CS 388H Introduction to Cryptography 26-Aug-2009

Lecture 1: Class Introduction

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Class Overview

This course reviews the foundations of Cryptography and will cover topics such as formal notions of security, encryption, signatures, complexity assumptions, zero knowledge, and multi-party computation.

Most of the material will be based on "Introduction to Modern Cryptography" by Katz and Lindell. "Foundations of Cryptography: Volume I" is optional and more theoretical.

See course syllabus for grading policies and course schedule.

2 Foundations of Cryptography

1. Crypto concepts: public key crypto, zero knowledge, signatures
2. Foundation approach: this course will emphasize precision and rigor

   For example: What does it mean if someone says they use a cryptographic hash function? Does it imply the system is secure?

   Two notions of what makes a hash function cryptographic:

   (a) Hard to invert: Given y, hard to find an x st $f(x) = y$.
   (b) Hard to find collisions: Given f, hard to find $x_1 \neq x_2$ st $f(x_1) = f(x_2)$.

   Are these different?

   How would you show that hardness of invertability doesn't imply hardness of finding collisions?

   By counterexample: Create a function that is hard to invert but easy to find collisions. A good start is to consider a compressing function like $f : \{0,1\}^{2n} \rightarrow \{0,1\}^n$, which will necessarily have collisions.

   Now suppose some $\tilde{f} : \{0,1\}^{2n-1} \rightarrow \{0,1\}^n$ is hard to invert (by some unstated definition of "hard to invert"). We don't need assumptions about whether it is collision-resistant.

For $x \in \{0,1\}^{2n}$, let x' be the last $2n-1$ bits. Now define $f : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ where $f(x) = \tilde{f}(x')$.

f is a counterexample to the claim that for any function that is hard to invert, it is hard to find collisions because:

- f easy to invert would contradict the assumption that $\tilde{f}$ is hard to invert, and
- it is easy to find collisions with f: for any $t \in \{0,1\}^{2n-1}$, $t_0 = 0|t$ collides with $t_1 = 1|t$.

3. Foundational underpinnings: number theory, general assumptions

   Why do we generally prove statements that are implications rather than absolutes? (eg If factorization is hard, then this system exhibits some property.) Proofs of absolute statements often reduce to a proof of $P \neq NP$.

3 Perfect Secrecy: Too good to be true?

Let $\mathcal{C}$ be the cipher text space, let $\mathcal{K}$ be the key space, and let $\mathcal{M}$ be the message space. A basic encryption system has three algorithms (may be randomized):

1. Gen $\rightarrow k \in \mathcal{K}$
2. Enc$(m,k) \rightarrow c \in \mathcal{C}$
3. Dec$(c,k) \rightarrow m \in \mathcal{M}$

What is perfect secrecy? Let $C =$Enc(m,k) for some $m \in \mathcal{M}$ and $k \in \mathcal{K}$. A system exhibits perfect secrecy if $Pr[C = c|m = m_0] = Pr[C = c|m = m_1]$ for all $m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}$ over the distribution of $\mathcal{K}$ specified by Gen (actual value of k is unknown to attacker).

3.1 One-Time Pad Encryption

Let $\mathcal{M}, \mathcal{C}, \mathcal{K} = \{0,1\}^l$. The one-time pad is as follows:

1. Gen $\rightarrow k$ chosen uniformly at random from $\{0,1\}^l$
2. Enc$(m,k) = m \oplus k = c$
3. Dec$(c,k) = c \oplus k = m$

Correctness is established by verifying that $(m \oplus k) \oplus k = m$.

Perfect secrecy is established by showing that $Pr[C = c|m] = (\frac{1}{2})^l$ for all $m \in \mathcal{M}$, $c \in \mathcal{C}$.

3.2 Limitations

The one-time pad strategy is limited in that even if Adam and Brent get a chance to agree on some $k \in \mathcal{K} = \{0,1\}^l$, the message that can be sent can be length at most l.

This limitation that messages cannot be longer than keys generalizes to any perfectly secret system. Suppose $|\mathcal{M}| > |\mathcal{K}|$. (Note that there are more unique messages than unique keys if and only if the messages are longer than the keys.)

1. Let c be a cipher text that occurs with non-zero probability.
2. Then $\exists \tilde{m}, \tilde{k}$ st $\text{Enc}(\tilde{m}, \tilde{k}) = c$ with non-zero probability.
3. If the decryption algorithm is correct, then at most $|\mathcal{K}|$ messages m are such that $\text{Enc}(m, k) = c$. (Otherwise, there would not be enough keys to uniquely decrypt the cipher text.)
4. Then (because $|\mathcal{M}| > |\mathcal{K}|$) $\exists m'$ st $\forall k \in \mathcal{K}$ $\text{Enc}(m', k) \neq c$.

In the definition for perfect secrecy, use $m_0 = \tilde{m}$ and $m_1 = m'$. $Pr[C = c|\tilde{m}] > 0 \neq Pr[C = c|m'] = 0$, which violates perfect secrecy. Therefore, no system in which keys are shorter than messages can be perfectly secret.

4 Handouts

Three handouts were given in class:

1. CS 388H Course syllabus (2 pages)
2. Very basic number theory fact sheet, Part I (4 pages)
3. Basic number theory fact sheet, Part II (4 pages)

CS 388H Introduction to Cryptography 31-Aug-2009

Lecture 2: Introduction to Number Theory

Instructor: Brent Waters *TA: Sara Krehbiel*

Recall the major limitation of perfect secrecy that messages can be no longer than keys. Instead, we verify the security of more practical encryption systems with complexity assertions, often substantiated by number theory. To do that, we want to be able to make sense of a description like: "Group G of prime order p with generator g".

Note: Most of the material from this lecture can be found in the "Very basic number theory fact sheet (Part I)" handout passed out at the first lecture. Sections 7.1-7.3 of Katz and Lindell is another reference.

1 Announcements

1. Professor office hours: M 12-1, ACES 3.438
2. TA office hours: W 2-3, ENS basement, desk 1; F 11-12, ENS basement, desk 4
3. Number theory review: Instead of regular OH this Friday (9/4), Sara will hold a number theory review in ACES 3.116. We will cover the main points from this week's lectures and the number theory handouts, and we'll spend the bulk of the time going over points of confusion. Please come with questions.
4. Labor day: No class or OH next Monday (9/7)

2 Basic Number Theory Facts

- $\mathbb{Z}$ denotes the set of integers
- Fundamental theorem of arithmetic: Any integer a can be uniquely expressed as a product of primes. Ie $a = \prod_i p_i^{e_i}$ with p_i a prime and e_i a positive integer for all i.
- $a|b$ means a divides b ($\exists c \in \mathbb{Z}$ st $c \cdot a = b$).
- For any positive integers a and b, a can be uniquely written as $a = q \cdot b + r$ with $q, r \in \mathbb{Z}$ and $0 \leq r < b$.
- Greatest common divisors: Define $gcd(a, b)$ to be the largest integer d st $d|a$ and $d|b$. Note that $gcd(a, b) \geq 1$ $\forall a, b \in \mathbb{Z}$.

3 Modular Arithmetic

Let $a =_p b$ denote that $a = b \bmod p$. In the equations below, let $q \in \mathbb{Z}$ and $0 \leq r < p$.

- $\mathbb{Z}_p = \{0, 1, ..., p-1\}$ for some prime p.
  Note: Small primes are used as instructive examples, but the prime numbers typically used in cryptography are on the order of 300 digits (1024 bits).
- Addition: $m + g =_p r$ for r st $m + g = qp + r$. Eg 5 + 6 mod 7 = 4.
- Multiplication: $m \cdot g =_p r$ for r st $m \cdot g = qp + r$. Eg $5 \cdot 6$ mod 7 = 2.
- Exponentiation: $m^g =_p r$ for r st $m^g = qp + r$. Eg 5^6 mod 7 = 1.

Why can that particular last problem be determined without actually computing 5^6?

4 Fermat's Little Theorem

Theorem 4.1 *For all $g \neq 0$ in $\mathbb{Z}_p$, g is such that $g^{p-1} =_p 1$.*

Proof. Consider the set $S = \{g, 2g, ..., (p-1)g\}$. Assume that the S contains fewer than $p-1$ distinct elements. In other words, there exist some $r, s \in \mathbb{Z}_p^*$ with $r \neq s$ and $rg =_p sg$. Then $(r-s)g =_p 0$. Even in modular arithmetic, $ab = 0$ iff $a = 0$ or $b = 0$. $g \neq 0$ by definition, so it must be that $r - s =_p 0$. But this contradicts $r \neq s$, establishing that all $p-1$ elements of S are distinct.

$g \neq 0$ means each element of S is in $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$, which is just $\mathbb{Z}_p \backslash \{0\}$, so S constitutes a reordering of $\mathbb{Z}_p^*$. The product of the elements of S should equal the product of the elements of $\mathbb{Z}_p^*$:

$$\begin{aligned} \prod_{i=1}^{p-1} ig &=_p \textstyle\prod_{i=1}^{p-1} i \\ (p-1)!g^{p-1} &=_p (p-1)! \\ g^{p-1} &= 1 \end{aligned}$$

□

This theorem allows you to more quickly compute exponents: $5^{12} = 5^6 \cdot 5^6 = 1$ mod 7.

