Synthesising Efficient and Effective Security Protocols

Chen Hao, John Clark, Jeremy Jacob
Department of Computer Science
University of York, York, YO10 5DD
United Kingdom

ARSPA Workshop, Cork, 4 July 2004

Motivation

- Search techniques such as ***simulated annealing*** and ***genetic algorithms*** have proved successful across many domains
- Very little published discussion on the issue of protocol efficiency (non-functional requirements)
  - most work have focused on the *security* of protocols
- Can we use these heuristic search techniques to find secure and efficient protocols?

Protocol Design As Search

- We will express protocol design as a combinatorial search problem
- We will assign a fitness to protocol designs indicating how “good” they are
- We will use heuristic search technique (simulated annealing) to find a design with high fitness

Design As Search

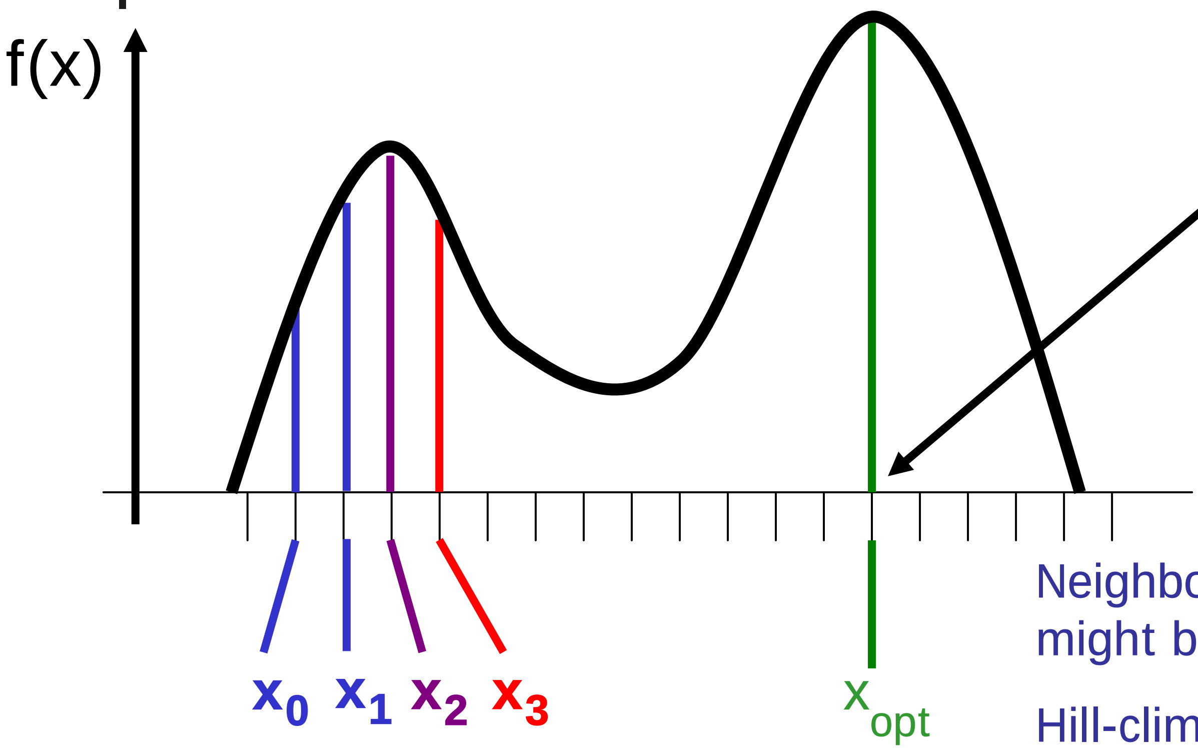
- choose initial value of *P*

  Until stopping criterion *do*

  choose new *P* from neighbourhood of old *P*

  end
- Guided search typically chooses assignment that improves the fitness
- Sometimes, fitness needs to get worse before it can get better

