INternet SECURITY and Cryptographic Protocols

G. Sivakumar

Computer Science Department

Indian Institute of Technology, Bombay

Mumbai 400076, India

siva@iitb.ac.in

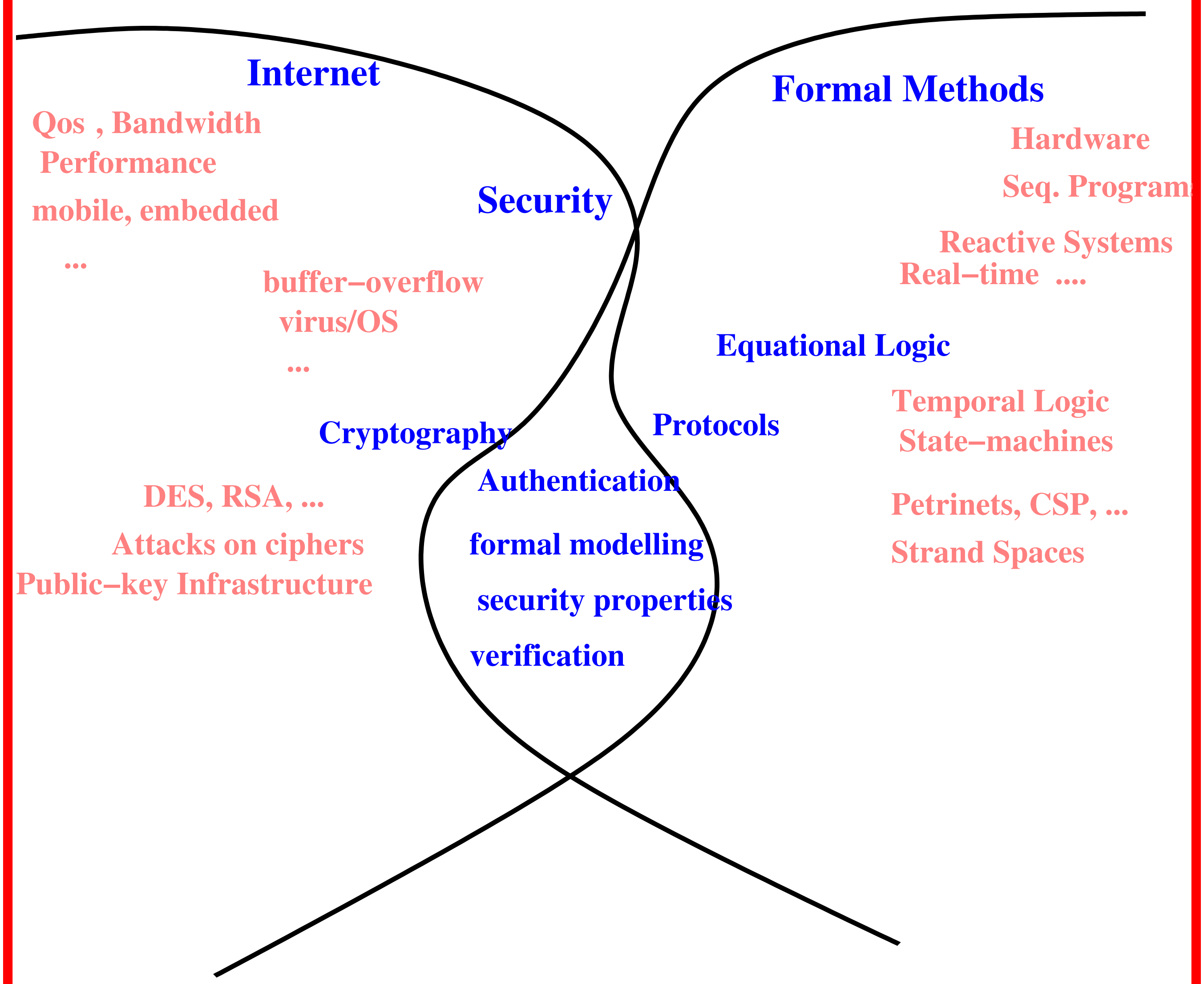
Outline of Talk

- Internet Security Overview [15 mins]
  - Requirements, Vulnerabilities, Threats
  - Cryptography (*sine qua non!*)
- Security Protocols [15 mins]
  - Session Keys (Diffie-Hellman)
  - Authentication (Needham-Schroeder)
- Formal Methods for Security Protocols [10 mins]
  - Specification of Protocol and Properties
  - Survey of Approaches and Tools
- A Taste of Rewriting [15 mins]
  - Equational Logic
  - Inductive Proofs

  - Modelling Protocols

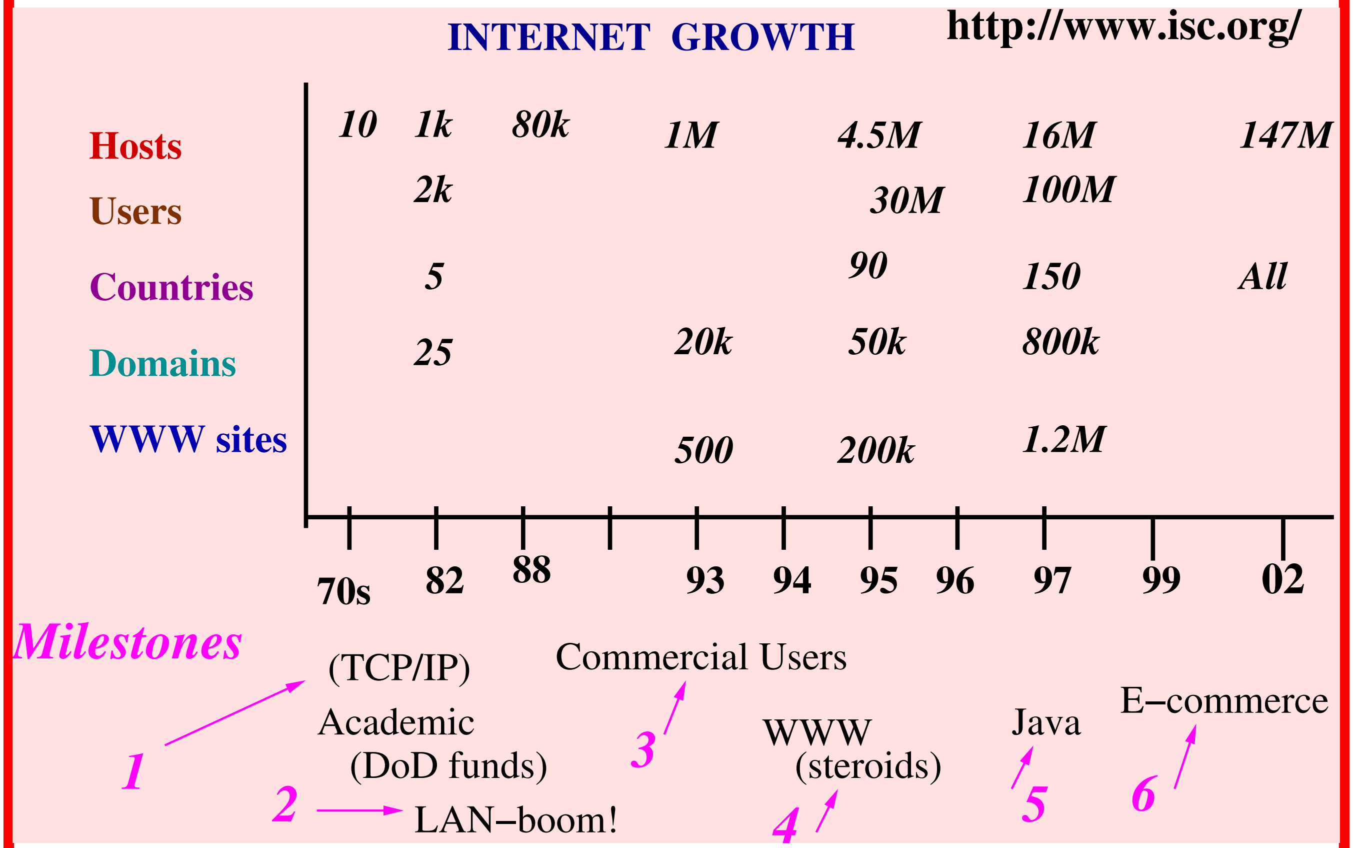
- Emerging Issues and Q&A

Scope of Talk



Any topic can be *reduced* to one's area of expertise!