5 Inverses, Generators, and Orders

- Definition: The inverse of $x \in \mathbb{Z}_p$ is $a \in \mathbb{Z}_p$ st $a \cdot x =_p 1$ and is denoted x^{-1}.
- Inversion algorithm: $x^{-1} =_p x^{p-2}$ by Fermat's little theorem ($x^{p-1} = x^{p-2} \cdot x =_p 1$). The Euclidean algorithm also provides inverses and will be discussed later.
- $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$ is the set of invertible elements in $\mathbb{Z}_p$.

- $\mathbb{Z}_p^*$ is a cyclic group. In general, group G is cyclic iff there exists a generator $g \in G$ st $G = \{1, g, g^2, g^3, ..., g^{|G|-1}\}$.
- Not every element of $\mathbb{Z}_p^*$ is a generator. (Note that g is a generator in a different sense here than in the proof of Fermat's little theorem – multiplicative vs additive.) Observe that 3 generates $\mathbb{Z}_7^*$ because $< 3 >= \{1, 3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5\}$ but 2 does not because $< 2 >= \{1, 2, 4, 1, 2, 4\}$. Instead, $< 2 >$ is called a subgroup of $\mathbb{Z}_p^*$.
- Define the order of $g \in \mathbb{Z}_p^*$, denoted $\text{ord}_p(g)$, to be the smallest $a \in \mathbb{Z}_p^*$ st $g^a =_p 1$.
- g generates $\mathbb{Z}_p^*$ iff $\text{ord}_p(g) = p - 1$.
- Lagrange's theorem: $\text{ord}_p(g) | p - 1$ for all $g \in \mathbb{Z}_p^*$. (Try to prove this on your own.)

Later we will see that when encrypting, it is desirable to have primes where $p - 1 = 2q$, where q is a large prime. Such p are called strong primes because it is more secure to have at least one large factor of $p - 1$.

6 Quadratic Residues

- $y \in \mathbb{Z}_p$ is a square root of $x \in \mathbb{Z}_p$ iff $y^2 =_p x$.
- An element has either 0 or 2 square roots in $\mathbb{Z}_p$. If a is a square root of x, so is $-a$.
- $x \in \mathbb{Z}_p^*$ is called a quadratic residue (QR) iff if has a square root in $\mathbb{Z}_p$.
- The Legendre symbol for $x \in \mathbb{Z}_p$ is denoted $(\frac{x}{p})$ and is 1 if x is a QR in $\mathbb{Z}_p^*$, -1 if x is not a QR in $\mathbb{Z}_p^*$, and 0 if $x =_p 0$.
- Euler's theorem: $x \in \mathbb{Z}_p$ is a QR iff $x^{(p-1)/2} =_p 1$. Consequences:
  1. $(\frac{x}{p}) =_p x^{(p-1)/2}$
  2. About half the elements of $\mathbb{Z}_p$ are QRs.

7 Arithmetic Computing in $\mathbb{Z}_p$

How many unit operations are needed for each of the following computations? Note that the binary length of p is $\log p$. Recall that $\log p$ is typically around 1024.

- Addition is one bit operation per digit: $O(\log p)$
- Multiplication is one long addition per digit: $O((\log p)^2)$
- Exponentiation (the naive approach) is p long multiplications: $O(p(\log p)^2)$.

How can we exponentiate more efficiently?

7.1 Repeated squaring algorithm for computing exponents

Consider y as the concatenation of its bits: $y = y_{\lceil \log y \rceil} | y_{\lceil \log y \rceil - 1} | ... | y_2 | y_1$. Take the following steps to compute $x^y \bmod p$:

1. Compute $O(\log y)$ multiplications of complexity $O((\log x)^2)$: $x, x^2, x^4, x^8, ..., x^{2^{\lceil \log y \rceil}}$.
2. Compute $O(\log y)$ multiplications of complexity $O((\log x)^2)$ (exponentiation is trivial): $x^y = x^{y_1} \cdot (x^2)^{y_2} \cdot ... \cdot (x^{2^{\lceil \log y \rceil}})^{y_{\lceil \log y \rceil}}$.

This algorithm is $O(\log y (\log x)^2) = O((\log p)^3)$ compared to the naive approach giving $O(p(\log p)^2)$.

8 Discrete Log Problem

Problem: Given a security parameter λ, choose a prime p of length λ and a generator $g \in \mathbb{Z}_p^*$. Choose a random $a \in \mathbb{Z}_p^*$ and compute $h =_p g^a$. Keep a secret, but reveal p, g, and h. The attacker's goal is to determine a.

Assumption 8.1 *The discrete log problem cannot be solved by any algorithm that satisfies both of the following conditions:*

1. *Algorithm runs in polynomial time.*
2. *Algorithm is correct with non-negligable probability.*

CS 388H Introduction to Cryptography 2-Sept-2009

Lecture 3: More Number Theory

Instructor: Brent Waters *TA: Sara Krehbiel*

Note: Most of the new material from this lecture can be found in the second of two number theory handouts passed out at the first lecture. Not everything in the handouts was covered in lecture. Sections 7.1-7.3 of the text is another number theory reference.

1 Previous Lecture: Review, Clarifications, Extensions

1.1 Complexity of arithmetic mod p

- Addition: $\log p$
- Multiplication: $(\log p)^2$
- Exponentiation: $(\log p)^3$

The following is a (perhaps clearer) restatement of the fast exponentiation algorithm and its correctness for integers of approximate size $p = 2^n$:

Let $y = 2^0 y_0 + 2^1 y_1 + ... + 2^n y_n$ where y is n binary digits ($y < 2^{n+1}$) with y_0 the least significant and y_n the most significant digit.

Then $x^y = x^{2^0 y_0 + 2^1 y_1 + ... + 2^n y_n} = (x^{2^0})^{y_0} \cdot ... \cdot (x^{2^n})^{y_n}$.

Computing the x^{2^i} requires n multiplications, computing $(x^{2^i})^{y_i}$ is free since y_i is 0 or 1, and taking the final product is another n multiplications. The total complexity is $O(n(\log 2^n)^2) = O((\log p)^3)$.

1.2 Applying Fermat's little theorem to exponentiation

We can reduce the size of the x^y computation before applying the above algorithm if $y >> p$. In particular, if $r =_{p-1} y$, then $x^y =_p x^r$. Recall Fermat's little theorem: $x^{p-1} =_p 1$ for any $x \in \mathbb{Z}_p^*$. Then we see that $x^y = x^{k(p-1)+r} = (x^{p-1})^k \cdot x^r =_p 1^k x^r = x^r$.

Later this lecture, we will generalize Fermat's little theorem for composite moduli.

1.3 Orders

Recall the definition of the order of some element $g \in \mathbb{Z}_p^*$ and Lagrange's related theorem:

- **Definition 1.1** *$ord_p(g)$ is the smallest q such that $g^q =_p 1$.*
- **Theorem 1.2** *$ord_p(g) | p - 1$.*

Note that conceptualizing groups as integers and multiplication modulo some prime is convenient and concrete, but the concepts of groups, generators, and orders are more general. In fact, there exist attacks for some cryptographic systems based on $\mathbb{Z}_p$ subgroups that fail with elliptic curve groups in which there exist analogous concepts of generators and orders.

1.4 Hard problems in $\mathbb{Z}_p$

1. (Review) Discrete log problem: Given g and $h =_p g^a$, compute a. The assumption is that no poly-time algorithm can correctly output a with better than negligible probability.
2. (New) Computational Diffie-Helman problem: Given g, g^a, and g^b, compute g^{ab}. A solution to the discrete log problem could be used to solve this problem.

2 Inverses Modulo Composite N

- The inverse of $x \in \mathbb{Z}_N$ is the element $a \in \mathbb{Z}_N$ st $ax =_N 1$.
- a and b are said to be relatively prime iff $\gcd(a, b) = 1$.
- x has an inverse in $\mathbb{Z}_N$ iff x and N are relatively prime.
- Let $\mathbb{Z}_N^* = \{a | \gcd(a, n) = 1\}$, the set of invertible elements in $\mathbb{Z}_N$.
- Euclid's algorithm (will be covered at Friday number theory review) computes x^{-1} for $x \in \mathbb{Z}_N^*$. It is more efficient than computing x^{p-2} in the special case of prime N.

