A First Step towards the Automatic Generation of Security Protocols

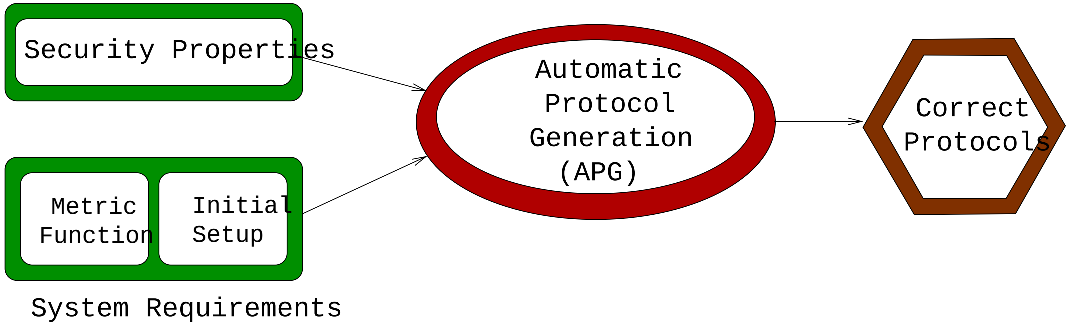
Adrian Perrig and Dawn Song
CMU, UCB

Difficulties in the Design of Security Protocols

- Usually ad-hoc, lacking formalism. Hidden assumptions weaken security.
- Error-prone. A Classic Example: Needham-Schroeder public key authentication protocol [NS78], in which Gavin Lowe discovered a flaw 18 years later! [Low96]
- Limited proof of security, low confidence
- Limited search capability of designer, results in suboptimal protocols
- Slow process. Fixing flaws can be expensive

Automatic Protocol Generation

- User enters security requirements and system specification and APG outputs the optimal secure protocol



- APG consists of a protocol generator and a protocol verifier, for which we use Athena

Advantages of APG

- Fully automatic, no user intervention
- High confidence
- High Quality
- Flexible
- Custom-tailored security protocols for each application

Grammar to Generate Security Protocols

- Grammar for representing messages in authentication protocols

$$Message ::= Atomic \mid Encrypted \mid Concatenated$$

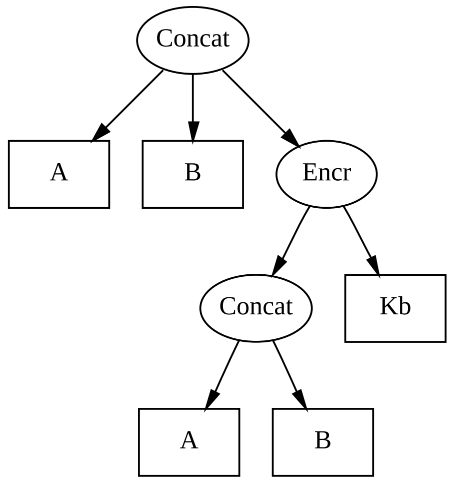
$$Atomic ::= \textit{Principalname} \mid \textit{Nonce} \mid Key$$

$$Encrypted ::= (Message, \textit{Key})$$

$$Key ::= \textit{PublicKey} \mid \textit{PrivateKey} \mid \textit{SymmetricKey}$$

$$Concatenated ::= Message,\ Message \mid Message,\ Concatenated$$

- Message representation through a tree structure



Metric Function to describe System Requirements

- Metric reflects the utility function, which defines the cost of a protocol
- Assign a cost to each operation

Operation	Sample 1	Sample 2
Sending cost per atomic element	1	3
Nonce generation	1	1
Symmetric encryption/decryption	3	1
Asymmetric encryption/decryption	7	2

- E.g. the cost of the message $A, B, \{A, B\}_{K_{AB}}$ is 8 (Sample 1).
- A correct protocol with the minimal cost is the optimal protocol (with respect to the metric function).