Internet’s Growth and Charter



Information **AnyTime, AnyWhere, AnyForm, AnyDevice, ...**

WebTone like DialTone

Security Concerns on Internet

Sample problems

- Highly contagious viruses
- Defacing web pages
- Credit card number theft
- On-line scams
- Intellectual property theft
- Wiping out data
- Denial of service
- Spam E-mails
- Reading private files
- Surveillance

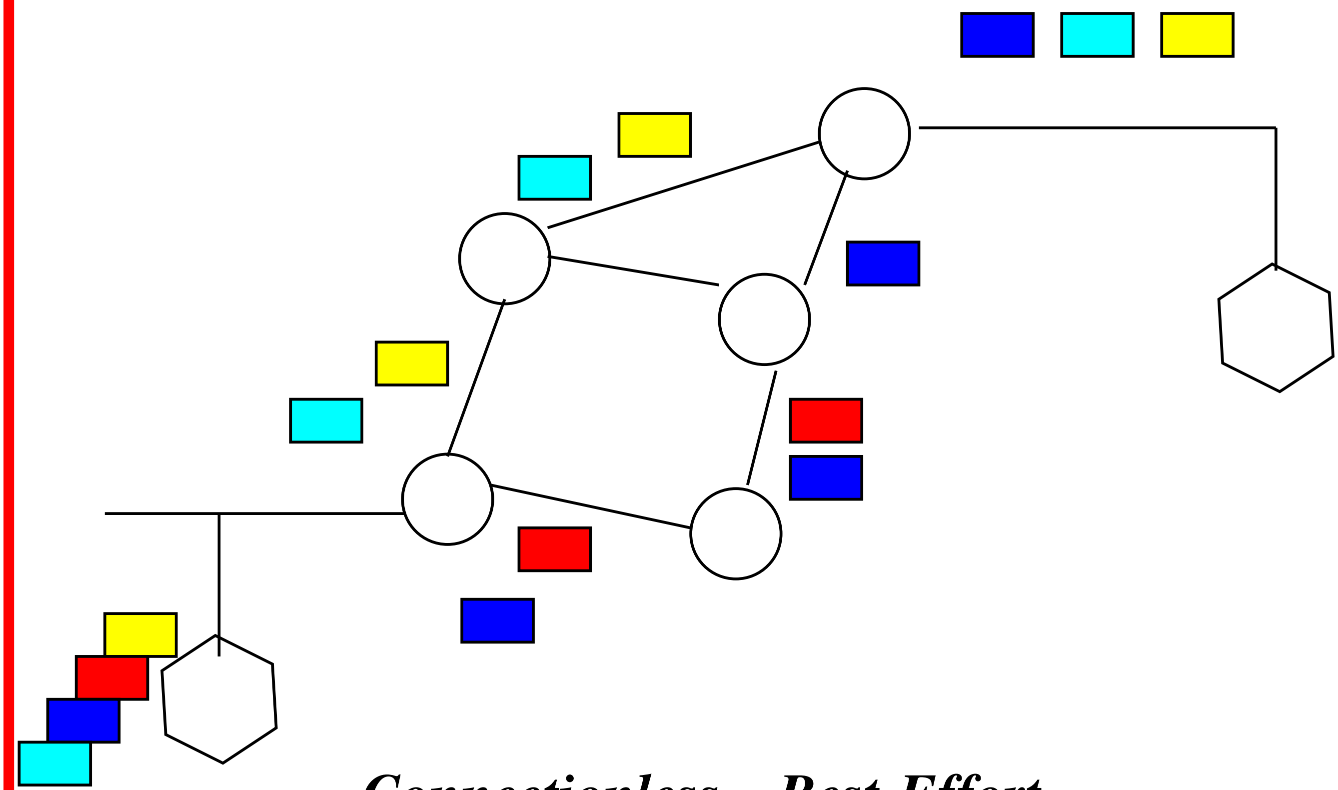
Who are the attackers?

- Unintended blunders
- Hackers driven by technical challenge
- Disgruntled employees or customers
- Petty criminals
- Organized crime
- Organized terror groups
- Foreign espionage agents
- Information warfare

Why is Internet Vulnerable?

Quick overview of how Internet works.

Datagram Routing through Internet

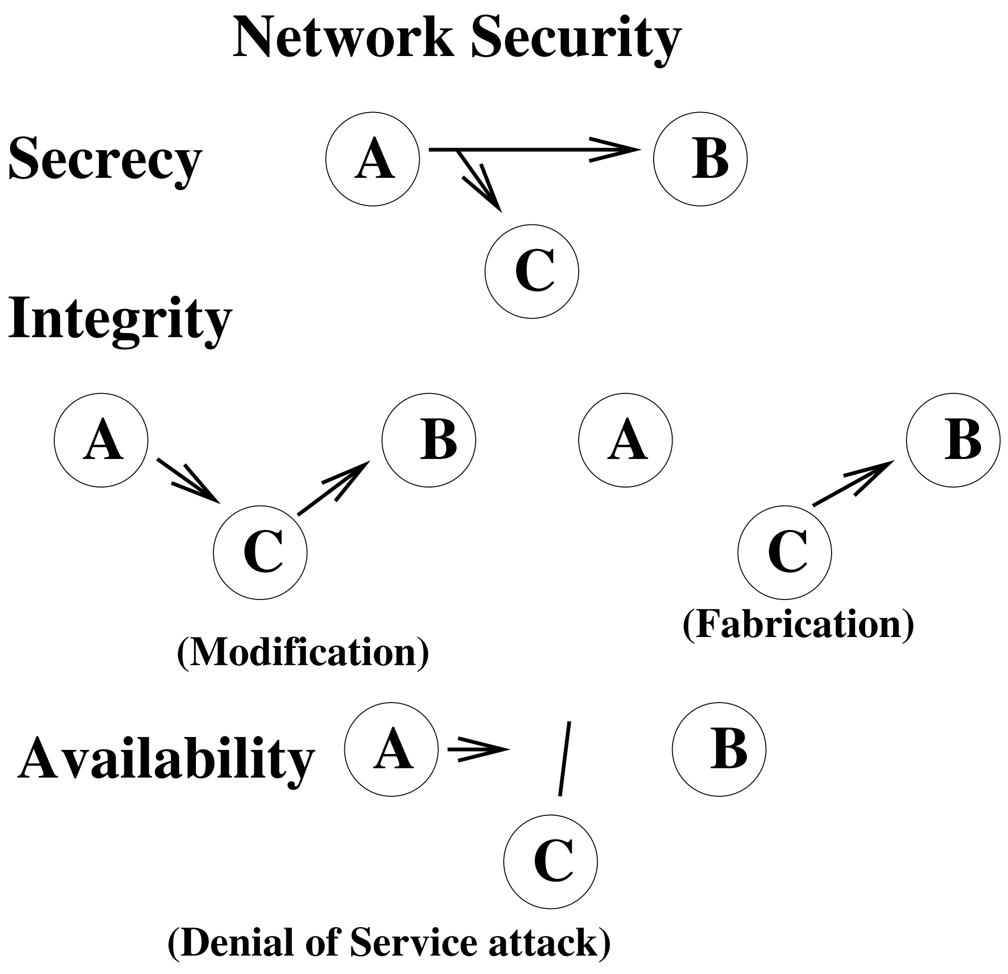


Connectionless   Best-Effort

Analogy with PostCards

Vulnerabilities

- **Application Security**
  - Buggy code
  - Buffer Overflows
- **Host Security**
  - Server side (multi-user/application)
  - Client side (virus)
- **Transmission Security**



