Probabilistic Security Games

Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute

Seminar in honour of Professor Bhamidi V. Rao
January 15, 2008

Cryptographic Protocols

- Designed for achieving cryptographic functionalities.
- Examples:
  - Encryption (public key or symmetric key).
  - Signature.
  - Public Key Agreement.
  - Different variants of these protocols.
- Two cardinal requirements:
  - security,
  - efficiency.

Protocol Components

- Smaller protocols.
- Algebraic/number theoretic operations.
- Security based on one or both of the following assumptions.
  - The smaller protocols are secure.
  - Some problem is computationally hard.

Secure and Efficient Protocols

- Design: requires algebraic insight.
- Defining and proving security: uses ideas from
  - computational complexity theory;
  - probability theory;
  - algebra and combinatorics.

Protocol Structure

A protocol consists of several (probabilistic) algorithms.

- Set-Up: this algorithm generates
  - public parameters (PP) following some (usually uniform) distribution;
  - corresponding secret key sk.
- Other algorithms depend on the type of protocol.
  - Encryption Protocol: encryption, decryption algorithms; possibly other algorithms.
  - Signing Protocol: signature generation, verification algorithms.

Security Definition

The basic idea behind the definitions is to capture computational indistinguishability.

Examples:

- Encryption protocols: captures the idea that a properly generated ciphertext is indistinguishable from random strings.
- Signature protocols: captures the idea that a message-signature pair looks like a random string.

Security Game

Security is modelled using an interactive game between an adversary and a simulator.

Basic Structure:

- Simulator: sets up the protocol; gives public parameter (PP) to adversary; keeps secret key sk.
- Adversary: asks the simulator certain questions,
  - answering questions may require sk;
  - questions are considered as oracle queries;
  - there may be more than one oracle;
- Adversary: produces some information at the end of the game.

Security Game (contd.)

Example: encryption protocol.

- Simulator: runs Set-Up.
- Adversary: executes Phase 1 queries.
- Adversary: presents the simulator with two equal length messages M_0 and M_1.
- Simulator: chooses a random bit γ and gives the adversary an encryption C^* of M_γ
- Adversary: executes Phase 2 queries (with some natural restrictions)
- Adversary: finally outputs a bit γ'

Adversary's Advantage

For an adversary $\mathcal{A}$,

$$\mathsf{Adv}(\mathcal{A}) = \left| \mathsf{Pr}[\gamma = \gamma'] - \frac{1}{2} \right| .$$

- Resource constraints on $\mathcal{A}$.
  - bound on runtime,
  - bound on the number of oracle queries.
- $\mathsf{Adv}(t, q)$: maximum (supremum) of $\mathsf{Adv}(\mathcal{A})$, over all adversaries $\mathcal{A}$ running in time t and making q oracle queries.

Security Assurance

If

smaller protocols are secure
and
some problem Π is computationally hard

then

the main protocol is secure.

Structure of Proofs

A Game Sequence

$G_0,$
$G_1,$
$\vdots$
G_k

- Let X_i be the event that $\gamma = \gamma'$ in Game G_i. We consider

$$\begin{aligned}
&\mathsf{Pr}[X_0],\\
&\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1],\\
&\vdots\\
&\mathsf{Pr}[X_{k-1}] - \mathsf{Pr}[X_k]\\
&\mathsf{Pr}[X_k].
\end{aligned}$$

Structure of Proofs (contd.)

- G_0 is the game which defines the security of the protocol and so

$$\mathsf{Adv}(\mathcal{A}) = |\mathsf{Pr}[\gamma = \gamma'] - 1/2| = |\mathsf{Pr}[X_0] - 1/2|.$$

- G_k is designed such that the bit γ is statistically hidden from the adversary. So,

$$\mathsf{Pr}[X_k] = 1/2.$$

- Games G_{i-1} and G_i differ:
  - the difference is not too much;
  - the adversary should not be able to notice whether he is playing Game G_{i-1} or Game G_i.

Structure of Proofs (contd.)

- More precisely, $\mathsf{Pr}[X_{i-1}] - \mathsf{Pr}[X_i]$ is bounded above by
  - either, the advantage of an adversary in breaking one or the smaller protocols;
  - or, the advantage of solving problem Π.

$$
\begin{aligned}
\mathsf{Adv}(\mathcal{A}) &= |\mathsf{Pr}[X_0] - 1/2| \\
&= |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_k]| \\
&\leq |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| \\
&\quad + |\mathsf{Pr}[X_1] - \mathsf{Pr}[X_2]| \\
&\quad + \cdots \\
&\quad + |\mathsf{Pr}[X_{k-1}] - \mathsf{Pr}[X_k]|.
\end{aligned}
$$

Some Hard Problems

Let $G = \langle g \rangle$ be a (suitable) group.

Diffie-Hellman (DH) Problem:

Instance: (g, g^a, g^b),
a, b unknown (and randomly chosen).

Task: Compute g^{ab}.

Decision Diffie-Hellman (DDH) Problem:

Instance: (g, g^a, g^b, h),
a, b unknown (and randomly chosen).

