CS 395T

Probabilistic Polynomial-Time Calculus

Security as Equivalence

- Intuition: encryption scheme is secure if ciphertext is indistinguishable from random noise
- Intuition: protocol is secure if it is indistinguishable from a perfectly secure “ideal” protocol
- Security is defined as observational equivalence between protocol and its ideal functionality
  - Both formal methods and cryptography use this approach, but with different notions of what it means for the adversary to “observe” the protocol execution

Bridging the Gap

- Cryptography: observational equivalence is defined as computational indistinguishability
  - No probabilistic poly-time algorithm can tell the difference between the real and the ideal protocol with more than negligible probability
- Formal methods: observational equivalence is defined as some form of process bisimulation
  - No probabilitities, no computational bounds
- Goal: bridge the gap by explicitly supporting probability and complexity in process calculus

Standard Example: PRNG

- **Pseudo-random sequence**

  P_n: let b = n^k*-bit sequence generated from* n *random bits ("seed")*

  in PUBLIC⟨b⟩ end

- **Truly random sequence**

  Q_n: let b = *sequence of* n^k *random bits*

  in PUBLIC⟨b⟩ end

- P is a cryptographically strong pseudo-random number generator if the two sequences are observationally equivalent $P \approx Q$
  - Equivalence is <u>asymptotic</u> in security parameter n

Process Calculus Approach

[Abadi-Gordon and others]

- Write protocol in process calculus
  - For example, applied pi-calculus
- Express security using observational equivalence
  - Standard relation from programming language theory

    $P \approx Q$ iff for all contexts $C[\ ]$,

    same observations about $C[P]$ and $C[Q]$
  - Inherently compositional (quantifies over all contexts)
  - Context (environment) represents adversary
- Use proof rules for $\approx$ to prove observational equivalence to the "ideal" protocol

Challenges

- Probabilistic formal model for crypto primitives
  - Key generation, random nonces, randomized encryption
- Probabilistic attacker
  - Replace nondeterminism with probability
  - Need a formal way of representing complexity bounds
- Asymptotic form of observational equivalence
  - Relate to polynomial-time statistical tests
- Proof rules for probabilistic observational equivalence

Nondeterminism Is Too Strong

- Alice encrypts message and sends to Bob

  $A \rightarrow B: \{ msg \}_K$

- Adversary "nondeterministically" guesses every bit of the key

  Process E_0 $c\langle 0\rangle \mid c\langle 0\rangle \mid \ldots \mid c\langle 0\rangle$

  Process E_1 $c\langle 1\rangle \mid c\langle 1\rangle \mid \ldots \mid c\langle 1\rangle$

  Process E $c(b_1).c(b_2)\ldots c(b_n).decrypt(b_1 b_2 \ldots b_n, msg)$

In reality, at most 2^{-n} chance to guess n-bit key

PPT Calculus: Syntax

- Bounded π-calculus with integer terms

$P ::= 0$		
$\mid$	$c_{q(\|n\|)}\langle T \rangle$	send up to q(|n|) bits
$\mid$	$c_{q(\|n\|)}(x).P$	receive
$\mid$	$\upsilon c_{q(\|n\|)}.P$	private channel
$\mid$	$[T=T]\ P$	test
$\mid$	$P \mid P$	parallel composition
$\mid$	$!_{q(\|n\|)}\ P$	bounded replication

Size of expressions is polynomial in |n|

Terms may contain symbol n;

channel width and replication bounded by polynomial of |n|

Probabilistic Operational Semantics

- **Basic idea: alternate between terms & processes**
  - Probabilistic scheduling of parallel processes
  - Probabilistic evaluation of terms (incl. rand)
- **Outer term evaluation**
  - Evaluate all exposed terms, evaluate tests
- **Communication**
  - Match up pairs “send” and “receive” actions
  - If multiple pairs, schedule them probabilistically
    - Probabilistic if multiple send-receive pairs

alternate

Probabilistic Scheduling

- Outer term evaluation
  - Evaluate all exposed terms in parallel
  - Multiply probabilities
- Communication
  - E(P) = set of eligible subprocesses
  - S(P) = set of schedulable pairs
  - Schedule private communication first
  - Probabilistic poly-time computable scheduler that makes progress