Security Requirements

Informal statements (formal is much harder)

- Confidentiality Protection from disclosure to unauthorized persons
- Integrity Assurance that information has not been modified unauthorizedly.
- Authentication Assurance of identity of originator of information.
- Non-Repudiation Originator cannot deny sending the message.
- Availability Not able to use system or communicate when desired.
- Anonymity/Pseudonomity For applications like voting, instructor evaluation.
- Traffic Analysis Should not even know who is communicating with whom. Why?

And all this with postcards (IP datagrams)!

Security Mechanisms

- **System Security:** “Nothing bad happens to my computers and equipment”

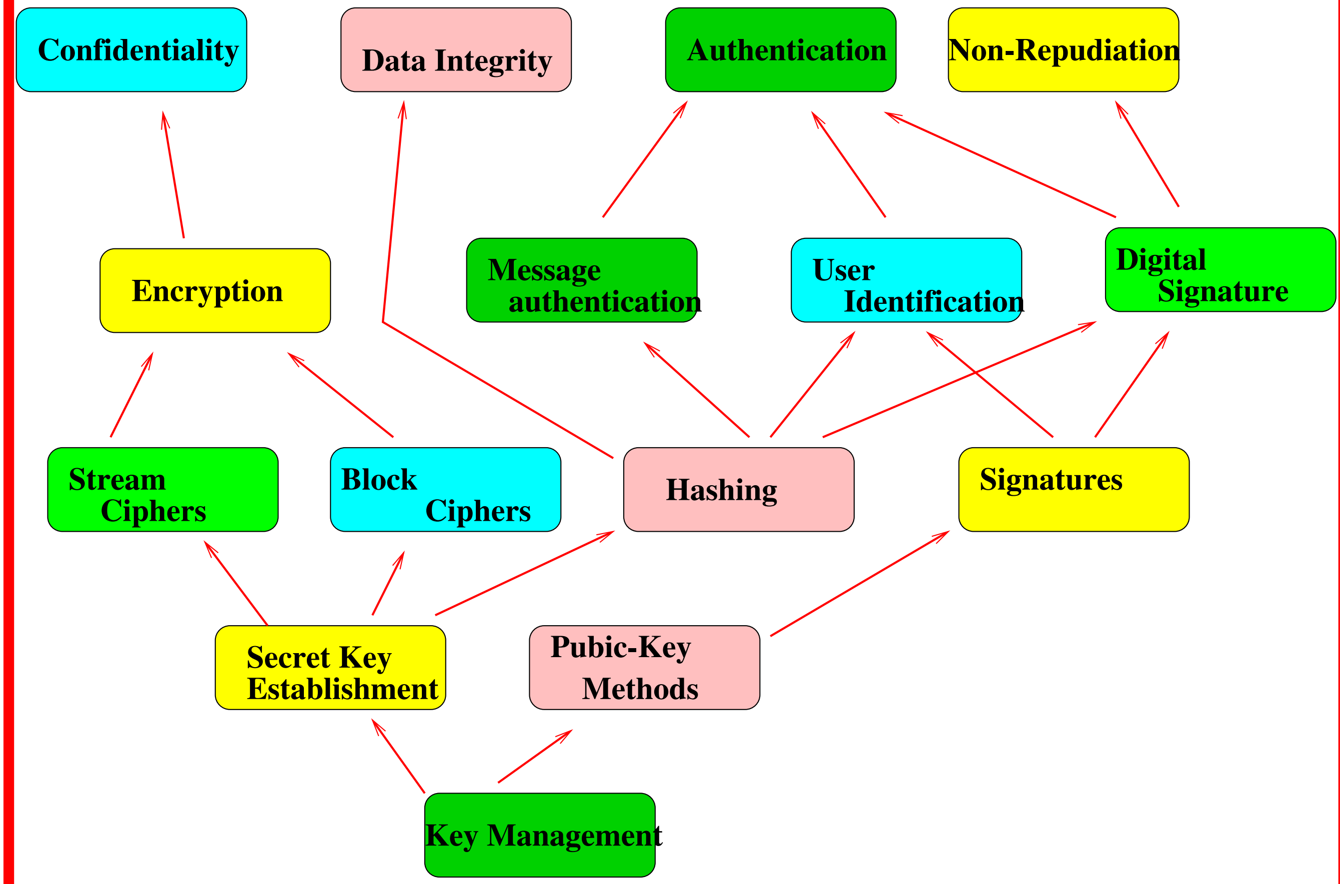
  virus, trojan-horse, logic/time-bombs, ...

- **Network Security:**
  - **Authentication Mechanisms** “you are who you say you are”
  - **Access Control** Firewalls, Proxies “who can do what”

- **Data Security:** “for your eyes only”
  - Encryption, Digests, Signatures, ...

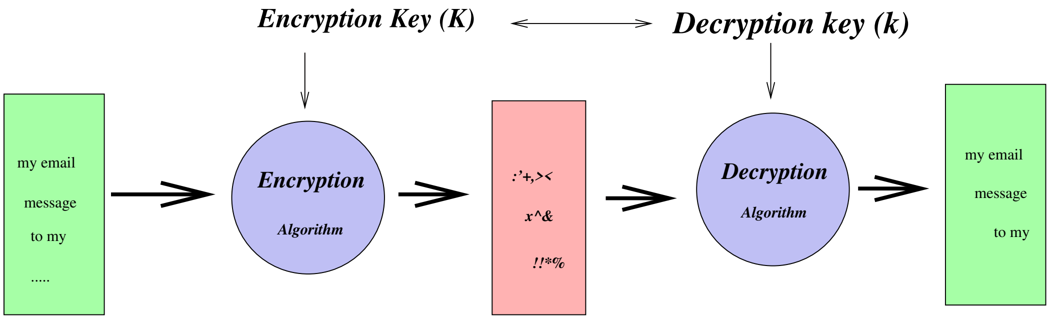
Cryptography and Data Security

- *sine qua non* [without this nothing :-]



