	$\mathrm{Indis}(oute; \{\mathrm{in}_e, \mathrm{out}_e\})$

Conclusion: Hoare Logics for proving

- **Asymmetric Encryption Schemes**
  - An OCAML prototype of our 30 rules
  - Extensions done for proving IND-CCA using IND-CPA + Plaintext Awareness
  - Exact Security
- **Symmetric Encryption Modes**
  - Counters
  - FOR loops
  - Exact Security
  - An OCAML prototype of our 21 rules
- **Message Authentication Codes (MACs)**
  - Different property: *Unforgeability*
  - Almost-universal Hash function
  - Keep track of possible collisions
  - FOR loops
  - An OCAML prototype of our 44 rules

Revisited [Benaloh'94] Homomorphic Encryption

$\{0\}_{pk_S}$  $\{1\}_{pk_S}$

$$\prod_{i=1}^{n}\{v_i\}_{pk_S} = \{\sum_{i=1}^{n} v_i\}_{pk_S}$$

Result [FLA'11]

- Original Benaloh's scheme is **ambiguous (33%)**:

  $dec(enc(14, pk_S), sk_S) = 14 \mod 15 \text{ or } 14 \mod 5 = 4$

- Proposition of **corrected version**
- Proof using Kristian Gjosteen result

Impact on an election: Result can change (either 14 or 4)

Security Properties of E-Voting Protocols

Fairness

Individual Verifiability

Eligibility

Universal Verifiability

Correctness

Privacy

Receipt-Freeness

Robustness

Coercion-Resistance

Motivation

Existing several models for Privacy, but they

- designed for a specific type of protocol
- often cannot be applied to other protocols

Our Contributions:

- Define **fine-grained** Privacy definitions to **compare** protocols
- Analyze **weighted votes** protocols
- **One coercer is enough**

4 Dimensions for Privacy [DLL'12a, DLL'11]

Modeling in Applied π-Calculus

1. **Communication btwn the attacker & the targeted voter**

   [DKR09]  

   Vote-Privacy (VP) Receipt-Freeness (RF) Coercion-Resistance (CR)

2. **Intruder is controlling another voter**

   Outsider (O)  Insider (I)

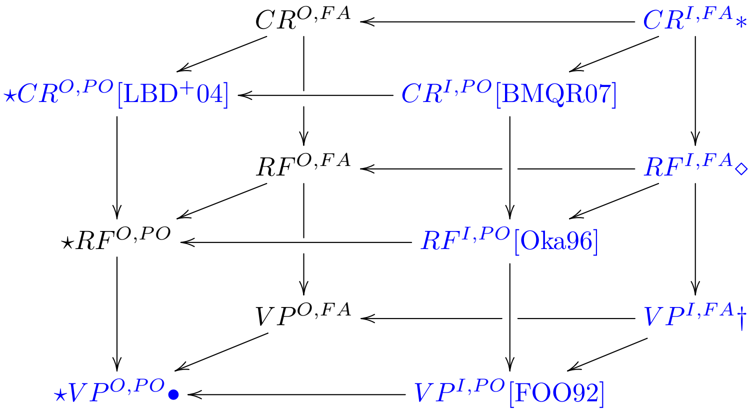
3. **Secure against Forced-Abstention**: (FA) or not (PO)

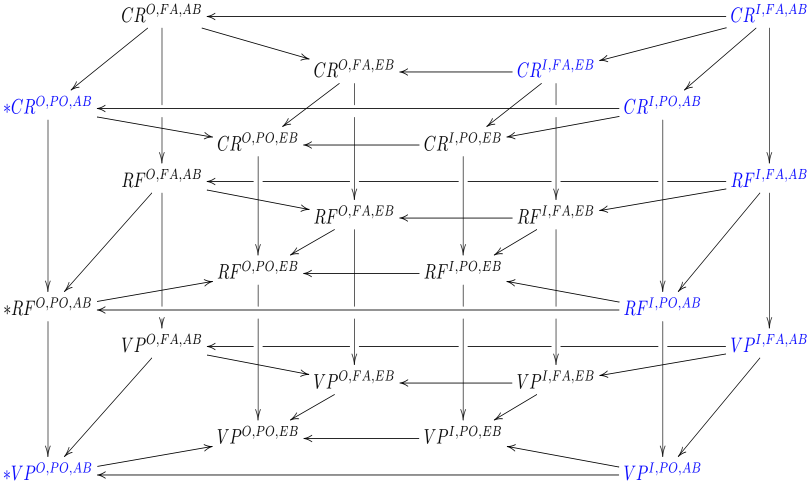
4. **Honest voters behavior**:

   $\exists$  $\forall$

Relations without ∃ and ∀



All relations among the notions

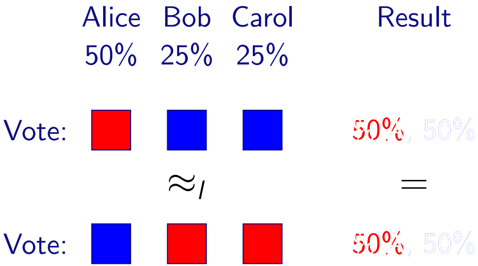


Privacy for Weighted Votes [DLL'12b]

	Alice 66%	Bob 34%	Result
Vote:	red	blue	66%, 34%
	$\not\approx_l$		$\neq$
Vote:	blue	red	34%, 66%

Privacy for Weighted Votes [DLL'12b]

Still: Some privacy is possible!

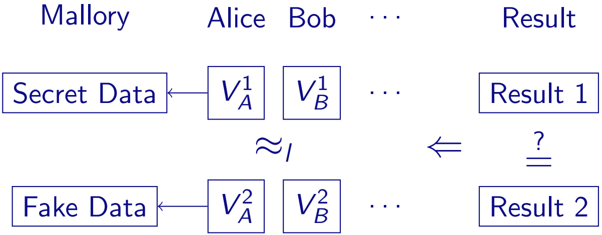


Definition of Vote-Privacy (VP) for weighted votes

Idea: Two instances with the same result should be bi-similar

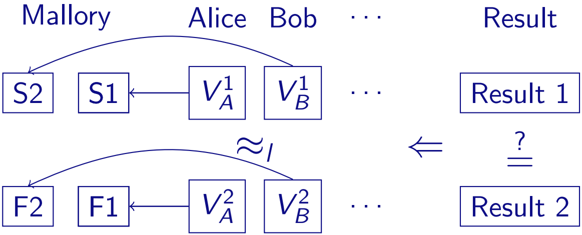
	Alice	Bob	⋯		Result
Vote 1:	V_A^1	V_B^1	⋯		Result 1
	$\approx_l$			$\Leftarrow$	$\stackrel{?}{=}$
Vote 2:	V_A^2	V_A^2	⋯		Result 2

Single-Voter Receipt Freeness (SRF)



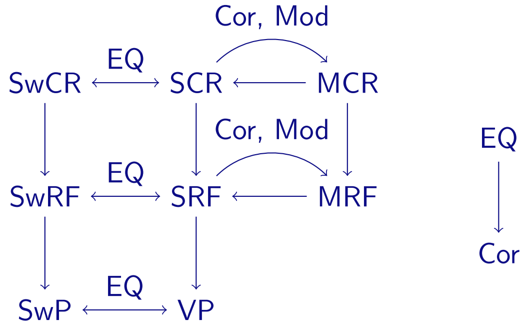
If a protocol respects (EQ), then (SRF) and (SwRF) are equivalent.

Multi-Voter Receipt Freeness (MRF)

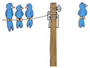


(MRF) implies (SRF) and (MCR) implies (SCR).

One Coerced Voter is enough!



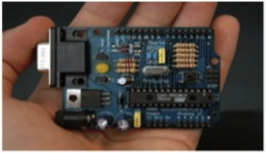
Unique decomposition of processes in the applied π-calculus.



Challenges in WSNs

Nodes

- Broadcast communication
- Low computation power
- Battery



- **Cryptography:** Lightweight, energy- and resource-aware ...
- **Properties:** (k)-neighborhood, routing ...
- **Intruders:** Black-hole, wormhole, Byzantine, independent ...