Local Search - Hill Climbing



Really want to obtain x_{opt}

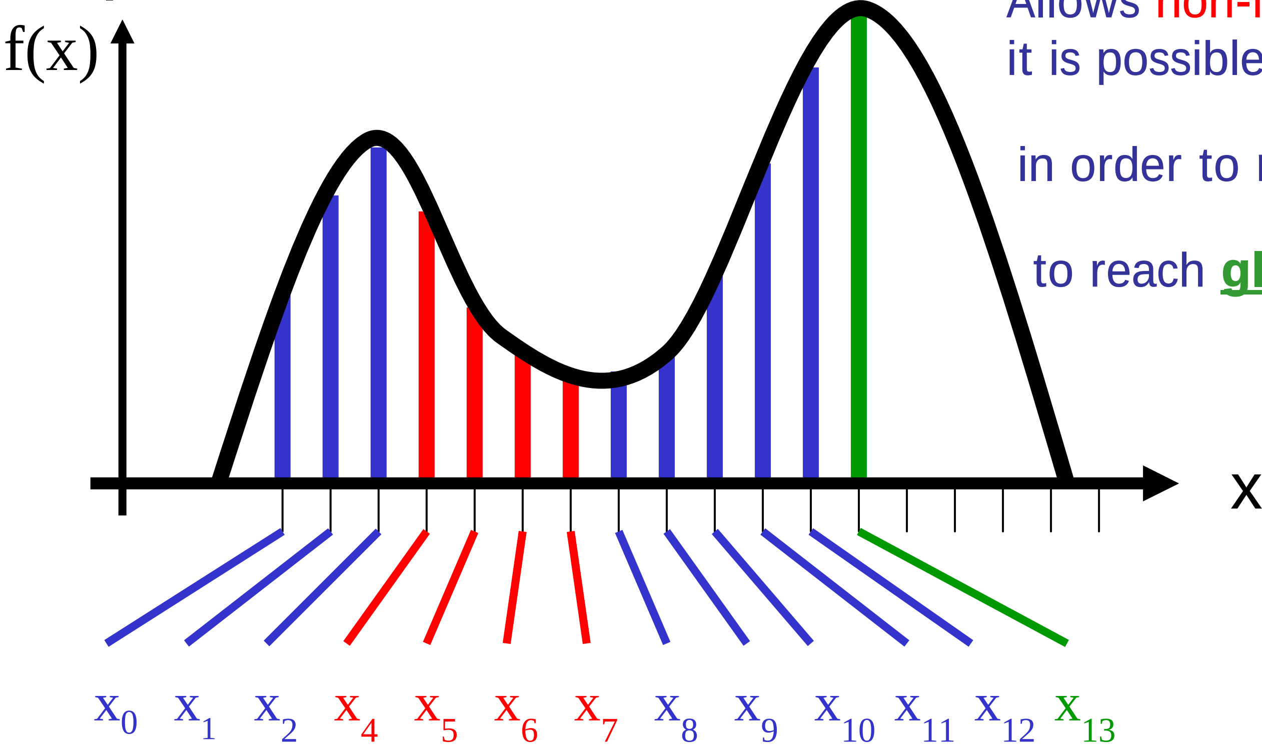
Neighbourhood of a point x_n might be $N(x_n) = \{x_{n-1}, x_{n+1}\}$

Hill-climb goes $x_0 \rightarrow x_1 \rightarrow x_2$ since

$$f(x_0) < f(x_1) < \underline{f(x_2)} > f(x_3)$$

and gets stuck at x_2 (local optimum)

Simulated Annealing



Allows non-improving moves so that it is possible to go down

in order to **rise again**

to reach **global** optimum

In practice neighbourhood may be very large and trial neighbour is chosen randomly. Possible to accept worsening move when improving ones exist

Simulated Annealing

- **Improving moves** always accepted
- **Non-improving moves** may be accepted probabilistically and in a manner depending on the temperature parameter **T**. Loosely
  - the **worse** the move, the **less likely** it is to be accepted
  - the **cooler** the temperature, the **less likely** a worsening move is to be accepted
- The temperature **T** starts high and is gradually cooled as the search progresses
  - Initially, <u>virtually anything</u> is accepted; at the end, only improving moves are allowed (and the search effectively reduces to hill-climbing)

Simulated Annealing

- Current candidate x Maximization formulation

$x := x_0$

$T := T_0$

at each temperature consider 400 moves

repeat until stopping criterion is met

repeat 400 **times**

pick $y \in N(x)$ with uniform probability

pick $U \in (0,1)$ with uniform probability

$\Delta = f(y) - f(x)$

if $(\Delta > 0)$ current $x := y$ (**accept**)

else if $(\Delta > T \times \ln U)$ current $x := y$ (**accept**)

else reject

$T := T \times 0.97$

Solution is best so far

Temperature cycle

Always accept improving moves

Accept worsening moves probabilistically

Gets harder to do this the worse the move

Gets harder as temperature decreases

Simulated Annealing

$T = 100$

Do 400 trial moves $T = T \times 0.97$

Do 400 trial moves $T = T \times 0.97$

Do 400 trial moves $T = T \times 0.97$

⋮

Do 400 trial moves $T = T \times 0.97$

⋮

Do 400 trial moves $T = 0.00001$

Specification

- Security Goals
  - pre/post conditions in SVO & GNY Logics
  - Illustrative example
- Efficiency Goals
  - e.g. minimise the number of messages, server interactions and so on
- Fitness of a protocol based on both security criterion and efficiency criterion
- Aim
  - find a protocol with high fitness