Symmetric/Private-Key Algorithms

SYMMETRIC/SHARED KEY Encryption



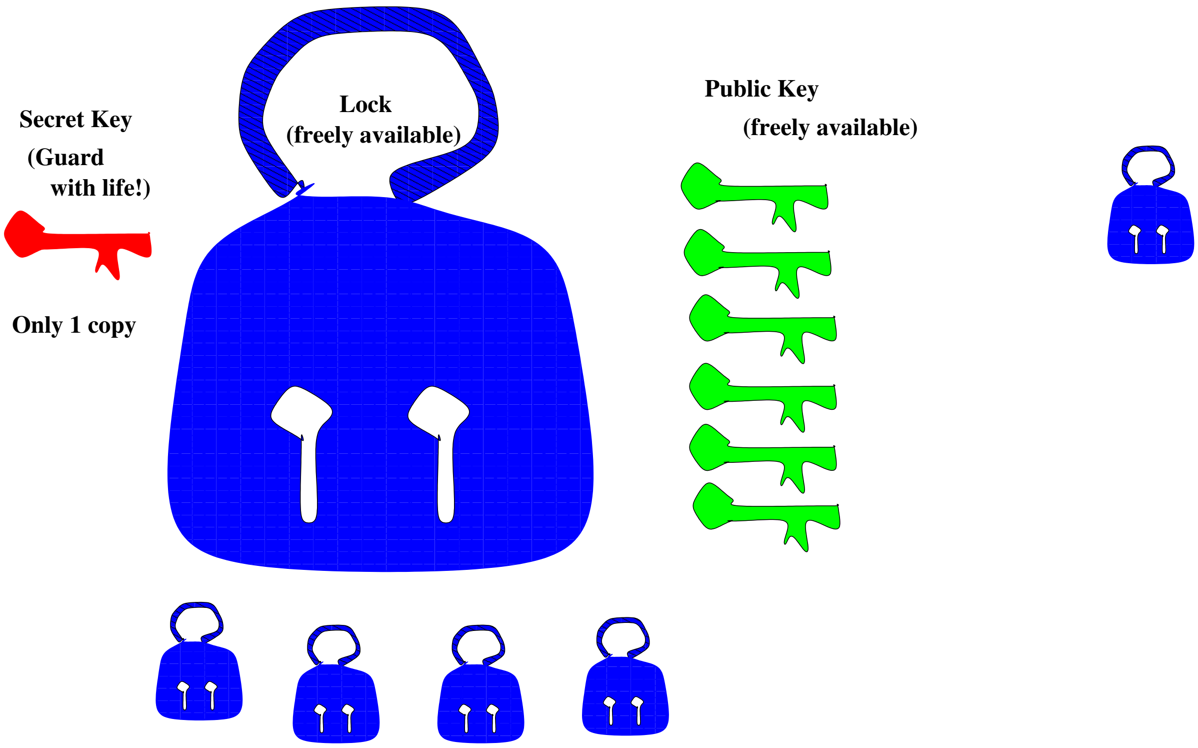
Advantages

* ***Fast***
* ***Special Hardware***
* ***Built-in Authentication***
  Message Integrity

Disadvantages

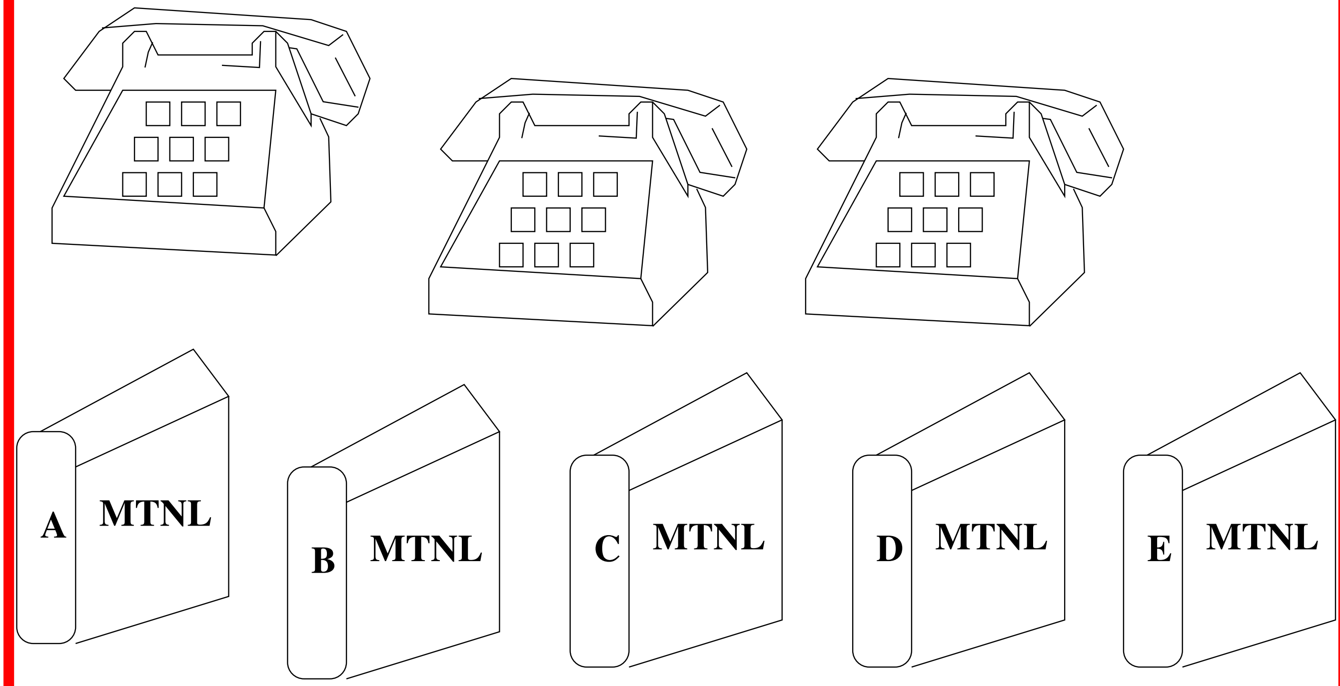
* ***How to Exchange Keys?***
* ***Puzzle in Tannenbaum***

Asymmetric/Public-Key Algorithms



- Keys are duals (lock with one, unlock with other)
- Cannot infer one from other easily
- How to encrypt? How to sign?

One way Functions



One way functions are boring

4034572 **(Ortega)**

5767799 **(Narayan)**

TRAPDOOR 1–way Functions (CD)

Mathematical Equivalents

- Factoring large numbers (product of 2 large primes)
- Discrete Logarithms

One-way Functions

- Computing *f(x)* = *y* is easy.
- Eg. $y = 4^x \; mod \; 13$ (If x is 3, y is —?)

n	4^n mod 13	10^n mod 13
1	4	10
2	3	9
3	12	12
4	9	3
5	10	4
6	1	1
7	4	10
$\vdots$	$\vdots$	$\vdots$

- Note: need not work with numbers bigger than 13 at all!
- But given y = 11, finding suitable x is not easy!
- Can do by brute-force (try all possibilities!)
- No method that is **much** better known yet!

Motivation for Session keys

Combine Symmetric (fast) and Asymmetric (very slow) Methods using session (ephemeral) keys for the following additional reasons.

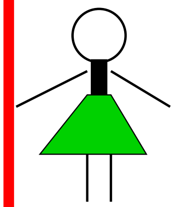
- Limit available cipher text (under a fixed key) for cryptanalytic attack;
- Limit exposure with respect to both time period and quantity of data, in the event of (session) key compromise;
- Avoid long-term storage of a large number of distinct secret keys (in the case where one terminal communicates with a large number of others), by creating keys only when actually required;
- Create independence across communications sessions or applications. No replay attacks.

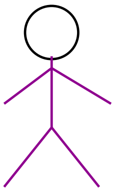
How to establish session keys over insecure medium where adversary is listening to everything?

Can be done even without any public key! Randomization to rescue (like in CSMA/CD of Ethernet).