Question: Is $h = g^{ab}$ or is h a random element of G?

"Solving" Π

$$\mathsf{Adv}_{\mathrm{DDH}}(\mathcal{B}) = \big|\mathsf{Pr}[\mathcal{B} \Rightarrow 1 \big| h \text{ is "real"}] - \mathsf{Pr}[\mathcal{B} \Rightarrow 1 \big| h \text{ is "random"}]\big|.$$

Here $\mathcal{B}$ is an algorithm to solve Π.

- probabilistic algorithm;
- outputs a bit, i.e., 0 or 1;
- $\mathcal{B} \Rightarrow 1$ denotes that $\mathcal{B}$ outputs 1.

Relating to Π

Input: an instance I of Π.

- Simulator ($\mathcal{B}$): sets up the protocol using I,
  - gives PP to the adversary;
    distribution of PP should be proper.
  - sk is (usually) unknown to the simulator;
    (sk is implicitly defined by I.)
- Adversary ($\mathcal{A}$): makes the oracle queries.
  Simulator ($\mathcal{B}$): has to reply (using I, but, possibly without knowing sk).
- Adversary ($\mathcal{A}$): submits M_0 and M_1;
  Simulator ($\mathcal{B}$): generates challenge ciphertext C;
  – if I is real, then C is a proper encryption of M_γ;
  – if I is random, then C is random.

Relating to Π (contd.)

- Adversary ($\mathcal{A}$): outputs γ';
- Simulator ($\mathcal{B}$): outputs $\gamma \oplus \gamma' \oplus 1$.

Suppose that all queries made by the adversary can be answered by the simulator.

- $\mathsf{Pr}[\mathcal{B} \Rightarrow 1 | I \text{ is real}] = \mathsf{Pr}[\gamma = \gamma']$.
- $\mathsf{Pr}[\mathcal{B} \Rightarrow 1 | I \text{ is random}] = 1/2$.
- $\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}_{\Pi}(\mathcal{B})$.

Relating to Π (contd.)

- Suppose that $\mathcal{A}$ makes a query which cannot be answered by $\mathcal{B}$.
- Then, $\mathcal{B}$ aborts and outputs a random bit.
- The relation $\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}_{\Pi}(\mathcal{B})$ no longer holds.
- If the events "$\mathcal{B}$ aborts" and "$\gamma = \gamma'$" are independent, then

$$\mathsf{Adv}(\mathcal{A}) \leq \mathsf{Adv}_{\Pi}(\mathcal{B})/\mathsf{Pr}[\mathcal{B} \text{ aborts}].$$

(security degradation)

- If the events "$\mathcal{B}$ aborts" and "$\gamma = \gamma'$" are not independent, then the situation is complicated.

"Artificial" Abort

- A technique introduced by Brent Waters (2005).
- Basic idea:
  - obtain a lower bound λ on abort;
  - introduce an extra (artificial) abort stage;
  - ensure that the overall probability of abort is close to λ irrespective of the actual queries made by the adversary.
- Carry out a probability analysis to relate the advantages of $\mathcal{A}$ and $\mathcal{B}$.

Artificial Abort (contd.)

- Consider the point in the game where the adversary has produced γ'.
- At this stage
  - Instance I of Π is known.
  - All queries (i.e., the transcript $\mathcal{T}$) made by the adversary are known;
- Recall that PP is produced from I and is independent of $\mathcal{T}$.
- PP and $\mathcal{T}$ uniquely determine whether $\mathcal{B}$ will abort or not.

Artificial Abort Procedure

- Perform the following for k steps.
  - Generate a new PP from I.
    (Recall that set-up is probabilistic).
  - Given PP and $\mathcal{T}$ determine whether $\mathcal{B}$ needs to abort or not.
- Let $\eta' = \#\text{aborts}/k$.
- If $\eta' \geq \lambda$, then
  - abort with probability λ/η';
  - i.e., not abort with probability $1 - \lambda/\eta'$.
- If not aborted till this point, return $\gamma \oplus \gamma' \oplus 1$.

Summary of Analysis

- Chernoff Bound Analysis:
  - if $k \geq (256/(\lambda\epsilon^2))\ln(16/(\lambda\epsilon))$,
  - then

$$\lambda - (\lambda\epsilon)/2 \leq \mathsf{Pr}[\overline{\mathsf{ab}}|\gamma = \gamma'] \leq \lambda + (\lambda\epsilon)/2.$$

$$\lambda - (\lambda\epsilon)/2 \leq \mathsf{Pr}[\overline{\mathsf{ab}}|\gamma \neq \gamma'] \leq \lambda + (\lambda\epsilon)/2.$$

- The rest of the analysis can be completed.
- Effect on runtime of simulator: k additional iterations.

Open Question: Is there a way to avoid this?

Summary

- Cryptographic Protocols.
- Use of probabilistic games to model security.
- Game sequence style of proofs.
- Complications arising in probability analysis of game sequences.
- An open problem on artificial abort.

Thank you for your attention!