An Overview of Common Adversary Models

Karl Palmskog
palmskog@kth.se

2012-03-29
Requirements of Software Systems

1. Functional
   - Correctness: partial, termination, liveness, safety, ...

2. Nonfunctional
   - Performance: time/memory/message complexity, ...
   - Security: ...

3. Architectural

4. ...
Security Requirements: Some Questions

- Why do we need security?
  - Assuming a nonadversarial world, do we need security at all?
  - What are we protecting?
  - Who are we protecting it from?

- How do we describe security?
  - What assumptions must be made?
  - What are the capabilities of the adversary?
“Encryption is not synonymous with security.”

— Martín Abadi
Examples of Assumptions About the World

<table>
<thead>
<tr>
<th>Type</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fundamental</td>
<td>$\mathcal{P} \neq \mathcal{NP}$</td>
</tr>
<tr>
<td>Fundamental</td>
<td>exists 1-way functions</td>
</tr>
<tr>
<td>Problem-Specific</td>
<td>Decision Diffie-Hellman</td>
</tr>
<tr>
<td>Problem-Specific</td>
<td>Computational Diffie-Hellman</td>
</tr>
<tr>
<td>Problem-Specific</td>
<td>Discrete Logarithms</td>
</tr>
<tr>
<td>Situation-Specific</td>
<td>exists trusted party</td>
</tr>
</tbody>
</table>
Building Blocks of Cryptography

1-way hash
- SHA

shared-key
- AES

public-key
- RSA
Examples of Properties of the Adversary

**Computational Power**
- unlimited/bounded/structurally limited

**Intent**
- curious/hostile
“We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material.”

— Needham/Schroeder (1978)
Current Standard Capabilities of the Adversary

- Participate in some protocol runs
- Know certain data in advance
- Intercept message on some or all communication paths
- Inject any messages that it can produce
Unconditional Security: “trust nothing”

- Adversary has unbounded computational resources
- Must not obtain information from observing ciphertext

**Definition**

A cryptosystem has *perfect secrecy* if the *a posteriori* probability that the plaintext is $x$, given that the ciphertext $y$ is observed, is identical to the *a priori* probability that the plaintext is $x$. 
The Formal Model: “trust your primitives”

- Assume perfect cryptographic primitives (“black boxes”)
- Messages exchanged are terms on cryptographic primitives
- Adversary is restricted to only reason on terms, e.g.
  - substitute terms for variables in equations
  - use equation terms in other equations
- Example equations for symmetric cryptography:

\[
\forall x \forall y \ sdec(senc(x, y), y) = x
\]

\[
\forall x \forall y \ scheck(senc(x, y), y) = ok
\]
An Overview of Common Adversary Models

- Models
- The Formal Model

The Formal Model Illustrated

\[
\text{senc}(m_1, K_B) \quad \text{senc}(m_2, K_A)
\]

\[O\]

\[A \quad B\]
The Formal Model Illustrated

\[ senc(m_1, K_B) \]
The Formal Model Illustrated

\[ \text{senc}(m_1, K_B) \]

\[ \text{senc}(m_2, K_A) \]
Example Properties in the Formal Model

Secrecy

Adversary cannot obtain the secret

Correspondance

Authentication

Strong Secrecy

Adversary does not see the difference when the value of the secret changes
Pros and Cons of the Formal Model

+ simple
+ tool support
+ necessary for security
  - insufficient for security
  - unrealistic?
The Computational Model: “limit trust in your primitives”

- Messages are bitstrings
- Adversary is a polynomial-time probabilistic Turing machine
- Adversary can do low-level bit operations on messages
- Assumes Computational Diffie-Hellman
The Computational Model Illustrated
The Computational Model Illustrated
An Overview of Common Adversary Models

Models

The Computational Model

The Computational Model Illustrated

A

B

0010010011

0101011010

0010010011

0101011010
Example Properties in the Computational Model

Secrecy
Adversary cannot obtain the secret

Correspondences
Authentication

Resilience
Probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions
Pros and Cons of the Computational Model

+ sufficient for probabilistic security
+ reduction-based
+ realistic?
  - complicated
  - tool support
Byzantine Fault Tolerance

- Distributed system with \( n \) nodes connected in a network
- \( m < n \) nodes behave erratically (can omit or falsify messages)

Lemma

Suppose we have a network with nodes \( n_1, n_2 \) and \( n_3 \), where \( n_3 \) behaves erratically. Then \( n_1 \) and \( n_2 \) cannot become in agreement on a value by network communication.

Theorem

Reaching agreement by network communication (without using cryptographic assumptions) is only possible when \( n \geq 3m + 1 \).
Multiparty Computation

- $n$ parties communicating through a network
- Each party has private input and knows function to compute
- $t < n$ parties are passively or actively corrupted
Example Properties in Multiparty Computation

Secrecy

Players’ inputs remain secret

Correctness

Results of the computation are correct

Resilience

Above holds despite corruption
An Overview of Common Adversary Models

- Models
- Universal Composability Framework

Universal Composability Framework

- Adversary is any interactive probabilistic polynomial time Turing machine
- Exists “operating system” that takes care of subprotocols
- Asynchronous network in *ideal* or *real* communication model
  - Ideal “Dummy” parties, but has trusted party performing ideal functionality
  - Real “Real” parties, adversary and environment
Universal Composability Real Model Illustrated
Universal Composability Ideal Model Illustrated
An Overview of Common Adversary Models

Some Properties of Universal Composability

- A protocol $\pi$ in the real model securely realizes an ideal functionality $\mathcal{F}$ if for any real adversary $A$, there exists ideal adversary $S$ such that no environment $Z$ can tell\textsuperscript{1} whether it is interacting with real or ideal model.
- If the protocol $\pi$ securely realizes some functionality $\mathcal{F}$, $\pi$ can be used instead of the functionality regardless of how $\mathcal{F}$ is employed.
- Protocols remain secure even if arbitrarily composed with other instances of the same or other protocols.

\textsuperscript{1}with non-negligible probability
Assume variables in a program $P$ are divided into levels, e.g.:
- $L$ (low) for publicly visible variables
- $H$ (high) for private, or secret, variables

Assume adversary:
- Knows the syntax and semantics of $P$
- Can observe $L$-variables before and after executing $P$
Example Properties of Information Flow

- Information about secret $s$ can be exposed by:
  - explicit flow: a variable in $L$ being assigned $s$
  - implicit flow: branching on $s$ and assigning to variable in $L$

- Define noninterference for program $P$:

  \[
  \forall \sigma_1, \sigma_2. \, \sigma_1 \approx_L \sigma_2 \Rightarrow P(\sigma_1) \approx_L P(\sigma_2)
  \]

- Can be generalized to distributed systems
  - use logics of knowledge
An Overview of Common Adversary Models

Summary

Applicability

- Adversary model as part of designing a software system
- Explicit assumptions about
  - Cryptographic primitives
  - Resources of adversary
  - Intent of adversary
  - Authenticity requirements
  - Secrecy requirements
- Tradeoff between correctness, resource usage, security and performance
The Ideal System

**Functional Requirements**
Certificate of adherence to specification

**Performance Requirements**
Certificate of adherence for performance model to performance requirements and evidence that it represents real system

**Security Requirements**
Certificate of security against specified adversary model
Questions to Ponder

- What is the adversary model for a simple web service?
- How do XSS attacks fit into this model?
Further Reading