Diffie-Hellman Key Establishment Protocol

Alice $\longleftarrow$ P, g $\longrightarrow$ Bob





P = 13, g = 4

Choose N_a 5

Choose N_b 8

Send $g^{N_a} \bmod P = m_a$

Send $g^{N_b} \bmod P = m_b$

Compute $4^8 \bmod 13 = 3$

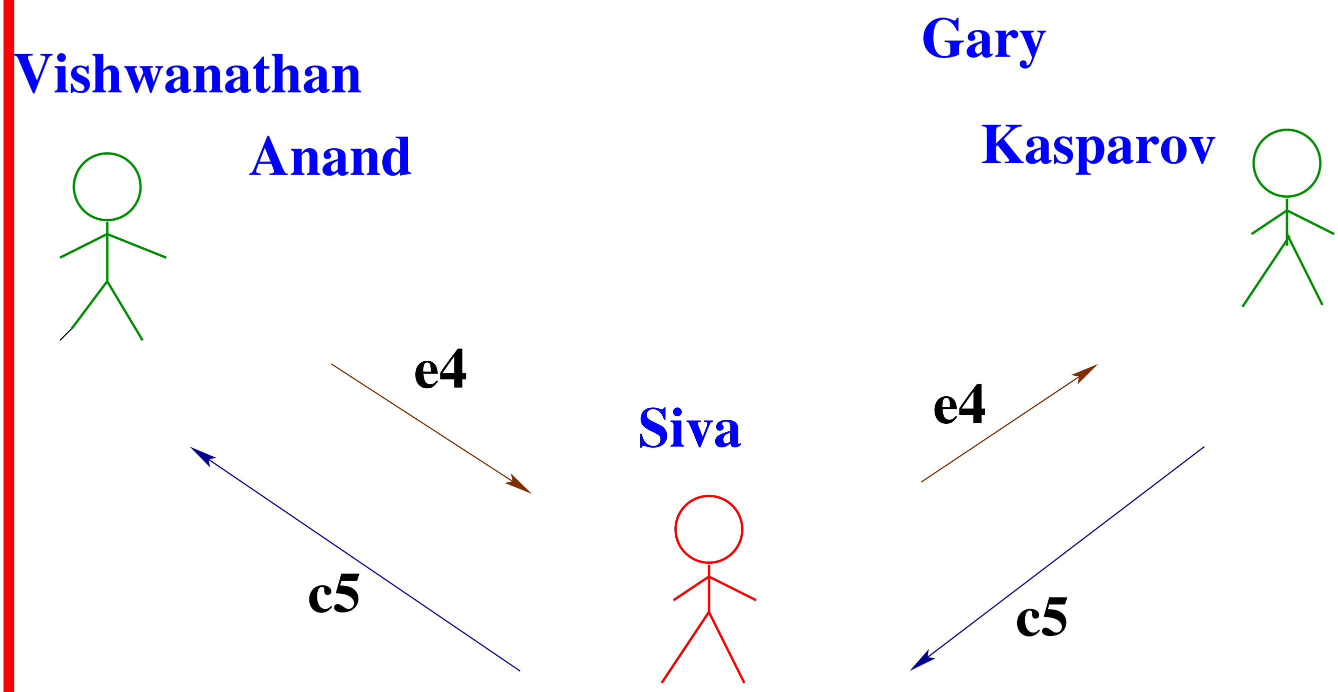
Compute $4^5 \bmod 13 = 10$

$K_{ab} = m_b^{N_a} \bmod P$

$m_a^{N_b} \bmod P = K_{ba}$

$3^5 \bmod 13 = 9 = 10^8 \bmod 13$

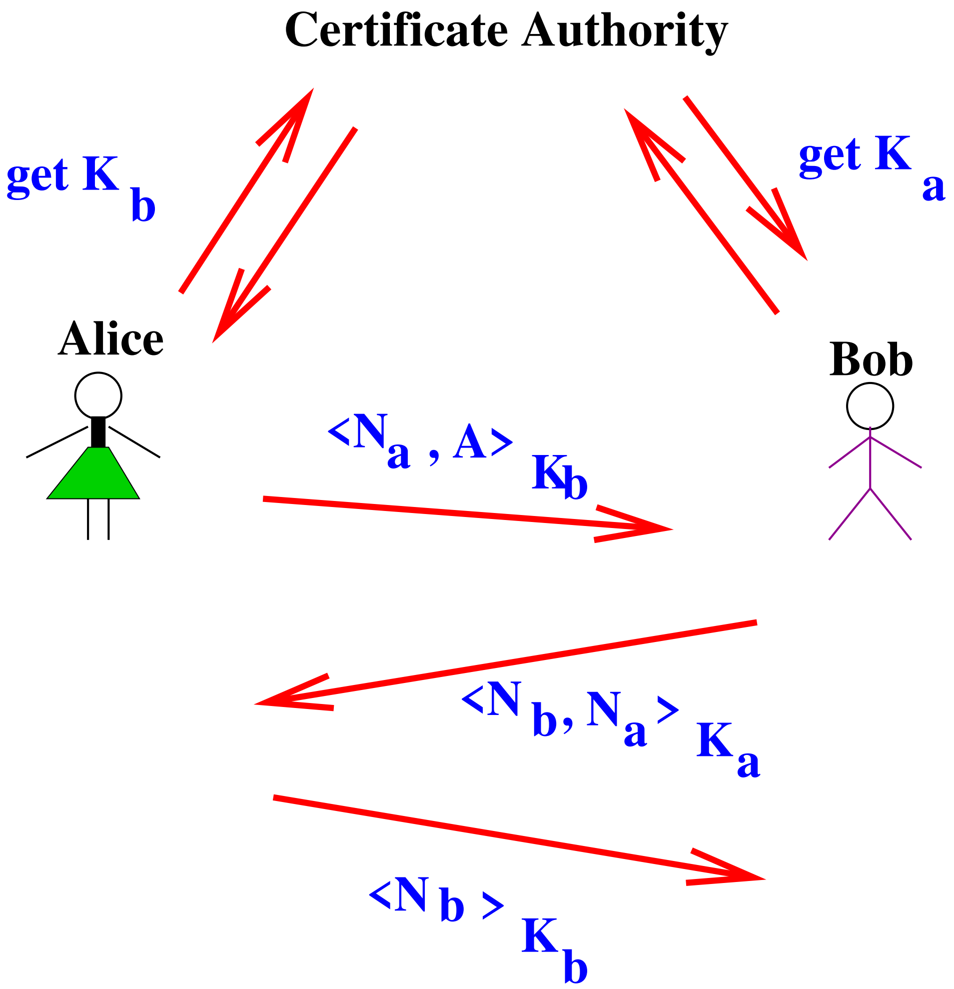
Man-in-the-middle attack



- Authentication was missing!
- Can be solved if Kasparov and Anand know each other's public key (Needham-Schroeder).
- Yes, but different attack possible.