3 Euler's Totient Function

Definition 3.1 *$\varphi(N) = |\mathbb{Z}_N^*|$, the number of invertible elements in $\mathbb{Z}_N$.*

Fact 3.2 *If N's prime factorization is given by $N = p_1^{e_1} \cdot \ldots \cdot p_n^{e_n}$, then $\varphi(N) = N \prod_{i=1}^{n}(1 - \frac{1}{p_i})$.*

Remark 3.3 *$\varphi(p) = p - 1$ for prime p.*

Theorem 3.4 *Euler's theorem: $a^{\varphi(N)} =_N 1$ for all $a \in \mathbb{Z}_N^*$.*

Remark 3.5 *Euler's theorem generalizes Fermat's little theorem.*

4 Chinese Remainder Theorem (CRT)

Theorem 4.1 *Let $N = pq$ with p, q relatively prime. Given $r_1 \in \mathbb{Z}_p$ and $r_2 \in \mathbb{Z}_q$, there exists a unique, efficiently computable element $s \in \mathbb{Z}_N$ such that $s =_p r_1$ and $s =_q r_2$.*

Remark 4.2 *The CRT establishes the existence and uniqueness of the inverse f^{-1} of the function $f : \mathbb{Z}_N \to (\mathbb{Z}_p, \mathbb{Z}_q)$ define as $f(s) = (r_1, r_2)$ where $s =_p r_1$ and $s =_q r_2$.*

Claim 4.3 *$f(s \cdot t) = f(s) \cdot f(t)$.*

Proof. Let $s = k_1 p + s_1 = k_2 q + s_2$, $t = c_1 p + t_1 = c_2 q + t_2$.
Then $s \cdot t = (k_1 c_1 p + k_1 t_1 + c_1 s_1)p + s_1 t_1 = (k_2 c_2 q + k_2 t_2 + c_2 s_2)q + s_2 t_2$.
So $f(s \cdot t) = (s_1 t_1, s_2 t_2) = (s_1, s_2) \cdot (t_1, t_2) = f(s) \cdot f(t)$. □

Remark 4.4 *$s \cdot t \bmod N$ can be computed more efficiently as $f^{-1}(f(s) \cdot f(t))$. Assume f and f^{-1} are free. For p, q of approximately equal size, computing $s_1 \cdot t_1$ and $s_2 \cdot t_2$ costs $2(\log p)^2$. Directly multiplying s and t costs $(\log(p^2))^2 = 4(\log p)^2$. There are more dramatic efficiency gains when we use CRT to compute exponents.*

5 Squares in $\mathbb{Z}_N$

Claim 5.1 *A element $s \in \mathbb{Z}_N$ is a QR iff it is a QR in $\mathbb{Z}_p$ and it is a QR in $\mathbb{Z}_q$.*

Corollary 5.2 *For prime p, q, half the elements of $\mathbb{Z}_p$ and half the elements of $\mathbb{Z}_q$ are QR, so the number of QR in $\mathbb{Z}_N$ is $\frac{p-1}{2} \cdot \frac{q-1}{2} = \frac{\varphi(N)}{4}$.*

Definition 5.3 *Jacobi symbol: For $x \in \mathbb{Z}_N$ define $\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right) \cdot \left(\frac{x}{q}\right)$, where the RHS is the product of Legendre symbols.*

Remark 5.4 *The Jacobi symbol can be computed for $x \in \mathbb{Z}_N$ without knowing N's factors.*

6 Hard Problems in $\mathbb{Z}_N$

Note: That factoring is hard is the weakest assumption; if this assumption fails, so do all the others.

1. Factoring assumption: Given $N = pq$, it is hard to output p and q. (A solution would immediately give $\varphi(N)$.)
2. Strong RSA assumption: Given $N = pq$, an attacker chooses a public $e \geq 3$ and the system produces some $h = a^e$. Finding a is hard.
3. QR assumption: Given an element x that either is a square in $\mathbb{Z}_N^*$ or is not a square in $\mathbb{Z}_N^*$, determine whether it is a square. Since a random guess is correct with probability .5, we call an algorithm's advantage ε such that it answers correctly with probability $.5 + \varepsilon$. No poly-time algorithm has a non-negligible advantage.

CS 388H Introduction to Cryptography 4-Sept-2009

Number Theory Review

Instructor: Brent Waters *TA: Sara Krehbiel*

This review covered most of the topics discussed in lectures 2 and 3. This material can be found in the class lecture notes, the two number theory handouts, and sections 7.1-7.3 of the text, so it will not be repeated here beyond listing the topics covered. The only new material that was covered was Euclid's extended algorithm for finding inverses in $\mathbb{Z}_N^*$, which is explained here. Points of clarity for questions that were not answered in the review are also listed here.

1 Topics

- Fundamental theorem of arithmetic
- Fermat's little theorem
- Modular arithmetic
  - Practice
  - Complexity results
  - Square and multiply algorithm for fast exponentiation
  - Using Fermat's little theorem to further optimize exponentiation
- Inverses, generators, and orders
- Lagrange's theorem
- Inverstion in $\mathbb{Z}_p$ using Fermat's little theorem
- Inverstion in $\mathbb{Z}_N$ using extended Euclidean algorithm (see section 2)
- Euler's totient function, its value wrt to N's prime factorization, and Euler's theorem
- Chinese remainder theorem
- Quadratic residues: Legendre symbols in $\mathbb{Z}_p$; Jacobi symbols in $\mathbb{Z}_N$
- Hard problems:
  - In $\mathbb{Z}_p$: Discrete log and Diffie-Helman
  - In $\mathbb{Z}_N$: Factoring, RSA, and QR assumptions

2 Extended Euclidean Algorithm for Inverses in $\mathbb{Z}_N$

The basic Euclidean algorithm computes $gcd(x, y)$ (in $O(\log x)$ time) using the fact that $gcd(x, y) = gcd(x \bmod y, y)$. Verify this by noting that if d is the largest integer that divides x and y it also must be the largest integer that divides y and r, where $x = qy + r$ (assume for simplicity that $x \geq y$. The Euclidean algorithm is stated recursively as follows:

Algorithm 1 Euclidean algorithm for computing the GCD of two integers

```
if y=0 then
  return x
else
  return GCD(y, r)
end if
```

Extend Euclid's algorithm to keep track of integers a and b such that $gcd(x, y) = ax+by$:

Algorithm 2 Extended Euclidean algorithm, returns $\{d, a, b\}$ where $gcd(x, y) = d = ax+by$

```
if y=0 then
  return {x, 1, 0}
else
  {d, a, b} := GCD-EXT(y, r)
  return {d, b, a-bq}
end if
```

Proof. Correctness of the extended Euclidean algorithm is established by induction.
Base case: If $y = 0$, then GCD$(x, y) = x$ and $1x + 0y = x$ clearly holds.
Inductive hypothesis: For some k, the algorithm is correct for all x, y with $y \leq k$. The recursive call establishes that $ay + br = d$. Then $d = ay - bqy + br + bqy = (a - bq)y + bx$. This establishes the correctness of the final returned value and completes the proof by induction. $\square$

2.1 Using the algorithm to compute inverses

Recall that $x^{-1} \in \mathbb{Z}_N$ exists iff gcd$(N, x) = 1$. If the extended Euclidean algorithm returns GCD-EXT$(N, x) = \{1, a, b\}$, this means $aN + bx = 1$, so $bx =_N 1$ and b satisfies the definition for $x^{-1} \in \mathbb{Z}_N$.

3 Points of Clarification

3.1 Lagrange's theorem

An informal motivation for the proof of Lagrange's theorem was given in the review session. I said I would include a more formal explanation in these notes, but this has been reserved as an exercise for the first homework.

3.2 Proof of the statement: x is a quadratic residue in $\mathbb{Z}_p$ iff $x^{(p-1)/2} =_p 1$

One direction of this proof was given. Namely, if there exists some y such that $y^2 = x$ (alternatively, $x^{1/2} = y$), we know by Fermat's little theorem that $y^{p-1} = 1$, so $(x^{1/2})^{p-1} = 1$, establishing that $x^{(p-1)/2} = 1$.

The proof of the other direction ($x^{(p-1)/2} =_p 1 \rightarrow x$ is a QR in $\mathbb{Z}_p$) is as follows: Let g be a generator of $\mathbb{Z}_p$. Then any x st $x^{(p-1)/2} =_p 1$ can be written as g^i for some i. Then $g^{i(p-1)/2} = 1$. Because g is a generator, its order is $p-1$, so by Fermat's little theorem, $i/2$ must be an integer. Then $k = g^{i/2}$ is a square root of x, so x is a QR.

CS 388H Introduction to Cryptography 9-Sept-2009

Lecture 4: Collision Resistant Hash Functions

Instructor: Brent Waters *TA: Sara Krehbiel*

Note: Most of the new material from this lecture can be found in the second of two number theory handouts passed out at the first lecture. Not everything in the handouts was covered in lecture. Sections 7.1-7.3 of the text is another number theory reference.

1 Handouts

Homework 1 was passed out today and is due Wednesday, September 23.

2 Terminology

We want to build terminology so we can rigorously talk about things like the hardness of problems and reason about statements like "all efficient algorithms have at most a negligible advantage."

- λ denotes a security parameter, which determines, for example, the size of messages accepted by an encryption system.
- Efficient algorithms are algorithms with polynomial run time. The class of polynomials functions is nice for several reasons, most importantly because it is closed under addition, multiplication, and function composition.
- Negligible functions are all functions $g(x)$ such that for any polynomial $p(x)$, there exists a z such that $x > z$ implies $g(x) < 1/p(x)$.
  - Negligible functions definitely approach zero as x approaches infinity, but this condition is not strong enough.
  - The condition that negligible functions are of the form $1/poly(x)$ is also not strong enough because you could run an attack with this type of success probability a poly number of times and the result would be an algorithm with non-negligible probability of success (if you could identify successes from failures).
  - It is necessary that a polynomial number of repetitions of a negligible algorithm also be negligible, but this is also not strong enough.
  - It is sufficient that the function be no greater than $1/2^{poly(x)}$. The definition as stated is slightly more permissive, but checking that an algorithm's success probability is less than $1/2^x$ is a good baseline.

3 Collision Resistant Hash Functions (CRHFs)

The process of studying hash functions will be organized as follows:

1. Build intuitions
2. Specify definitions
3. Construct function
4. Prove properties

3.1 Intuition

What is collision resistance? No or few collisions? We want it to be hard to find to messages that produce the same hash value. But then the identity function is perfectly collision resistant, and it doesn't add any security. For security, we want the function to shrink/compress the key.