Fitness Function

- We need a fitness function to capture the attainment of goals (Security Criterion) and evaluate how "efficient" (Efficiency Criterion) a protocol is
- $$f(P) = s(P) + e(P)$$

Security Fitness

$$s(P) = \sum_{i=1}^{N} \left(\sigma + \delta(i) \right) \times G(P, i)$$

A large constant that weights security much more heavily than efficiency

Number of new required security goals that message i of P achieves

Weights among individual messages (e.g. Early Credit strategy: the weights are monotonically decreasing with i. The notion is that satisfying goals early should be rewarded)

Efficiency Fitness

$$e(P) = m(P) + c(P) + r(P)$$

$$m(P) = \mu \times M(P)$$

Punish protocols with many messages

$$c(P) = \kappa \times C(P)$$

Punish protocols with more encryption

$$r(P) = \sum_{a \in A(P)} \rho(a) \times R(P, a)$$

Punish number of interactions with particular principals

Decoding

- Abstract design space = protocols expressed in SVO logic
- Encoded search space = sequences of non-negative integers
- Decode integer sequences as SVO protocols so that we can evaluate the fitness of these protocols

SVO Logic

- Efficiently unify previous logics (BAN, GNY, AT and VO)
- SVO rules: define deductions from receipt of a message
- Message comprehension and message interpretation steps of SVO almost preclude automated reasoning
- We use GNY recognisability rule and message extension to overcome this limitation

Back

Illustrative example

- Goals
  - A has K_{ab}

    A believes $A \xleftrightarrow{K_{ab}} B$
- Assumptions
  - A has (A, B, S, N_a, K_{as}); S has $(A, B, S, K_{as}, K_{ab})$;

    A believes $A \xleftrightarrow{K_{as}} S$; S believes $A \xleftrightarrow{K_{as}} S$;

    A believes $fresh(N_a)$; A believes $\phi(N_a)$;

    S believes $A \xleftrightarrow{K_{ab}} B$; A believes $(S$ controls $A \xleftrightarrow{K_{ab}} B)$