Needham-Schroeder Protocol

Needham–Schroeder Authenticated Key Exchange

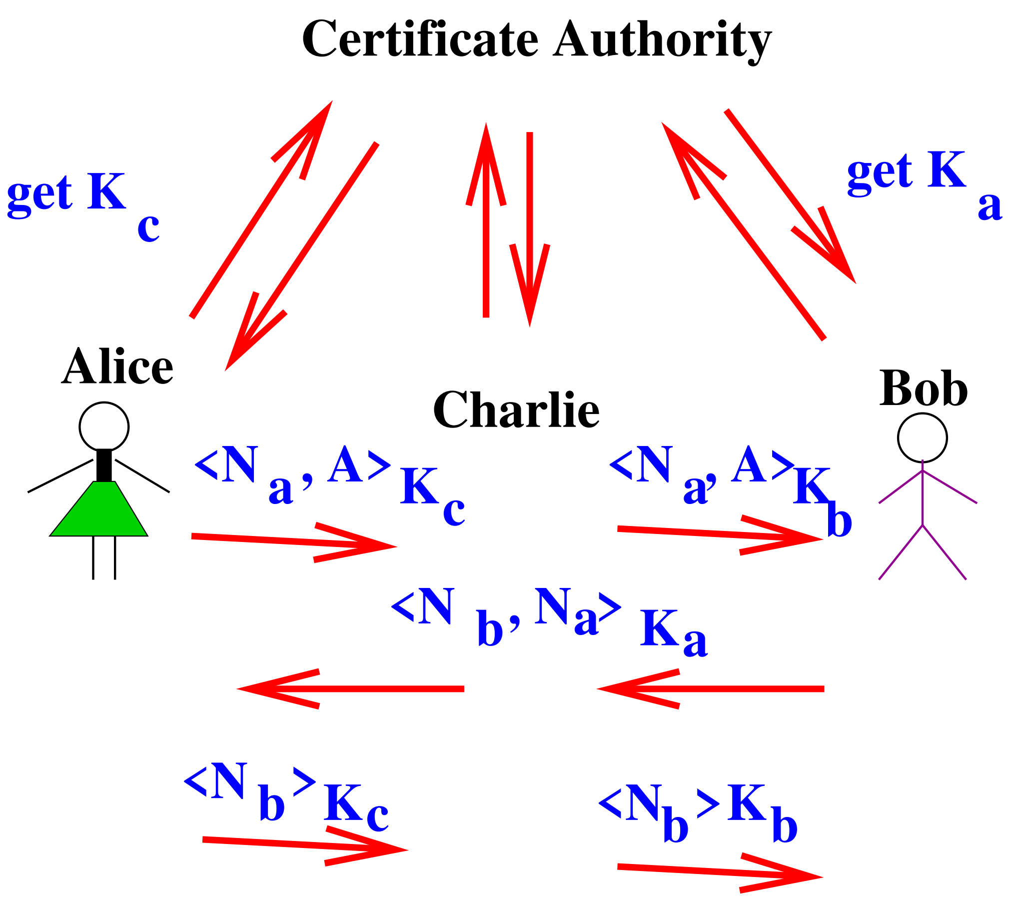


Alice and Bob have authenticated each other?

They have also established a session key $f(N_a, N_b)$

Things looked rosy for 10 years. Then attack discovered.

Attack by Lowe (1995)



Alice (correctly) thinks she is talking to Charlie

Bob has been fooled into thinking he is talking to Alice!

Why Are Security Protocols Often Wrong?

They are *trivial* programs built from simple primitives, BUT, they are complicated by

- concurrency
- a hostile environment
  - a bad user controls the network
  - Concern: active attacks masquerading, replay, man-in-middle, etc.
- vague specifications
  - we have to guess what is wanted
- Ill-defined concepts

Protocol flaws rather than cryptosystem weaknesses

Need for Formal Methods

Countermeasure: formal design and analysis

- Formal Modelling and Specification of Protocol

  Abstract encryption model, formal specification

- Specification of Required Properties

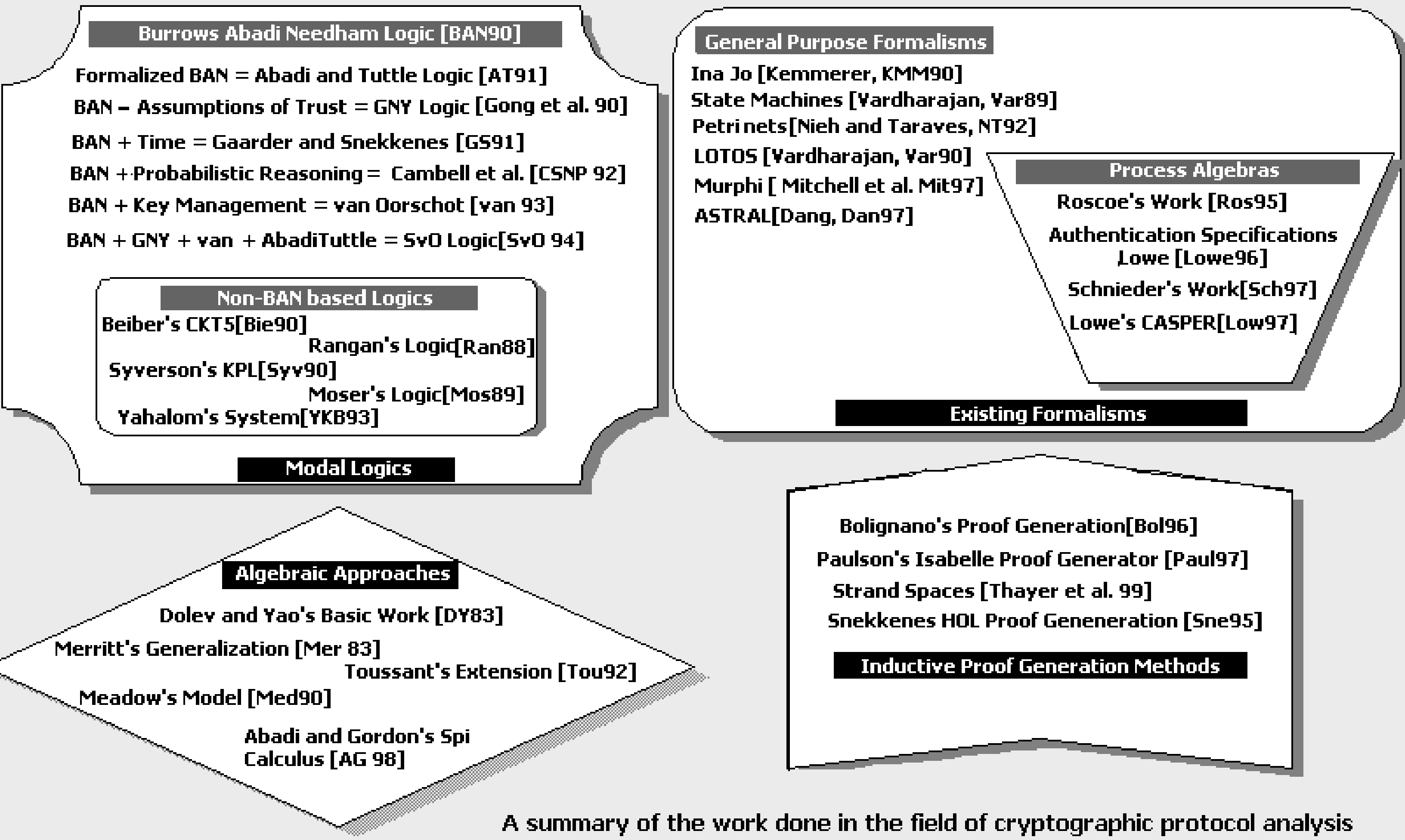
- Verification of Properties

  Inductive proofs, state-space search, authentication logics

- Generation of Counter-Example

  Analysis can find flaws, suggest improvements, prove conditional correctness

Formal Approaches Overview



A summary of the work done in the field of cryptographic protocol analysis

Why so many approaches?

When all you have is a hammer, everything looks like a nail!