Well-known cryptographic hash functions include MD4, MD5, SHA, and WHIRLPOOL. They are widely used for their speed, but not theoretically (and eventually not practically) secure: MD4 can be attached by hand, MD5 can be attacked with a laptop, etc. Part of this vulnerability is due to the fact that they have no (or few, discrete) security parameters.

In contrast, we will base our hash functions on theoretical problems that are believed to be hard, such as the discrete log problem. These constructions tend to produce less efficient functions, but they enable us to prove statements about security like "if the discrete log problem is hard to solve, this function is collision resistant."

3.2 Definitions

Keyed hash function H is a CRHF family if:

- Gen$(\lambda) = K$. A key (needn't be secret) is generated based on the security parameter.
- $H_K : \{0,1\}^m \to \{0,1\}^l$ where m and l are functions of λ with $m > l$ and H is efficient.
- For every efficient attack algorithm A, $Pr[A(K) \to \{m_0, m_1\} : H_K(m_0) = H_K(m_1)]$ is negligible. (Ie efficient attack algorithms are only successful at producing collisions with negligible probability.)

3.3 Construction

- Gen(λ) produces a key $K = (G, p, g, h)$ where p is a prime with $\log p \approx \lambda$, G is the set associated with a multiplicative group of order p, and g and h are two (random) generators of G.
- Define $H_K : \mathbb{Z}_p \times \mathbb{Z}_p \to G$ as $H_K(x_a.x_b) = g^{x_a} h^{x_b}$. Note: We must assume there is an efficient representation of G's elements that is smaller than $2 \log p$ to ensure that H_K is compressing.
- To establish security, we next want a proof that if the discrete log problem is hard, H is collision-resistant.

3.4 Proof

Theorem 3.1 *If there exists an efficient algorithm A that produces collisions in H with non-neglibile probability ε, then there exists an efficient algorithm B that solves the discrete log problem with non-negligible probability ε'.*

High level motivation for the proof: We show that A's existence implies B's existence if we can construct B from any hypothetical A by converting a DL challenge into a collision-finding challenge, feeding the challenge to A, and converting A's result into a solution to the DL challenge.

Proof. Assume that an efficient and non-negligibly successful attack algorithm A exists. Construct B as follows:

1. Accept a DL challenge: (G, p, g, h) with $h = g^a$
2. Give A the key: $K = (G, p, g, h)$
3. A runs; take A's output: $m_0 = (x_a, x_b)$ and $m_1 = (y_a, y_b)$ that collide in H_K
4. If $H_K(m_0) = H_K(m_1)$, $m_0 \neq m_1$ (ie if A was successful), continue; else fail and quit
5. Compute and return $a = (x_a - y_a)(y_b - x_b)^{-1}$ as the solution to the DL challenge

To check whether this algorithm constitutes a B that satisfies the requirements for a DL problem solver, check for correctness, non-negligible advantage, and polynomial run time.

1. If A is correct, then
   - $H(m_0) = g^{x_a} h^{x_b} = g^{y_a} h^{y_b} = H(m_1)$
   - $g^{x_a} g^{a x_b} = g^{y_a} g^{a y_b}$ (we know $h = g^a$ for some a, so assume this and solve for a)
   - $g^{a(x_b - y_b)} = g^{y_a - x_a}$
   - $a(x_b - y_b) = y_a - x_a$ (because the group is multiplicative and g is a generator)
   - $a = (x_a - y_a)(y_b - x_b)^{-1}$
2. A is succeeds with an ε advantage, and we've just illustrated that B succeeds if and only if A succeeds, so B's advantage is ε, which we've specified is non-negligible.
3. Steps 1-2 take constant time. Step 3 is the run time of A, which is give to be polynomial. Step 4 requires two exponentiations, which we've seen can be done in polynomial time. Step 5 is a couple additions, an inverse computation (using Euclid's inverse algorithm), and a multiplication, all of which can be done in polynomial time. B is efficient.

This completes the proof that the H is a CRHF if the discrete log problem is hard. □

CS 388H Introduction to Cryptography 14-Sept-2009

Lecture 5: Signature Schemes

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lecture 4 Recap

In the last lecture we gave an example construction and proof of a CRHF family. In particular, we defined a hash function $H(x_a, x_b) = g^{x_a} h^{x_b}$ for generators g and h and showed that any efficient algorithm with a non-negligible advantage that would find collisions for this function could be used to construct a wrapper algorithm to solve the discrete log problem. This proved that breaking H is at least as hard as solving the discrete log problem.

1.1 A note on modular arithmetic in exponents

In our CRHF proof, we used the following assertion:

Fact 1.1 *For a multiplicative group of prime order p, $z =_p c(a+b)$ implies $(g^a g^b)^c =_p g^z$.*

2 Motivation for Signature Schemes

2.1 Commonly-used schemes

- DSA (similar to ElGamel signature scheme)
- RSA

Both are number theory-based, but build intuition more generally.

2.2 Desired properties for signatures

1. Unique
2. Not forgeable
3. Verifiable
4. Cheap (to create and to verify)

2.3 Uses

- Sensitive emails
- https/ssl (embedding session keys in protocols)
- Software updates

2.4 General framework for signature scheme

1. Signer chooses a secret key (SK) and corresponding verification key (VK)
2. Signer publishes VK
3. Signer produces a signature σ using a message m in the message space $\mathcal{M}$ and SK
4. Receiver uses m, σ, and VK to verify the identity of the sender

3 Three Algorithms for Digital Signature Schemes

1. Setup(λ) $\rightarrow$ SK, VK
2. Sign(m, SK) $\rightarrow \sigma$
3. Verify(m, VK, σ) $\rightarrow$ {true, false}

Notes:

- This constitutes a correctness definition in that Verify should return true if σ was indeed produced by Sign.
- Each algorithm should be efficient.
- Setup must be a randomized algorithm. Sign may be, as can verify in some cases.

4 Security Definition for Signing Schemes (GMR '84)

A signature scheme is said to be secure, or more specifically, existentially unforgeable under chosen message attack, if any poly-time attack algorithm A has at most a negligible advantage (Adv_A), where Adv_A is the probability of the attacker issuing a message m^* and successfully forged signature σ^*. In other words, $Adv_A = Pr[\text{Verify}(m^*, \text{VK}, \sigma^*) = \text{true}]$ in the following scenario:

1. Challenger runs Setup(λ) $\rightarrow$ SK, VK
2. Attacker gets VK and a signing oracle
3. Attacker can submit q messages m to the oracle and receive σ = Sign(m, SK), where q is a polynomial in λ
4. Attacker outputs m^*, σ^* (with m^* not previously submitted to the oracle)

Note that this signing oracle probably provides the attacker with a much greater advantage than he would receive in practice. The signing oracle results in a more conservative or stronger definition of security than one that may be a more practical model of the information available to attackers.

5 One-Way Functions

Definition 5.1 *A function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ (with n a function of security parameter λ) is called a one-way function if it is easy to compute (P(λ)) and hard to invert (NP(λ)). Formally, for any efficient A, $Pr_{x \in \{0,1\}^n}[A(y = f(x)) \rightarrow z : f(z) = f(x)] = negl(\lambda)$.*

Note that in the above example, z needn't be equal to x unless f is injective.

Notice that $f(x) = g^x$ for group generator g satisfies the definition of a one-way function assuming the hardness of the discrete log problem. (But in general one-way functions needn't be established by the hardness of number theory problems in particular.)

5.1 Signature scheme using one-way function f with only 1 message in message space

Let f be a one-way function; let the message space $\mathcal{M}$ be such that $|\mathcal{M}| = 1$ with $m = null$ the only element of $\mathcal{M}$. Define a signature scheme with the following algorithms:

1. Setup(λ):
   - Choose random x of length λ
   - Set SK = x
   - Set VK = $f(x)$
2. Sign(m, SK) = SK
3. Verify(m, VK, σ) = true if and only if $f(\sigma)$ = VK

Note that the attacker can't actually make use of the oracle since he must forge on m^* and there are no other messages in $\mathcal{M}$ that he could use to query the Sign algorithm.

This scheme only works for $|\mathcal{M}| = 1$ since the signing and verifying algorithms don't use the message at all. In practice, we want schemes that accommodate larger $\mathcal{M}$, but don't want schemes that associated a unique SK, VK pair with each possible message because it would require keeping track of many keys for a message space consisting of even just short messages.

5.2 Lamport one-time signature scheme

The Lamport signature scheme signs an n-bit message.

1. Setup(λ):
   - Choose SK = $x_{1,0}|x_{1,1}|...|x_{n,0}|x_{n,1}$ where each $x_{i,\{0,1\}}$ is a string of length λ chosen uniformly at random from all such strings
   - Choose VK = $y_{1,0} = f(x_{1,0})|y_{1,1} = f(x_{1,1})|...|y_{n,0} = f(x_{n,0})|y_{n,1} = f(x_{n,1})$
2. Sign($m = m_1|...|m_n$, SK) = $x_{1,m_1}|...|x_{n,m_n}$
3. Verify(m, σ, VK) = true iff $f(\sigma_i) = y_{i,m_i}$ for each $i = 1, ..., n$

The proof of the security of the Lamport signature scheme will be given next lecture.