Sacrifice Completeness to Achieve Practicality

- Vast protocol space
  - Even for two-party mutual authentication protocols might take years for a protocol verifier to explore
  - Our goal is to make APG interactive
- Limiting the depth of the messages reduces the protocol space
- Don't consider permutation of message components
  $\{A, N_A\}_{K_{AB}} \equiv \{N_A, A\}_{K_{AB}}$

The Athena security protocol verifier [Son99]

- Automatic verifier for security protocols
- Model checker / theorem prover hybrid
- Uses the Strand Space Model [THG98]
- Athena either proves correctness (without a bound on the number of sessions) or gives a counterexample
- Highly efficient, on the order of 10 prot/s (3 parties, 4 rounds)

Case Study: Automatic Generation of Two-Party Mutual Authentication Protocols

- Explore two-party mutual authentication protocols for different settings
  - Authentication using either symmetric or asymmetric keys
  - Principals are either bandwidth-limited or communication-limited
- Good starting point - large number of known protocols to compare against

Overcome the Protocol Space Explosion Problem

- Despite the optimisations, the protocol space is still vast
- Solution: Add a simple and fast protocol verifier to the generator
- Look for simple impersonation attacks
- Recognize simple replay attacks
- Result: Fast to check, yet highly effective

Type	Cost	Generated	I.A.	R.A.	Comb.	Cand.	Corr.
Symmetric	10	19856	12098	18770	19449	407	2
Asymmetric	14	46518	46378	40687	46408	110	1

Impersonation Attack Module

- Each principal has an impersonator, I_A for A, I_B for B
- Each impersonator is updated as follows
  - Knows all principal names
  - Knows all public keys
  - Receives all of its principal's nonces
  - Eavesdrops messages and reads what it can decrypt
- Example protocol:

$$\begin{aligned} \text{Protocol}: \quad & A \rightarrow B : N_A, A \\ & B \rightarrow A : N_B, \{N_A, A, B\}_{K_{AB}} \\ & A \rightarrow B : N_A, N_B \end{aligned}$$

I_A can easily impersonate A

Replay Attack Module

- Detects attacks where an eavesdropper can impersonate a principal by replaying messages from a previous run
- Example protocol:

$$\begin{aligned} \text{Protocol :} \quad & A \rightarrow B : A, \{N_A, A\}_{K_{AB}} \\ & B \rightarrow A : \{N_A, N_B, A, B\}_{K_{AB}} \\ & A \rightarrow B : N_A, B \end{aligned}$$

- An adversary can impersonate A by replaying messages 1 and 3

Results: Symmetric-Key Authentication Protocols

- Minimal protocols (cost $=$ 10) for sample 1 costs
- Optimal protocols for computation-limited systems

$$\text{Protocol :} \quad \begin{aligned} &A \rightarrow B : N_A, A \\ &B \rightarrow A : \{N_A, N_B, A\}_{K_{AB}} \\ &A \rightarrow B : N_B \end{aligned}$$

$$\text{Protocol :} \quad \begin{aligned} &A \rightarrow B : N_A, A \\ &B \rightarrow A : \{N_A, N_B, B\}_{K_{AB}} \\ &A \rightarrow B : N_B \end{aligned}$$

Results: Symmetric-Key Authentication Protocols II

- For bandwidth-limited devices, we want to minimise communication overhead
- Increasing the sending cost reveals the following optimal protocol

$$\begin{aligned}\text{Protocol :} \qquad & A \rightarrow B : \{N_A, A\}_{K_{AB}} \\ & B \rightarrow A : \{N_A, N_B\}_{K_{AB}} \\ & A \rightarrow B : N_B\end{aligned}$$

Results: Asymmetric-Key Authentication Protocols

- In the case of asymmetric keys, the fixed version of the Needham-Schroeder protocol is optimal for communication-limited and computation-limited settings

$$\begin{aligned} \textsf{Protocol:} \quad & A \rightarrow B : \{N_A, A\}_{K_B} \\ & B \rightarrow A : \{N_A, N_B, B\}_{K_A} \\ & A \rightarrow B : N_B \end{aligned}$$

Remaining Challenges / Future Work

- Current work is on three-party authentication protocols
- Protocol space grows exponentially in protocol complexity
- Automatic generation of source code
- Repair of flawed protocols, protocol optimisation

Conclusions

- Initial results look promising, APG needs further study
- Even though two-party mutual authentication protocols were intensely studied, APG discovered novel and efficient protocols
- APG generates custom-tailored optimal protocols for each application

References

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[Son99] Dawn Song. Athena: An automatic checker for security protocol analysis. In *Proceedings of the 12th Computer Science Foundation Workshop*, 1999.

[THG98] F.Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In

Proceedings of 1998 IEEE Symposium on Security and Privacy, 1998.