Specification of Protocol

Common Authentication Protocol Specification Language

http://www.csl.sri.com/users/millen/capsl/

- High-level message-list based language with abstract encryption operators

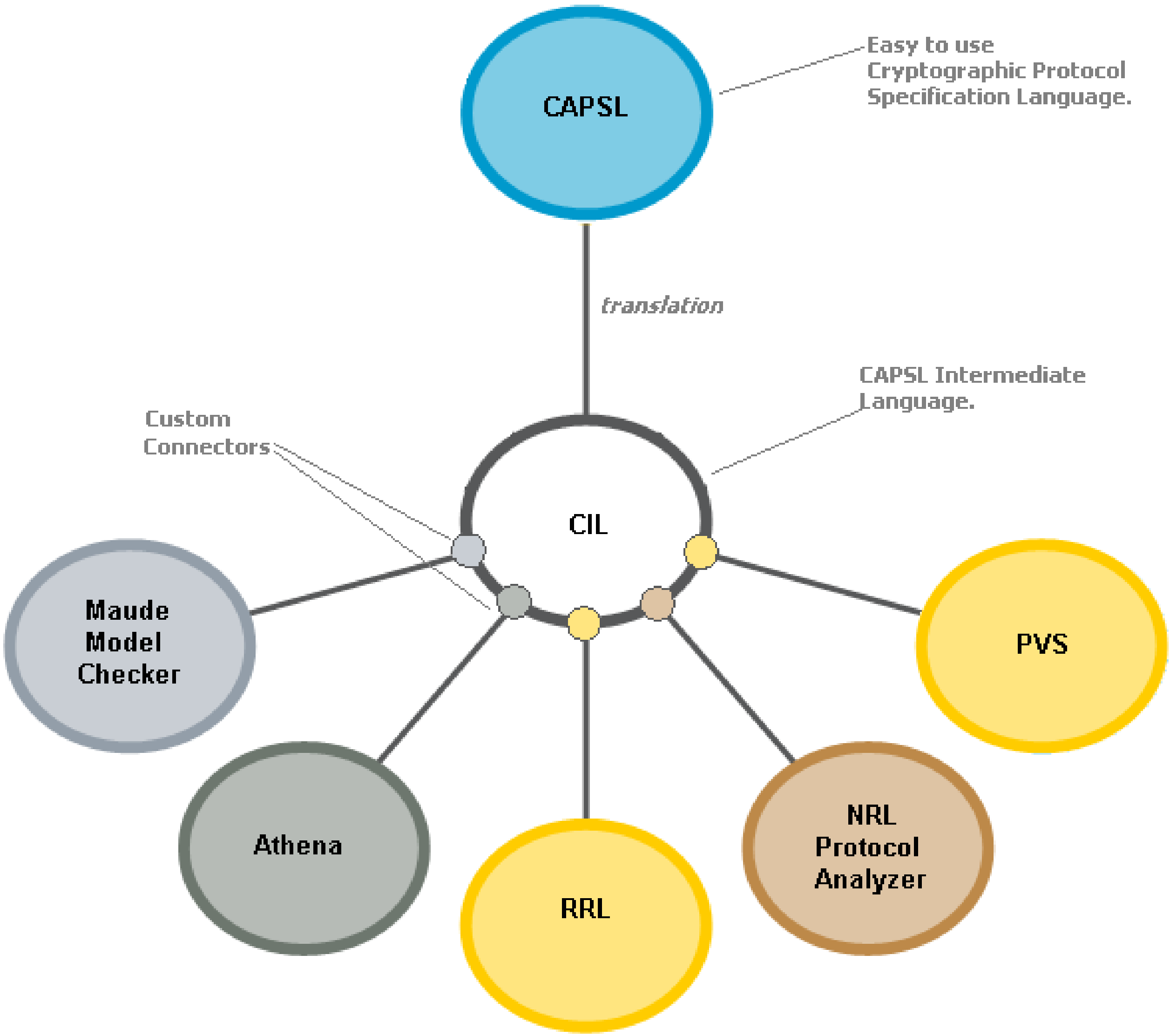
```
A -> B: {A}K
```

- Declarations:
  - strong typing and abstract data type extensions
  - initialization, named expressions
  - security goals
- Actions between messages: tests, assignments

CAPSL Example

```
PROTOCOL Short;
IMPORTS ClientServer;
VARIABLES
A,S: PKUser;
N: Nonce, FRESH, CRYPTO;
ASSUMPTIONS
HOLDS A: S;
MESSAGES
A -> S: A,{A,N}SK(A);
S -> A: {S,N}PK(A);
GOALS
SECRET N: A, S;
END;
```

Emerging Picture



The CAPSL Framework

Tools for Security Analysis and Verification

Tool Name	Theoretical Base	Little Human Effort	Automatic Attack Discovery	Precise Semantics	Unbounded Principal Size.
BAN	Belief Logics				☑
FDR	CSP	☑	☑	☑	
Isabelle	Proof Generation			☑	☑
Athena	Strand Spaces	☑	☑	☑	☑
Interrogator	Algebraic Systems			☑	☑
Cryptyc	Spi Calculus	☑		☑	☑
NRL Protocol Analyzer	Term Rewriting Systems			☑	☑

Classical Logic based Approaches

- First-order logic + Equality (can be semi-decided by resolution)
- RRL (Rewrite Rule Lab)
  - Conditional Equations/Rules ($l \rightarrow r \;\; if \;\; c$)
  - Successful Uses
    * Algebraic Theories and Data Types
    * Geometry Theorem Proving
    * Hardware Verification (family of adders, multipliers, SRT division)
    * Protocol modeling
- PVS (Prototype Verification System)
  - Higher-order Logic (pass functions as arguments, quantify over functions)
  - Strong typing (subtypes and dependent types)
  - Very extensively used
- STeP (Standford Temporal Prover)
  - SPL and Fair Transition Systems
  - Concurrent and Reactive Systems

Equational Logic

Standard example (Group Theory)

$$\begin{array}{rcl} 0 + x & = & x \\ -(x) + x & = & 0 \\ (x + y) + z & = & x + (y + z) \end{array}$$

Equational logic ("replace equals by equals")

$$0 \; = \; 0 + x \; = \; (-0 + 0) + x \; = \; -0 + (0 + x) \; = \; -0 + x$$

Validity: Is $-0 = 0$? Is $-(-(x)) = x$? (for all x)?

Satisfiability: Is there x such that $x + x = 0$?

Formal definition of **term**, **substitution**, **matching**, **unification** skipped!

We can give an "efficient" **decision** procedure for validity and a **semi-decision** procedure for satisfiability if we can find a **convergent (canonical) rewrite system** equivalent to the above equations.

Rewrite Systems

A **rule** is an "oriented" equation (one-way replacement).

Numbers are built from **constructors** (0, s) and some functions $+, *, fact, gcd$ are **defined** as follows.

$$
\begin{array}{rcl}
0 + x & \rightarrow & x \\
s(x) + y & \rightarrow & s(x + y) \\
0 * x & \rightarrow & 0 \\
s(x) * y & \rightarrow & y + (x * y) \\
fact(0) & \rightarrow & s(0) \\
fact(s(x)) & \rightarrow & s(x) * fact(x) \\
gcd(0, x) & \rightarrow & x \\
gcd(x, x + y) & \rightarrow & gcd(x, y)
\end{array}
$$

Derivations and Normal Forms

$$s(0) + (0 * s(0)) \rightarrow s(0) + 0 \rightarrow s(0 + 0) \rightarrow s(0)$$

Is definition of **gcd** complete?

- Can we **simplify** $gcd(s^2(0), s^4(0))$?
- We need matching modulo $+$
- How about $gcd(s^4(0), s^2(0))$?
- We need **commutativity** of gcd.

Equational Programming

(First-order) Functional Programming

- Evaluate $fact(s^2(0) * s^3(0))$
- **Matching** is the parameter-passing mechanism for applying rules.
- No backtracking if definition is **confluent**.