CS 388H Introduction to Cryptography 16-Sept-2009

Lecture 6: Lamport One-Time Signature Schemes

Instructor: Brent Waters *TA: Sara Krehbiel*

Last class introduced the algorithms for the Lamport one-time signature scheme. We here restate the algorithms and then prove its security from the security of one-way functions.

1 Algorithms

Lamport signs n-bit messages using an SK of length $2n\lambda$ and a VK of length $2n$ times the length of the elements in the range of some OWF f.

1. Setup(λ):
    - Choose SK = $x_{1,0}|x_{1,1}|...|x_{n,0}|x_{n,1}$ where each $x_{i,\{0,1\}}$ is a string of length λ chosen uniformly at random from all such strings
    - Choose VK = $y_{1,0} = f(x_{1,0})|y_{1,1} = f(x_{1,1})|...|y_{n,0} = f(x_{n,0})|y_{n,1} = f(x_{n,1})$
2. Sign($m = m_1|...|m_n$, SK) = $x_{1,m_1}|...|x_{n,m_n}$
3. Verify(m, σ, VK) = true iff $f(\sigma_i) = y_{i,m_i}$ for each $i = 1, ..., n$

2 Proof of Security

Claim 2.1 *If there exists an efficient, one-time attack algorithm A that forges a Lamport one-time signature with non-negligible probability, then there exists some efficient B that inverts one-way functions.*

Proof. To fit this proof framework, B will receive a function f and some y^* randomly selected from the range of f from the OWF challenger. B can use A to produce a z st $f(z) = y^*$. A is a black box that takes as input some f and VK from a random-looking distribution, requests a signature for a single m', and efficiently produces with non-negligible probability some forgery σ^* on $m^* \neq m'$. With this in mind, construct B as follows:

1. B receives f and some $y^* = f(\tilde{x})$ for a random $\tilde{x}$ unknown to B.
2. Since A will request a signature for some m' and then forge on $m' \neq m*$, there will be at least one bit j such that $m^*_j \neq m'_j$. B guesses this j.
3. B then chooses the VK to publish for A. Generate the SK and VK as in the Lamport scheme with one exception: Pick $b \in \{0,1\}$ and set element $y_{j,b}$ in the VK to the y^* given by the OWF challenger. (The corresponding $x_{j,b}$ generated randomly is now meaningless.) Give VK to A.

4. Next, A requests B to sign m'. If $m'_j = b$, B does not have the corresponding SK, so B quits. Otherwise, return the components of the secret key corresponding to the bits of m' as directed by the Lamport signing scheme.

5. With non-negligible probability, A forges a signature σ^* on m^*. If $m^*_j \neq b$, B gains no information from the forgery and quits. Otherwise, $f(\sigma^*_j) = y_{j,b} = y^*$ if As forgery is correct.

6. Finally, B outputs σ^*_j as the solution to the OWF challenge.

To use this construction to establish the claim, we have to show that B is efficient and that B beats the OWF challenger with non-negligible probability. Everything B does other than waiting for A takes constant time. A is efficient, so this shows that B is efficient. B will beat the OWF challenger if all of the following happens: B picks a j such that $m'_j \neq m^*_j$, A successfully forges, and B picks b such that $m'_j \neq b$ and $m^*_j = b$. Given that A's advantage is some non-negligible ε, a lower bound on this joint probability is $\varepsilon/2n$, which is non-negligible. This completes the proof that the Lamport one-time signature scheme is at least as hard to break as the OWF problem. □

As an individual exercise, explore why this proof breaks down for $q = 2$ (ie if the Lamport attacker can request two signatures before forging).

3 Extending Lamport to Sign Larger Messages

If we use a CRHF $H : \{0,1\}^* \to \{0,1\}^n$, we can first hash the message and then use Lamport to sign the hashed message. The verifier checks that $f(\sigma_i) = y_{i,H(m)_i}$ for i 1 to n. Security is maintained by the above proof combined with the collision resistance of H.

4 Inherent Limitations and Future Direction

The obvious limitation of one-time signatures is that they can only be used one time! In the last two lectures, we have seen how to use the security of OWFs to establish the security of the Lamport scheme in signing 1-time, n-bit messages, and we just used CRHFs to extend this to 1-time security for messages of arbitrary length. Next we will show how to use tree-based techniques to construct a system that can securely sign an exponential number of arbitrarily-large messages.

CS 388H Introduction to Cryptography 21-Sept-2009

Lecture 7: Tree-Based Technique for Signing Many Messages

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lamport Recap

- What was nice about Lamport? It was based on OWFs, so there was no tricky number theory.
- What were the major drawbacks? Big keys, so a little inefficient, but more importantly it could only be used once.
- How did we show security? A reasonable Lamport attacker would break OWFs.
- What's next? Use Lamport signatures to motivate a scheme that allows us to sign more than one message without increasing key size.

2 Multi-Time from One-Time Signature Schemes

The following procedure allows us to store multiple VK, SK pairs and build multiple signatures that are verifiable via clever use of any one-time signature scheme algorithm.

1. $(\mathrm{VK}_\varepsilon, \mathrm{SK}_\varepsilon) :=$ Setup-OT
2. $(\mathrm{VK}_0, \mathrm{SK}_0) :=$ Setup-OT
3. $(\mathrm{VK}_1, \mathrm{SK}_1) :=$ Setup-OT
4. $\sigma_\varepsilon :=$ Sign-OT$(m = \mathrm{VK}_0|\mathrm{VK}_1, \mathrm{SK}_\varepsilon)$
5. Get message m_0 to sign
6. $\sigma_0 :=$ Sign-OT(m_0, SK_0)
7. Sign m_0 as $\sigma_{m_0} := \sigma_\varepsilon|\mathrm{VK}_0|\mathrm{VK}_1|\sigma_0$
8. Verify m_0 iff Verify-OT$(m = \mathrm{VK}_0|\mathrm{VK}_1, \sigma_\varepsilon, \mathrm{VK}_\varepsilon)$ and Verify-OT$(m_0, \sigma_0, \mathrm{VK}_0)$
9. Handle the second message m_1 analogously

We can continue generating keys and signatures in a similar fashion to get VK_{00}, SK_{00}, VK_{01}, SK_{01}, VK_{10}, $_{10}$, VK_{11}, and SK_{11} and be able to sign up to 4 messages. In fact, we could extend this for $\lambda = n$ levels to securely sign 2^n messages. However, if we did this all at once, the storage and setup costs would be completely inefficient. Instead, generate messages from the left to right of the leaves of the tree, generating only the path of SK, VKs and their siblings that is needed to sign some message m_q. The next section presents an algorithm for this process.

3 Efficient Depth-First Approach for Signing Many Messages

3.1 Algorithms

Algorithm 1 Setup(1^n)

```
Create $\mathrm{VK}_\varepsilon$, $\mathrm{SK}_\varepsilon$ with Setup-OT algorithm
Initialize counter $c := 0^n$
Initialize set of VKs produced $S := \{\varepsilon\}$
Set $\mathrm{VK} := \mathrm{VK}_\varepsilon$, $\mathrm{SK} := \mathrm{SK}_\varepsilon$
```

Algorithm 2 Sign(m, SK)

```
$\{c^{(i)}$ denotes the first $i$ bits of $c$; $c^{(0)} = \varepsilon, c^{(n)} = c\}$
for $i = 1 \ldots n$ do
  if $c^{(i)} \notin S$ then
    Run Setup-OT twice to get $\mathrm{VK}_{c^{(i-1)}|0}$, $\mathrm{SK}_{c^{(i-1)}|0}$; $\mathrm{VK}_{c^{(i-1)}|1}$, $\mathrm{SK}_{c^{(i-1)}|1}$
    $\sigma_{c^{(i-1)}} := \text{Sign-OT}(m = \mathrm{VK}_{c^{(i-1)}|0}|\mathrm{VK}_{c^{(i-1)}|1}, \mathrm{SK}_{c^{(i-1)}})$
    $S := S \cup \{c^{(i-1)}|0, c^{(i-1)}|1\}$
  end if
end for
$\sigma_c := \text{Sign-OT}(m, \mathrm{SK}_c)$
return $\sigma_m := \sigma_\varepsilon|\mathrm{VK}_0|\mathrm{VK}_1|\ldots|\sigma_{c^{(i)}}|\mathrm{VK}_{c^{(i)}|0}|\mathrm{VK}_{c^{(i)}|1}|\ldots|\sigma_{c^{(n-1)}}|\mathrm{VK}_{c^{(n-1)}|0}|\mathrm{VK}_{c^{(n-1)}|1}|\sigma_c|c$
{then increment $c$}
```

Fact 3.1 *Let the number of signatures on distinct messages that have been generated by the sign algorithm above be $Q = poly(n)$. Then $|S|$, the number of nodes for which a VK and SK have been generated, is bounded by $2Q + 2n$.*

Algorithm 3 Verify(m, σ_m, VK)

```
{check the path}
for $i = 1 \ldots n-1$ do
  if Verify-OT($\mathrm{VK}_{c^{(i-1)}|0}|\mathrm{VK}_{c^{(i-1)}|1}$, $\sigma_{c^{(i-1)}}$, $\mathrm{VK}_{c^{(i-1)}}$) = false then
    return false
  end if
end for
if Verify-OT($m$, $\sigma_c$, $\mathrm{VK}_c$) = false then
  return false
end if
return true
```

3.2 Security proof

Suppose that there exists an algorithm A_Q that requests $Q = poly(\lambda)$ signatures from our scheme (in order) and forges on some message (not necessarily in order) with non-negligible probability. We construct algorithm B from A_Q to break the one-time signature scheme by simulating for A_Q a modified version of the tree scheme's Sign algorithm and then using A_Q's forgery for a non-negligible advantage against the one-time challenger.