Our Contributions

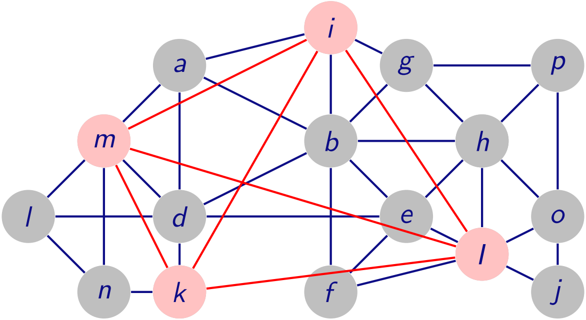
- (k)-Neighbourhood Verification [JL'12]
- Independent IntrudersIndependent Intruders [KL'12]
- Analysis of non-backtracking random walk [ADGL'12]
- Resilient routing algorithm [ADJL] Resilient routing algorithm [ADJL]

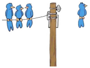
Usual Intruders

Dolev-Yao's Intruder [83]

Intruder Model in WSNs

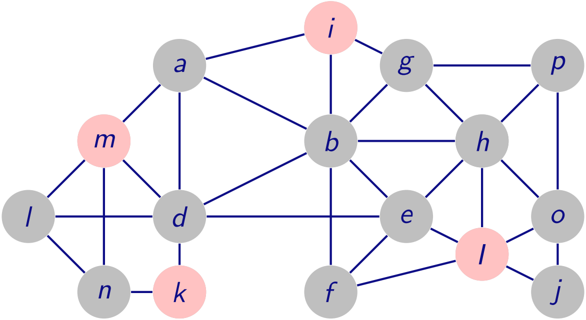
Several intruders with sharing [ACD12]





Independent Intruder Model

Independent intruders without sharing



Usual Constraints System

$$\begin{array}{rcl} T_1 & \Vdash & u_1 \\ T_2 & \Vdash & u_2 \\ & \vdots & \\ T_n & \Vdash & u_n \end{array}$$

- **Intruder knowledge monotonicity**: $T_1 \subseteq \cdots \subseteq T_n$.
- **Variable origination**: if x occurs in $vars(T_i)$ for certain T_i then there exists $k < i$ such that $x \in vars(u_k)$.

Partially Well-Formed Constraint System

Partially well-formed constraints system

$$\mathcal{C} = T_1^l \Vdash u_1 \wedge \cdots \wedge T_n^q \Vdash u_n$$

- Global Origination.
- Partial monotonicity:
  $T_k^j \subseteq T_i^j$ for every $j \in \{1, 2, \ldots, m\}$ such that $k < i$.

Quasi-Solved Form

$$
\begin{array}{lll}
R_{ax}: & \mathcal{C} \wedge T_i^j \Vdash u_i \rightsquigarrow \mathcal{C} & \text{if } T_i^j \cup \{x \mid T_k^j \Vdash x \in \mathcal{C}, k < i\} \vdash u_i \\
R_{unif}: & \mathcal{C} \rightsquigarrow_\sigma \mathcal{C}\sigma & \sigma = mgu(t_1, t_2),\ t_1, t_2 \in st(\mathcal{C}) \\
R'_{unif}: & \mathcal{C} \wedge T_i^j \Vdash u_i \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T_i^j\sigma \Vdash u_i\sigma & \sigma = mgu(t, f(t_1, t_2)), f \in \{\langle -, - \rangle, - :: -\}, \\
 & & t \in vars(u_i), t_1, t_2 \in st(T_k^l),\ \text{where } k \leq i \\
R_f: & \mathcal{C} \wedge T^j \Vdash f(u, v) \rightsquigarrow \mathcal{C} \wedge T^j \Vdash u \wedge T^j \Vdash v & \text{if } f \in \{senc, aenc, \langle -, - \rangle, - :: -, hmac, sig\} \\
R_{fail}: & \mathcal{C} \wedge T_i^j \Vdash u_i \rightsquigarrow \bot & \text{if } T_i^j = \emptyset, \text{or } vars(T_i^j \cup \{u_i\}) = \emptyset \\
 & & \text{and } T_i^j \nvdash u_i
\end{array}
$$

Soundness, completeness and termination.