Logic Programing

- Solve $x * y = s^4(0)$
- Enumerate **all** answers: $\{x \mapsto s(0), y \mapsto s^4(0)\}$, $\{x \mapsto s^2(0), y \mapsto s^2(0)\}$, ...
- **Unification** is the parameter passing mechanism.
- Backtracking needed for completeness!

"Efficient" methods for the above are possible when the rewrite system has useful properties of **termination** and **confluence.**

Data Types using Rewrite Systems

Quite easy to model and reason about many data types. nil and $\cdot$ constructors for list, $empty$ and $push$ constructors for stack.

$$\begin{array}{rcl} top(push(x,y)) & \rightarrow & x \\ pop(push(x,y)) & \rightarrow & y \\ append(nil, Z) & \rightarrow & Z \\ append(X \cdot Y, Z) & \rightarrow & X \cdot append(Y, Z) \\ rev(nil) & \rightarrow & nil \\ rev(X \cdot Y) & \rightarrow & append(rev(Y), X \cdot nil) \end{array}$$

Are properties such as **associativity** of append or $rev(rev(X)) = X$ valid? (Equational proofs exist?).

No! These are Inductive properties.

Inductive Properties and Proofs

Example of series summation

$$\begin{array}{rcl} ssum(0) & \rightarrow & 0 \\ ssum(s(x)) & \rightarrow & s(x) + ssum(x) \end{array}$$

Can we prove

$$s^2(0) * ssum(x) = s(x) * x$$

Anecdote about Ramanujan

```
   ans =  1 + 2 +  ...    + n
   ans =  n + (n-1) + ... + 1


 2 * ans =  n * (n+1)
```

Two methods for automating

- Structural Induction using **cover sets**
  - Base Case (0)
  - Inductive Case (s(x))
- Inductionless Induction

  Add "inductive theorem" as a rule and check if we can generate a **contradiction** (equality between different constructors such as $0 = 1$).

Not complete. But rest of rules are similar.

Note: Last rule above cannot be applied without doing a (recursive) **validity** proof!

Proving termination and confluence quite a challenge (my Ph.D. thesis was in this area).

Challenge: Do this without conditional rules.

RRL (Rewrite Rule Laboratory)

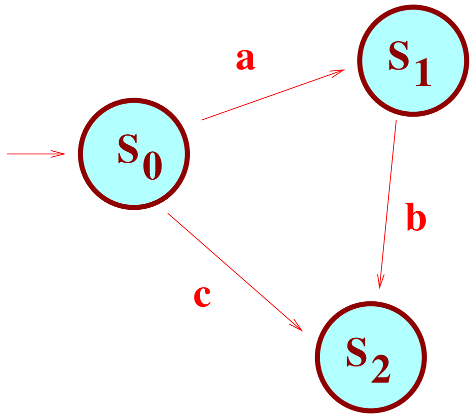
- A theorem prover written in Lisp.
- Initial version (1984)- Deepak Kapur and G. Sivakumar
- Hantao Zhang (U. of Iowa) added many major components.

Features of RRL

- A Rewrite Rule Based Prover for Equational and Inductive Reasoning
- Formulas as Terminating Rewrite Rules
- Conditional rewriting and Normalization
- Conditions discharged using Decision procedures and Rewriting (**Contextual Rewriting**)
- Heuristics for Induction Schemes, Generalization, Backtracking (**Cover Set** induction method)
- Alternative Proof Attempts, Automatic Lemma generation, Backtracking

Translating Finite State Systems

Very basic idea.



status(sys(S_0), a · X)→status(sys(S_1) , X)

status(sys(S_1) , b · X)→status(sys(S_2) , X)

status(sys(S_0) , c · X)→status(sys(S_2) , X)

Note: $sys(...)$ can easily capture the "state" (possibly infinite) of the system and use many data types (counters, lists, stacks, sorted queues etc.)

Mutual Exclusion (Readers/Writers)

Parameters

- list of active readers (xrd),
- list of active writers (xwr),
- queue of waiting readers (xrdQ),
- the queue of waiting writers (xwQ).

$sys(xrd, xwr, xrdQ, xwQ)$ denotes the system.

Input could be a string of requests

$rb(xt_1) \cdot wb(xt_2) \cdot rb(xt_3) ... \cdot nil$

read-begin request for xt_1 time units, next a write-begin request for xt_2 time units and so on. The xt_i are numbers.

Sample Rule

$$status(sys(xrd, nil, xrdQ, xwQ), rb(t_1) \cdot X) \rightarrow$$

$$status(sys(insert(rb(t_1), xrd), nil, xrdQ, xwQ), X)$$

Property Correspondence

Rewrite System Prop.	Relationship	Protocol Property
Sufficient Completeness	$\Longleftrightarrow$	Completeness
Sufficient Completeness	$\Longrightarrow$	Progress
Termination	$\Longrightarrow$	Boundedness
Non-termination	$\Longrightarrow$	Unbounded or No Progress
Confluence	$\Longleftrightarrow$	Confluency

Example applications

- Alternating-bit protocol
- Authentication protocols
- Distributed control systems
- ...

Ongoing work at Centre for Formal Design and Verification of Software (CFDVS)

http://www.cfdvs.iitb.ac.in/

Emerging Challenges

Open Problems (Meadows 2000)

- Growth in Protocol Complexity
  - Needham-Schroeder (1978): 6 pages
  - TLS: 80 pages
  - SET: 5 main sub-protocols, 3 manuals, nearly 1000 pages
- Open Ended (group messaging- number of principals/fields not known)
- Denial of service (fail-stop protocols)
- Anonymous comm. (voting, auction– more in next slide)
- Over Abstraction (attack on particular methods)
- Composability
- Practical acceptability of formal methods.

Online Voting

Are we ready for elections via Internet?

- George Bush (Nov 2000, dimpled chads)
- Pervez Musharaf (April 2002)

E-Voting Protocols Requirements

- No loss of votes already cast (reliability)
- No forging of votes (authentication)
- No modification of votes cast (integrity)
- No multiple voting
- No vote secrecy violation (privacy)
- No vulnerability to vote coercion
- No vulnerability to vote selling or trading protocols (voter is an adversary)
- No loss of ability to cast and accept more votes (availability, no denial of service)

Other Desirable Properties

must not only be correct and secure, but also be seen to be so by skeptical (but educated and honest) outsiders.

- Auditability:

  Failure or procedural error can be detected and corrected, especially the loss of votes.

- Verifiability: Should be able to prove
  - My vote was counted
  - All precincts were counted
  - The number of votes in each precinct is the same as the number of people who voted
  - No one I know who is ineligible to vote did so
  - No one voted twice
  - Etc.

without violating anonymity, privacy etc.

Zero Knowledge Proofs Challenge Puzzle: Spy-vs-Spy

References

Only a sample. See these for more.

- D. Dolev and A. Yao, "On the security of public-key protocols," IEEE Transactions on Information Theory, 2(29), 1983".
- Burrows and Abadi and Needham "A Logic of Authentication," Proceedings of the Royal Society, Volume 426, Number 1871, 1989.
- Needham R M and Scroeder M. D, "Using encryption for authentication in large networks of computers" Communications of the ACM, 1978, 993–999, December.
- A. D. Rubin and P. Honeyman, "Formal Methods for the Analysis of Authentication Protocols" CITI Technical Report 93-7, 1993 (available at citeseer.nj.nec.com/rubin93formal.html)
- Levente Buttyan, "Formal methods in the design of cryptographic protocols (state of the art)", 1999 (available at citeseer.nj.nec.com/buttyan99formal.html)
- Catherine Meadows, "Open Issues in Formal Methods for Cryptographic Protocol Analysis", 2001 (available at citeseer.nj.nec.com/meadows00open.html)