Algorithm 4 B(1^n)

```
Receive VK$^*$ from OTChallenger
Pick some $n^*$ from the set of nodes that will be in $S$ after $Q$ signatures are generated
Let $i^*$ := number of bits of $n^*$
if $n^* = \varepsilon$ then
    VK$_\varepsilon$ := VK$^*$, VK := VK$_\varepsilon$
    $c := 0^n$, $S := \emptyset$
else
    Run Setup($1^\lambda$) of tree signing schme
end if
Publish VK for $A_Q$
Sign up to $Q$ messages from $A_Q$ using modified SignB below
if $A_Q$ successfully forges $\sigma'$ on $m'$ and $\sigma'$ first deviates at $c'^{(i^*)} = n^*$ then
    Defeat one-time challenger with forged message $m = \mathrm{VK}'_{n^*|0}|\mathrm{VK}'_{n^*|1}$ and signature $\sigma'_{n^*}$
end if
```

The correctness of the construction can be established by verifying the following facts:

1. B satisfies all of A_Qs signature requests with signatures that would be verified by the tree signature scheme's verify algorithm. (The inputs for the black box A_Q are well-formed.)

2. B makes at most one signature request to the one-time challenger. (B works within the rules of the one-time signature challenge game.)

3. If A_Q's forgery first deviates at n^*, the specified components of that forgery constitute a message and signature that would be verified by the one-time challenger's signature scheme. (If B does not quit, B defeats the one-time challenger.)

B's efficiency and non-negligible advantage is established as follows: Let the (non-negligible) probability that A_Q successfully forges be ε. If A_Q successfully forges, then the node corresponding to the first deviation of the forged signature is n^* with probability $\geq 1/(2Q+2n)$. If this happens, B defeats the one-time signature challenger by the correctness of the construction. B's advantage is therefore at least $\varepsilon/(2Q + 2n)$, which is polynomial. The structure of the algorithms and B's dependence on efficient A_Q also guarantees that B runs in polynomial time. This completes the proof that our efficient many-message signature scheme is at least as secure as any one-time signature scheme.

Algorithm 5 SignB(m)

```
for $i = 1 \dots n$ do
  if $c^{(i)} \notin S$ then
    if $i = i^*$ and $c^{(i-1)} = n^{*(i-1)}$ then
      $\mathrm{VK}_{n^*} = \mathrm{VK}^*$
      if last bit of $n^*$ is 0 then
        Run Setup-OT to get $\mathrm{VK}_{c^{(i-1)}|1}$, $\mathrm{SK}_{c^{(i-1)}|1}$
      else
        Run Setup-OT to get $\mathrm{VK}_{c^{(i-1)}|0}$, $\mathrm{SK}_{c^{(i-1)}|0}$
      end if
    else
      Run Setup-OT twice to get $\mathrm{VK}_{c^{(i-1)}|0}$, $\mathrm{SK}_{c^{(i-1)}|0}$; $\mathrm{VK}_{c^{(i-1)}|1}$, $\mathrm{SK}_{c^{(i-1)}|1}$
    end if
    if $c^{(i-1)} = n^*$ then
      Request OTChallenger for signature $\sigma_{c^{(i-1)}}$ on $m = \mathrm{VK}_{c^{(i-1)}|0}|\mathrm{VK}_{c^{(i-1)}|1}$
    else
      $\sigma_{c^{(i-1)}} :=$ Sign-OT($m = \mathrm{VK}_{c^{(i-1)}|0}|\mathrm{VK}_{c^{(i-1)}|1}$, $\mathrm{SK}_{c^{(i-1)}}$)
    end if
    $S := S \cup \{c^{(i-1)}|0, c^{(i-1)}|1\}$
  end if
end for
if $n^* = c$ then
  Request OTChallenger for signature $\sigma_c$ on $m$
else
  $\sigma_c :=$ Sign-OT($m$, $\mathrm{SK}_c$)
end if
return $\sigma_m := \sigma_\varepsilon|\mathrm{VK}_0|\mathrm{VK}_1|\dots|\sigma_{c^{(i)}}|\mathrm{VK}_{c^{(i)}|0}|\mathrm{VK}_{c^{(i)}|1}|\dots|\sigma_{c^{(n-1)}}|\mathrm{VK}_{c^{(n-1)}|0}|\mathrm{VK}_{c^{(n-1)}|1}|\sigma_c|c$
{then increment $c$}
```

CS 388H Introduction to Cryptography 23-Sept-2009

Lecture 8: RSA, Full Domain Hash, and Random Oracles

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lecture 7 Recap

Last time we constructed a signature scheme that build trees of keys in a depth-first, on-demand manner. This used the one-time signature scheme (based on one-way function) to sign many messages. The downsides were that the signatures produced were linear in the size of the security parameter and we had to store the state, requiring considerable memory. More importantly, synchronizing the states of two machines that are supposed to be signing in parallel would introduce non-trivial implementation challenges.

Next, we will try to construct a signature scheme that is stateless, more efficient, and has shorter signatures. We will base these schemes on the RSA assumption.

2 RSA Assumption

Assumption 2.1 *Given $N = pq$, $h \in_u \mathbb{Z}_N^*$, and $e \in_u [1, \ldots, \varphi(N)]$ where $gcd(e, \varphi(N)) = 1$, computing y such that $y^e = h$ is hard without knowing $\varphi(N)$.*

3 Original ("Textbook") RSA Signature Scheme

3.1 Algorithms

1. Setup: Choose $N := pq$, publish VK $:= e$ as in RSA assumption, compute SK $:= d =_{\varphi(N)} e^{-1}$ (with Euclidean algorithm and $\varphi(N)$).
2. Sign($m \in \mathbb{Z}_N$): Return $\sigma := m^d$
3. Verify(σ, m, e): Check $\sigma^e = m$

3.2 Problems

- Some messages (like $m = 1$) are very easy to forge.
- Attackers can ask for two signatures and receive σ_0 for m_0 and σ_1 for m_1. Then it's easy to compute $\sigma = \sigma_0\sigma_1$ as the forgery for $m = m_0m_1$.
- Other basic algebraic tricks make forging certain messages easy.

4 Full Domain Hash RSA Signatures

4.1 Algorithms

1. Setup: Choose N, e, and d as before. Also publish a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$.
2. Sign($m \in \{0,1\}^*, d$): Return $H(m)^d$
3. Verify(σ, m, e): Check $\sigma^e = H(m)$

4.2 Approach to security analysis

For this to fix any of the security flaws of the original scheme, we need to prove some security property of H. For example, we may want something like SHAW1 that breaks the algebra that allowed us to make attacks like $(m_0 m_1, \sigma_0 \sigma_1)$ from (m_0, σ_0) and (m_1, σ_0).

5 Random Oracle Methodology

[Bellare and Rogaway, '93]

5.1 Intuition and definition

In the real world, an attacker has access to the actual code that implements a hash function H. In a hypothetical world, an attacker has only "oracle access" to H. An oracle generates random outputs for any input unless it has already been queried on that input, in which case returns the previous output.

5.2 Proof strategy

Proofs of security of a cryptosystem using the random oracle model will be proofs of statements of the form: "If there exists an attacker A that breaks a cryptosystem with random oracle access, then there exists an attacker B that contradicts the RSA assumption."

The underlying assumption that allows us to infer from a proof of such a statement that the cryptosystem is secure is that the hash function in question behaves like a random oracle, or alternatively that the attacker suffers no disadvantage from using the random oracle instead of the real H.

6 Proof of Security of the Full Domain Hash RSA Signature Scheme

We establish the security of the full domain hash (FDH) RSA under the random oracle model by giving a describing an algorithm B that would break the RSA assumption using a black box algorithm A that breaks the security of FDH with at most $Q = poly(N)$. As usual, the proof establishes B's efficiency and non-negligible advantage.

6.1 Description of B

1. B gets an RSA challenge N, e, and $h \in \mathbb{Z}_N^*$. B wins if it produces σ such that $\sigma^e = h$.
2. B gives N and e to A as the inputs for an FDH challenge.
3. B chooses $k \in_u \{1, \ldots, Q\}$ as guess for which random oracle query A will use to produce a forgery.
4. When A makes a random oracle query for the hash of m, B determines an output as follows: If A has previously queried on m, return the previous answer. If not, return $H(m) = h$ if it is the kth query and $H(m) = x_i^e$ for a random $x_i \in \mathbb{Z}_N^*$ otherwise.
5. When A makes a signature request for m, quit if m was the kth random oracle query and return $\sigma_m = x_j$ if m was the query $j \neq k$.
6. If A forges σ' on m' for m' not the kth oracle query, quit. Otherwise B outputs $\sigma^* = h^{1/e}$ to the RSA challenger.

6.2 Efficiency and advantage of B

Since there is constant time setup, a polynomial number of iterations of constant time steps 4 and 5, and step 6 is bounded by the (polynomial) runtime of A, B is efficient.