Simple Example

- **Process**

  rand is 0 or 1 with prob. ½

  - $c\langle rand+1\rangle \mid c(x).d\langle x+1\rangle \mid d\langle 2\rangle \mid d(y).e\langle y+1\rangle$

- **Outer evaluation**
  - $c\langle 1\rangle \mid c(x).d\langle x+1\rangle \mid d\langle 2\rangle \mid d(y).\ e\langle y+1\rangle$
  - $c\langle 2\rangle \mid c(x).d\langle x+1\rangle \mid d\langle 2\rangle \mid d(y).\ e\langle y+1\rangle$

  Each with prob ½

- **Communication**
  - $c\langle 1\rangle \mid c(x).d\langle x+1\rangle \mid d\langle 2\rangle \mid d(y).\ e\langle y+1\rangle$

Choose according to probabilistic scheduler

Complexity

- Bound on number of communications
  - Count total number of inputs, multiplying by $q(|n|)$ to account for bounded replication $!_{q(|n|)}P$
- Bound on term evaluation
  - Closed term T is evaluated in time $q_T(|n|)$
- Bound on time for each communication step
  - Example: $c\langle m\rangle \mid c(x).P \;\to\; [m/x]P$
    - Bound on size of m; previous steps preserve # of x occurrences
- For each closed process P, there is a polynomial q(x) such that for all n, all probabilistic poly-time schedulers, evaluation of P halts in time $q(|n|)$

How To Define Process Equivalence?

- Intuition: P and Q are equivalent if no test by any context can distinguish them
  - $|\ \text{Prob}\{\ C[P] \rightarrow \text{"yes"}\ \} - \text{Prob}\{\ C[Q] \rightarrow \text{"yes"}\ \}\ | < \varepsilon$
- How do we choose ε?
  - – Less than 1/2, 1/4, ... ? (not an equivalence relation)
  - – Vanishingly small? As a function of what?
- Solution: asymptotic form of process equivalence
  - Use security parameter (*e.g.*, key length)
  - Protocol is a family $\{\ P_n\ \}_{n>0}$ indexed by key length

Probabilistic Observat'l Equivalence

- Asymptotic equivalence within f
  - Families of processes $\{ P_n \}_{n>0}$ $\{ Q_n \}_{n>0}$
  - Family of contexts $\{ C_n \}_{n>0}$
  - $P \approx_f Q$ if $\forall$ context C[ ]. $\forall$ observation v. $\exists n_0$. $\forall n > n_0$
    $| \mathrm{Prob}(C_n[P_n] \rightarrow v) - \mathrm{Prob}(C_n[Q_n] \rightarrow v) | < f(n)$
- Asymptotic polynomial indistinguishability
  - $P \approx Q$ if $P \approx_f Q$ for every $f(n) = 1/p(n)$ where $p(n)$ is a polynomial function of n

Probabilistic Bisimulation

[van Glabbeek, Smolka, and Steffen]

- Labeled transition system
  - Evaluate process in a "maximally benevolent context"
  - Process may read any input on public channel or send output even if no matching input exists in process
  - Label with numbers "resembling probabilities"
- Bisimulation relation
  - If $P \sim Q$ and $P \xrightarrow{r} P'$, then exists Q' such that $Q \xrightarrow{r} Q'$ and $P' \sim Q'$, and vice versa
- Strong form of probalistic equivalence
  - Implies probabilistic observational equivalence, but not vice versa

Provable Equivalences (1)

Assume scheduler is stable under bisimulation

- $P \sim Q \;\Rightarrow\; C[P] \sim C[Q]$
- $P \sim Q \;\Rightarrow\; P \approx Q$
- $P \mid (Q \mid R) \approx (P \mid Q) \mid R$
- $P \mid Q \approx Q \mid P$
- $P \mid 0 \approx P$