Back

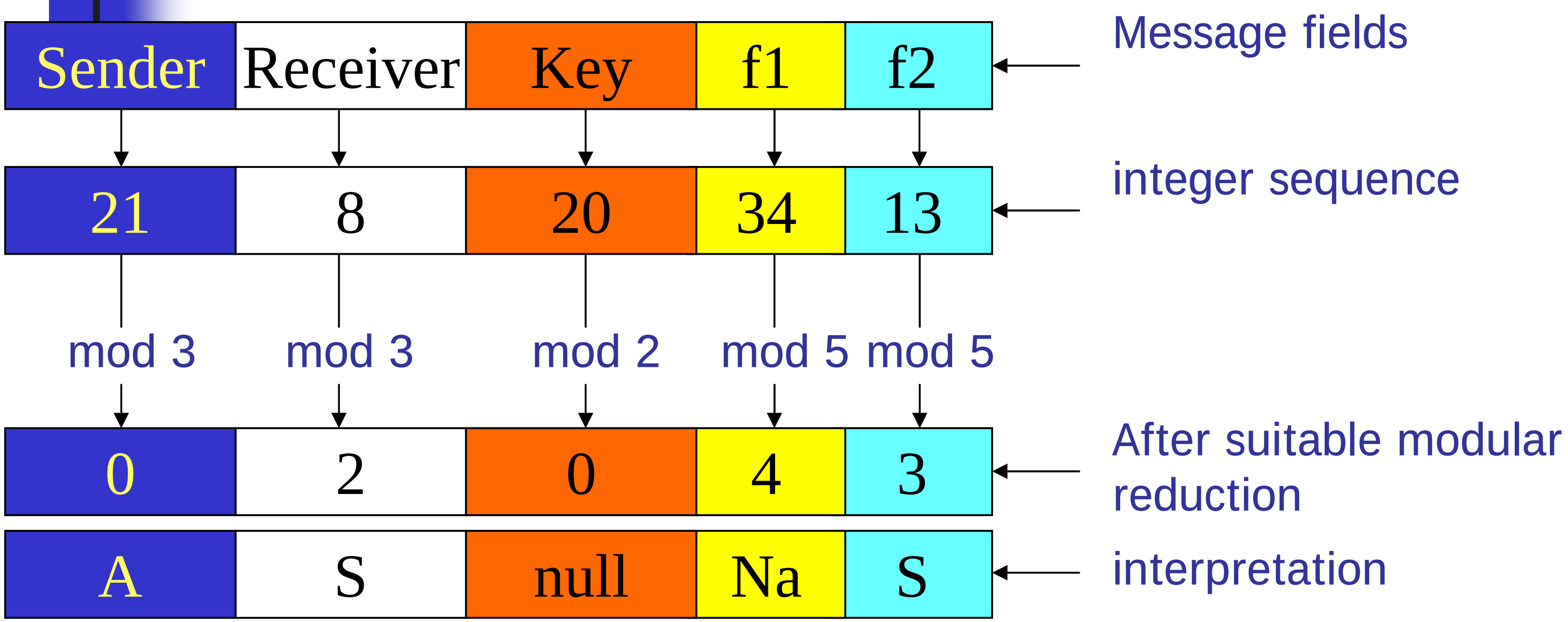
Illustrative example

- A feasible SVO protocol
  - $1.\ A \rightarrow S:\ A, B, N_a$

    $2.\ S \rightarrow A:\ \{N_a, K_{ab} \succ A \xleftrightarrow{K_{ab}} B\}_{K_{as}}$

Back

Messages as Integer Sequences



Interpretation for 3 principals A, B, S (A=0, B=1, S=2);

sender A currently holds 5 possessions and 2 keys

Af0 is the null possession and Ak0 is the null key

Search Strategy

- We can now interpret sequences of non-negative integers as a valid protocol
  - Interpret each message in turn updating belief/possession/key vectors after each message (by applying logic rules)
  - This is the **execution** of the abstract protocol
  - **Every protocol achieves something! The issue is whether it is something we want!**
- We generate the neighbourhood by randomly changing one integer and assessing fitness
  - This can change the sender, receiver or a component of any message

Examples

- Security Goals: (award +3000 for each achieved goal)

$$A \text{ has } K_{ab} \qquad A \text{ believes } A \xleftrightarrow{K_{ab}} B$$

$$B \text{ has } K_{ab} \qquad B \text{ believes } A \xleftrightarrow{K_{ab}} B$$

$$A \text{ believes } B \text{ has } K_{ab} \qquad B \text{ believes } A \text{ has } K_{ab}$$

- Assumptions: standard
- Efficiency Weights:
  - -200 for each message
  - -200 for each encryption
  - -100 and -50 for each server and client interaction respectively (for the 1st example)

Examples

1st Example

1. $A \rightarrow S: \; A, B, N_a$
2. $S \rightarrow A: \; \{N_a, K_{ab} \succ A \xleftrightarrow{K_{ab}} B\}_{K_{as}}$
3. $B \rightarrow S: \; B, A, N_b$
4. $S \rightarrow B: \; \{N_b, N_a, K_{ab} \succ A \xleftrightarrow{K_{ab}} B\}_{K_{bs}}$
5. $B \rightarrow A: \; \{B, N_b, N_a\}_{K_{ab}}$
6. $A \rightarrow B: \; \{N_b, A\}_{K_{ab}}$

4 server interactions

2nd Example

1. $A \rightarrow B: \; A, N_a$
2. $B \rightarrow S: \; B, N_b, A, N_a$
3. $S \rightarrow B: \; \{N_b, K_{ab} \succ A \xleftrightarrow{K_{ab}} B\}_{K_{bs}}$
4. $S \rightarrow A: \; \{N_a, N_b, K_{ab} \succ A \xleftrightarrow{K_{ab}} B\}_{K_{as}}$
5. $A \rightarrow B: \; \{A, N_a, N_b\}_{K_{ab}}$
6. $B \rightarrow A: \; \{B, N_a\}_{K_{ab}}$

3 server interactions

Conclusions

- We can use search to generate secure and efficient protocols
- We can generate protocols at logic level in a few minutes

Future Work

- Automated refinement to code
- Use protocols as candidates for further analysis with model checkers (give a different kind of analysis)
- Prettier user interfaces to the tool
- Can we use heuristic search to find flaws in protocols?