B breaks the RSA assumption with non-negligible advantage ϵ/Q because B has probability at least $1/Q$ of planting h in the correct random oracle query and A (independent of B's construction) has non-negligible advantage ϵ of being successful.

CS 388H Introduction to Cryptography 28-Sept-2009

Lecture 9: Public Key Encryption and IND-CPA Security

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Public Key Encryption

1.1 algorithms

- Gen(λ) $\rightarrow$ PK (public), SK (secret)
- Enc(PK, $m \in \mathcal{M}$, r = random bit string) $\rightarrow$ CT (ciphertext)
- Dec(SK, CT) $\rightarrow$ message m or failure $\perp$

1.2 Correctness property

Correctness requires that Dec(SK, Enc(PK, $m \in \mathcal{M}$, r)) $= m$ for all $m \in \mathcal{M}$ and random r with PK, SK produced by Gen.

1.3 Desired security properties

An eavesdropper (someone who observes CT and doesn't have access to SK) should not be able to read any part of m. We formalize this notion below.

2 IND-CPA Security

[Goldwasser and Micali]

A public key encryption scheme is said to be IND-CPA (indistinguishable under chosen plaintext attack) secure if any attacker (who implicitly knows the algorithms being used) has at most negligible advantage in the following challenge:

1. Challenger runs Gen(λ) $\rightarrow$ PK, SK; publishes PK.
2. Attacker chooses two messages m_0 and m_1 of equal length.
3. Challenger chooses $b \in \{0,1\}$ at random; outputs Enc(PK, m_b, r).
4. Attacker chooses $b' \in \{0,1\}$.

The attacker's advantage is $Adv_A = Pr[b' = b] - \frac{1}{2} = \dfrac{Pr[b' = 1|b = 1] - Pr[b' = 1|b = 0]}{2}$. The equivalence of these two definitions can be shown using conditional probability rules given that $Pr[b = 1] = Pr[b = 0] = 1/2$ when b is chosen randomly.

The IND-CPA security notion allows the attacker to choose between any two messages, so if there is any way to reveal any part or quality of a message without the secret key, some attacker could find it.

Why do we need randomness? Since the algorithms are known to the attacker, the attacker could compare his own encryption of m_0 and m_1 to the ciphertext output by the challenger and easily infer which message produced it.

What concepts do this definition not cover? The attackers model eavesdroppers in the game. We don't allow them choose ciphertexts and see how the Dec algorithm responds.

If we specify that even an attacker allowed to encrypt many messages must have a non-negligible advantage, does this result in a stronger concept of security?

3 Many-Message Security

Call a public key encryption scheme many-message secure if any attacker has at most negligible advantage in the following game:

1. Challenger runs Gen(λ) $\rightarrow$ PK, SK; publishes PK.
2. Attacker chooses two message vectors $\vec{M_0} = (m_{01}, \ldots, m_{0n})$ and $\vec{M_1} = (m_{11}, \ldots, m_{1n}$.
3. Challenger chooses $b \in \{0, 1\}$ at random; outputs $\vec{CT} = (Enc(PK, m_{b1}), \ldots Enc(PK, m_{bn}))$. (Random rs are also implicit parameters.)
4. Attacker chooses $b' \in \{0, 1\}$.

Define Adv_A as above. Are these notions equivalent?

4 Equivalence of IND-CPA and Many-Message Security

Note that if an encryption scheme is not IND-CPA secure, a trivial many-message attacker with $n = 1$ can copy the IND-CPA attacker to establish that it is not many-message secure. To prove that many-message security is not a stronger security definition but rather an equivalent one, we must prove the following theorem.

Theorem 4.1 *Any encryption system that satisfies IND-CPA also satisfies many-message security.*

Alternatively, we can prove the contrapositive: Suppose A breaks the many-message challenge for some encryption scheme E and show that from this we can break IND-CPA.

Proof. Consider the following fact:

Fact 4.2 *Consider probabilities $A_0, \ldots, A_n \in [0, 1]$. If we have $|A_{i+1} - A_i| < \varepsilon$ for all i, then we have $|A_n - A_0| < n\varepsilon$.*

Note: Our proof will associate cryptographic meaning to the A_i, but the fact holds for any arbitrary probabilities.

Now consider a series of $n+1$ experiments $Expt_0, \ldots, Expt_n$, where $Expt_i$ outputs the encryption of the first i messages from $\vec{M}_1$ and the last $n-i$ messages from $\vec{M}_0$. Define $A_i = Pr[\text{attacker } A \text{ outputs 1 on } Expt_i]$. It may be helpful to consider that A is not really constructed to handle these experiments, but it will always output 1 with some probability, so the notion is well-defined.

Lemma 4.3 *If an encryption scheme is IND-CPA secure, then $A_{i+1} - A_i$ is negligible $\forall i$.*

Proof. Assume there exists some $i = 0, \ldots n-1$ where $A_{i+1} - A_i$. Construct an attacker B for the IND-CPA challenge as follows:

1. B takes in PK from IND-CPA challenger
2. B forwards PK to A
3. A gives $\vec{M}_0$, $\vec{M}_1$ to B for encryption
4. B gives $m_{0,i+1}$, $m_{1,i+1}$ to IND-CPA challenger for encryption
5. B receives CT* as encryption of $m_{b,i+1}$ from IND-CPA challenger
6. B creates the ciphertext for A as follows:
   CT := Concatenate the encryption of the first i elements of $\vec{M}_1$
   CT := CT | CT*
   CT := CT | concatenate the last $n-(i+1)$ elements of $\vec{M}_0$
7. A outputs b'; B outputs b' to the IND-CPA challenger

We can see from the definition of $Expt_i$ and the way the CT is constructed that A sees $Expt_i$ if $b=0$ and $Expt_{i+1}$ if $b=1$. By assumption, the probability that A outputs 1 when $b=1$ is non-negligibly greater than the probability that A outputs 1 when $b=0$. Since B gives the IND-CPA challenger A's answer, this corresponds to a non-negligible advantage for B. This proves the contrapositive of the lemma and therefore the lemma itself. $\square$

The above lemma holds for $i = 0 \ldots n-1$. Apply the fact to conclude that $|A_n - A_0|$ is negligible if an encryption scheme is IND-CPA secure. This completes the proof of he theorem that many-message security is implied by IND-CPA security, thus establishing the equivalence of many-message and IND-CPA security. $\square$

Now that we have proved equivalence of the security notions, we can ignore the notationally-burdensome many-message security concept and focus on proving IND-CPA security.

CS 388H Introduction to Cryptography 30-Sept-2009

Lecture 10: Security of RSA Encryption in Random Oracle Model

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lecture 9 Recap: Hybrid Techniques

Last lecture we saw an example of using hybrid techniques in a security proof. In proving the equivalence of IND-CPA security and many-message security, we gave an attack algorithm hybrid experiments it wasn't trained to succeed on, but it's "success" probability was nonetheless well-defined, so we could use it to make statements about its success defeating the true challenger.

2 RSA Encryption: A First Attempt

- Gen(λ) $\to$ Choose $N = pq$ and $e \in \{1, \ldots, \varphi(N))\}$ coprime with N; publish PK $= (N, e)$, compute SK $= e^{-1}$
- Enc(PK, $m \in \mathcal{M}$, r = random bit string) $\to$ Choose random $x \in \mathbb{Z}_N^*$; output CT $= (c_1 = x^e, c_2 = m \oplus x)$
- Dec(SK, CT) $\to$ Output $m = c_1^d \oplus c_2$

Recall that you can compute the Jacobi symbol of a number in $\mathbb{Z}_N$ without knowing the factorization of N. Furthermore, $\left(\frac{x}{N}\right) = \left(\frac{x^e}{N}\right)$. Then the attacker knows not to guess b if $\left(\frac{c_1}{N}\right) \neq \left(\frac{c_2 \oplus m_b}{N}\right)$, giving him an advantage against the IND-CPA challenger.

Next, use hash functions to eliminate this security vulnerability.

3 RSA Encryption: Take Two

- Gen(λ) $\to$ Choose $N = pq$, $e \in \{1, \ldots, \varphi(N))\}$ coprime with N, and hash function $H : \mathbb{Z}_N^* \to \{0,1\}^k$; publish PK $= (N, e)$, compute SK $= e^{-1}$
- Enc(PK, $m \in \{0,1\}^k$, r = random bit string) $\to$ Choose random $x \in \mathbb{Z}_N^*$; output CT $= (x^e, m \oplus H(x))$
- Dec(SK, CT) $\to$ Output $m = H(c_1^d) \oplus c_2$

Reality check: We have preserved the correctness of the previous algorithm, but removed the potential for the attack we identified on the first scheme presented.

4 Security of RSA Encryption Scheme with Hashing

We establish the security of the previous scheme by proving the following theorem:

Theorem 4.1 *If the RSA assumption holds, then our encryption scheme is IND-CPA secure in the random oracle model.*

Proof. Assume there exists an attack algorithm $\mathcal{A}$ in the random oracle model that breaks the IND-CPA security of our scheme. Construct $\mathcal{B}$ as follows to defeat the RSA challenge using $\mathcal{A}$:

1. $\mathcal{B}$ accepts RSA challenge: N, e, h (challenge is to find x st $x^e = h$)
2. $\mathcal{B}$ gives PK $= N, e$ to $\mathcal{A}$
3. $\mathcal{B}$ gets request from $\mathcal{A}$ for an IND-CPA challenge with input messages m_0 and m_1
4. $\mathcal{B}$ gives $\mathcal{A}$ IND-CPA challenge CT$^* = (h, r)$, $r \in_u \{0,1\}^k$
5. Suppose $\mathcal{A}$ queries random oracle for $H(y)$. $\mathcal{B}$ checks if $y^e = h$ (ie whether $\mathcal{A}$ has "guessed" the underlying x st $c_1 = x^e$). If $y^e = h$, $\mathcal{B}$ outputs y as the answer to the RSA challenge and quits. Otherwise, $\mathcal{B}$ continues to accept random oracle queries.