Provable Equivalences (2)

- $P \approx \upsilon c. (c\langle T\rangle \mid c(x).P)$ if $x \notin FV(P)$
- $P\{a/x\} \approx \upsilon c.(c\langle a\rangle \mid c(x).P)$ if bandwidth of c large enough
- $P \approx 0$ if no public channels in P
- $P \approx Q \Rightarrow P\{d/c\} \approx Q\{d/c\}$ if c, d have the same bandwidth, d is fresh
- $c\langle T\rangle \approx c\langle T'\rangle$ if $Prob[T \to a] = Prob[T' \to a]$ for all a

Connection with Cryptography

- Can use probabilistic observational equivalence in process calculus to carry out proofs of protocol security
- Example: semantic security of ElGamal public-key cryptosystem is equivalent to Decisional Diffie-Hellman
- Reminder: semantic security is indistinguishability of encryptions
  - $enc_k(m)$ is indistinguishable from $enc_k(m')$

Review: Decisional Diffie-Hellman

n is security parameter (*e.g.,* key length)

G_n is cyclic group of prime order p,

length of p is roughly n,

g is generator of G_n

For random a, b, c $\in$ {0, ... , p-1}

$$\langle g^a, g^b, g^{ab} \rangle \approx \langle g^a, g^b, g^c \rangle$$

ElGamal Cryptosystem

n is security parameter (*e.g.,* key length)

G_n is cyclic group of prime order p,

length of p is roughly n, g is generator of G_n

- **Keys**
  - Private key = $\langle g, x \rangle$, public key = $\langle g, g^x \rangle$
- **Encryption** of $m \in G_n$ is $\langle g^k, m \cdot (g^x)^k \rangle$
  - $k \in \{0, . . . , p\text{-}1\}$ is random
- **Decryption** of $\langle v, w \rangle$ is $w \cdot (v^x)^{-1}$
  - For $v=g^k$, $w=m \cdot (g^x)^k$ get $w \cdot (v^x)^{-1} = m \cdot g^{xk}/g^{kx} = m$

DDH ⇒ Semantic Security of ElGamal

- Start with $\langle g^a, g^b, g^{ab} \rangle \approx \langle g^a, g^b, g^c \rangle$ (random a,b,c)
- Build up statement of semantic security from this
  - $in(c, \langle x,y \rangle).out(c, \langle g^k, m \cdot g^{xk} \rangle) \approx$ $in(c, \langle x,y \rangle).out(c, \langle g^k, n \cdot g^{xk} \rangle)$

    Encryption of m is observationally equivalent to encryption of n
- Use structural transformations
  - E.g., $out(c,T(r)) \approx out(c,U(r))$ (any random r)

    implies $in(c,x).out(c,T(x)) \approx in(c,x).out(c,U(x))$
- Use domain-specific axioms
  - E.g., $out(c, \langle g^a,g^b,g^{ab} \rangle) \approx out(c, \langle g^a,g^b,g^c \rangle)$ implies

    $out(c, \langle g^a,g^b,m \cdot g^{ab} \rangle) \approx out(c, \langle g^a,g^b,m \cdot g^c \rangle)$ (any M)

Semantic Security of ElGamal $\Rightarrow$ DDH

- Harder direction: “break down” vs. “build up”
  - Want to go from

    $in(c,\langle x,y \rangle).out(c,\langle g^k, m \cdot g^{xk} \rangle) \approx in(c, \langle x,y \rangle).out(c,\langle g^k, n \cdot g^{xk} \rangle)$

    to $\langle g^x, g^k, g^{kx} \rangle \approx \langle g^x, g^k, g^c \rangle$
- Main idea: if m=1, then we essentially have DDH
- Proof “constructs” a DDH tuple
  - Hide all public channels except output challenge
  - Set the message to 1
- Need structural rule equating a process with the term simulating the process
  - Special case: process with 1 public output