Example of Quasi-Solved Form:

$$
\begin{array}{lll}
T_1^1 & = & \{a, b\} \Vdash x \\
T_2^2 & = & \{x\} \Vdash a \\
T_3^3 & = & \{x\} \Vdash b
\end{array}
$$

Procedure for finding a solution to a quasi-solved form.

Resilient Routing Algorithms

Even with "**perturbation**" a **resilient** protocol should work "**well**"

- Perturbation: abnormal behavior, node destruction, battery ...
- Well: Hitting time, average delivery rate...

Existing protocols

Probabilistic *vs* **Deterministic**
Random walk GBR, GFG

Our Goal: Design an efficient resilient routing algorithm using a reputation mechanism

Our Resilient Algorithm: TLCNS [ADJL]

Shared symmetric key K_{OS} between the sink and all nodes O.

- Each node O sends: $\{Data, N_O\}_{K_{OS}}, H(N_O), O, F$
- Sink S acknowledges: N_O, O

3 lists for each node:

- $M_{ack} = [(H(N_O), A), (H(N_B), C)]$: List of hashed nonces and sender identity.
- $M_{Queue} = [(N_O^1, A), (N_O^2, B)]$: List of messages sent
- $L_{Routing} = [A, B, C]$: List of "preferred" first hops (FIFO)

Why does it work?

- Each node prefers preferred next hop
- All neighbours are possible

Scenario for testing the Resilience

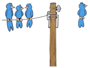
- Simulation using SINALGO
- $|L_{Routing}| = 10$, $|M_{Queue}| = 5$ and $|M_{ack}| = 3$
- 200 nodes, 1 sink

Intruders:

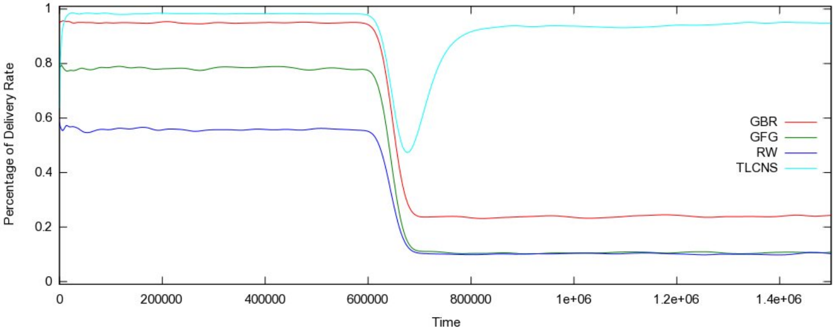
- Black Holes: Node not forwarding any message
- Worm Holes: False link in the topology

Scenario in 2 phases:

- Static: 10 Black holes + 10 Wormholes
- Dynamic: 20 Black holes
  (Wormholes → Black Holes)

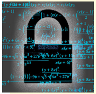


Results


1
0.8
0.6
0.4
0.2
0
Percentage of Delivery Rate
0
200000
400000
600000
800000
1e+06
1.2e+06
1.4e+06
Time
GBR
GFG
RW
TLCNS

Summary

Automatic proofs of programs (Hoare Logic)

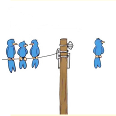


- Generic Asymmetric Encryption [CDELL'08, CDELL'10]
- Generic Encryption Mode: counter + For loop [GLL'09]
- Generic MAC: Double execution + For loop [GLL'13]

Cryptography & Process Algebra (Applied π-Calculus)

- Revisited Benaloh's encryption scheme [FLA'11]
- Privacy notions [DLL'12a, DLL'11]
- Weighted votes [DLL'12b]

Constraints Solving & Randomized Algorithms



- Neighbourhood Discovery Verification [JL'12]
- Independent Intruders [KL'12]
- Design of routing algorithms [AGDL'12, ADLP'11]

Future Work

- **Computer-Aided Cryptography**:
  - Hoare Logic for other primitives: Pairing, E-Stream ...
  - How to prove Benaloh' scheme?
  - Using verification for the synthesis of new schemes
- **Properties**:
  - E-auctions: Non cancellation, Non repudiation, Privacy ...
  - Non-functional properties for WSNs: energy consumption.
- **Intruder Model**:
  - With a battery
  - Mobility

Thank you for your attention.

Questions ?