For this construction to constitute a valid reduction from the IND-CPA security of RSA encryption to the RSA assumption, we have to justify that any non-negligibly successful RSA encryption attacker $\mathcal{A}$ will indeed query the random oracle on $y = h^{e^{-1}}$ with non-negligible probability. We do this with the following lemma:

Lemma 4.2 *Suppose $\mathcal{A}$ has $Adv_{\mathcal{A}} = \varepsilon \notin negl$ in breaking the IND-CPA security of the RSA encryption scheme with hashing. Then $\mathcal{A}$ must query $H(c_1^{e^{-1}})$ with non-negligible probability (when the encryption challenger outputs $CT^* = (c_1, c_2)$ after receiving m_0, m_1 from $\mathcal{A}$).*

Proof. Let W be the event that $\mathcal{A}$ outputs the correct b in the IND-CPA challenge for the RSA encryption scheme, and let Q be the event that $\mathcal{A}$ queries the random oracle for $H(c_1^{e^{-1}})$ and $\bar{Q}$ be the event's complement.

$$\begin{aligned}
Prob[W] &= Prob[W|\bar{Q}] \cdot Prob[\bar{Q}] + Prob[W|Q] \cdot Prob[Q] \\
&\leq 1/2 \cdot Prob[\bar{Q}] + Prob[Q] \\
&= 1/2 + 1/2 \cdot Prob[Q]) \\
1/2 + \varepsilon &\leq 1/2 + 1/2 \cdot Prob[Q] \\
2\varepsilon &\leq Prob[Q]
\end{aligned}$$

ε non-negligible implies 2ε is is non-negligible, and the lemma is proven. □

The lemma establishes that $\mathcal{B}$ successfully breaks the RSA assumption if $\mathcal{A}$ successfully attacks RSA encryption. This concludes the proof of the theorem about the security of the RSA encryption scheme under the random oracle model. □

At a high-level, this proof used $\mathcal{A}$'s reliance on the random oracle to extract information based on the queries rather than employing the "plug and pray" technique we've seen before. Next lecture, we'll see a security proof under the Decisional Diffie-Helman problem that doesn't rely on the random oracle model.

CS 388H Introduction to Cryptography 5-Oct-2009

Lecture 11: ElGamal Encryption

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lecture 10 Recap and Lecture 11 Agenda

Last time, we proved the security of an RSA encryption scheme by implicitly forcing the black-box attacker to use the random oracle. On the one hand, this was a reasonable stipulation because the random oracle was the only way for the attacker to see an encryption as anything other than random, but in some sense it violates the black-box nature of the attacker.

This time, we'll see our last encryption scheme before moving onto foundational cryptography concepts. We will show that the ElGamal encryption scheme is secure under the Decisional Diffie-Helman assumption without having to rely on assumptions about the black-box attacker's use of the random oracle.

2 Decisional Diffie-Helman Problem

Consider the following game:

1. Challenger chooses a group $\mathbb{G}$ of prime order p, two random elements $g, R \in \mathbb{G}$, and two integers $a, b \in \mathbb{Z}_p$.
2. Challenger gives attacker $\mathcal{A}$ a tuple $(\mathbb{G}, g, g^a, g^b, T)$, where T is either g^{ab} or R.
3. $\mathcal{A}$ guesses whether $T = g^{ab}$ or $T = R$.

Define $\mathcal{A}$'s advantage as $Adv_{\mathcal{A}} = Pr[\mathcal{A}$ guesses correctly$] - 1/2$.

$\mathbb{G}$ is said to be DDH-secure if $Adv_{\mathcal{A}}$ is negligible.

3 ElGamal Encryption Scheme

- Gen(λ) $\rightarrow$
  - Choose prime p (with $|p| \approx \lambda$) and create group $\mathbb{G}$ of prime order p
  - Choose $g \in \mathbb{G}$, $y \in \mathbb{Z}_p$
  - Set PK $= (g, h = g^y)$, SK $= y$
- Enc(PK, $m \in \mathbb{G}$, $r \in_u \mathbb{Z}_p$) $\rightarrow$ Output CT $= (c_1 = g^r, c_2 = mh^r)$
- Dec(SK, CT) $\rightarrow$ Decrypt $m = \dfrac{c_2}{c_1^y}$

Notes: So far, we've encrypted messages expressed as integers, but here we are encrypting group elements. It's fine for know to only understand abstractly that messages could be expressed as elements of an arbitrary group. As in previous messages, the randomness r is needed for security. The multiplier h^r in c_2 can be thought of as the blinding factor that obscures m.

4 Security of ElGamal Encryption Scheme

To show security of the ElGamal encryption scheme, we must establish that if the DDH assumption holds for some group $\mathbb{G}$ (ie $\mathbb{G}$ is DDH-secure), then ElGamal implemented with $\mathbb{G}$ is IND-CPA secure.

As usual, our overall approach for the proof of this theorem will be to assume there exists an attack algorithm $\mathcal{A}$ that defeats the ElGamal IND-CPA challenger and construct an algorithm $\mathcal{B}$ from $\mathcal{A}$ that break the DDH assumption.

However, we should first consider what might be challenging about this proof: there is a mismatch in the goals of the two attackers in that the IND-CPA attacker ($\mathcal{A}$) distinguishes between two chosen messages m_0 and m_1 whereas the DDH attacker ($\mathcal{B}$) distinguishes between one chosen message m' and a random message R.

To get around this, we will show that the advantage of any attacker in each of a sequence of games is approximately equal, and then show that there is negligible advantage in the second. Consider the following two games:

1. Define GameReal to be the IND-CPA challenge: attacker picks 2 messages and guesses which was encrypted as $\text{CT} = (g^r, m_b h^r)$.

2. Define GameRand to be the same as GameReal except that the attacker guesses which message was encrypted as $\text{CT} = (g^r, R')$, where R' is a random group element.

Claim 4.1 *Suppose there exists an attack algorithm $\mathcal{A}$ with $Adv_{\mathcal{A}-Real} - Adv_{\mathcal{A}-Rand} = \varepsilon$ (non-negligible). Then there exists and algorithm $\mathcal{B}$ that break the DDH assumption.*

Proof. We prove the claim by constructing $\mathcal{B}$ as follows:

1. $\mathcal{B}$ receives a DDH challenge $(\mathbb{G}, g, C = g^c, D = g^d, T = \{g^{cd}, R\})$

2. $\mathcal{B}$ gives $\mathcal{A}$ a public key (which implicitly includes the description of $\mathbb{G}$) for an ElGamal encryption scheme: $\text{PK} = (g, C)$

3. $\mathcal{B}$ receives two messages m_0 and m_1 chosen by $\mathcal{A}$ for the IND-CPA challenge, and $\mathcal{B}$ randomly chooses $b = \{0, 1\}$

4. $\mathcal{B}$ gives $\mathcal{A}$ $\text{CT}^* = (c_1^* = D, c_2^* = m_b T)$

5. $\mathcal{B}$ receives $\mathcal{A}$'s guess b' of which message was encrypted and guesses $T = g^{cd}$ if $\mathcal{A}$ was correct and $T = R$ otherwise

Step 4 determines whether $\mathcal{A}$ is playing GameReal or GameRand. $\mathcal{B}$ does not know whether $T = g^{cd}$ or $T = R$, but if the former is true, then $\mathcal{A}$ sees a valid ElGamal encryption of m_b (since d is hidden from $\mathcal{A}$ so it is no different from r in the encryption scheme), and if the latter is true then $\mathcal{A}$ sees a random power of g and a random element R' since m_b multiplied by a random element R just gives another random element.

In step 5, $\mathcal{B}$ finds out whether $\mathcal{A}$ succeeded or failed. $\mathcal{B}$ does not know which game $\mathcal{A}$ was playing, but by assumption, $\mathcal{A}$ is ε better at GameReal. Therefore, $\mathcal{B}$ takes $\mathcal{A}$'s success as an indication that $\mathcal{A}$ was playing GameReal and therefore that $\mathcal{B}$ was given $T = g^{cd}$ by the DDH challenger in step 1. This gives $\mathcal{B}$ a non-negligible advantage of $\varepsilon/2$ against the DDH assumption and the claim is proven. $\square$

It is not difficult to see that $Adv_{\mathcal{A}-Rand}$ is zero, since the game does not provide $\mathcal{A}$ with a ciphertext containing any information (obscured or otherwise) about m_0 or m_1. Because of this, the above claim immediately proves the following theorem:

Theorem 4.2 *Any algorithm that attacks the IND-CPA security of the ElGamal encryption scheme (ie some algorithm $\mathcal{A}$ with $Adv_{\mathcal{A}-Real} = \varepsilon \notin negl$) could be used to construct (as above) an algorithm $\mathcal{B}$ that breaks the DDH assumption.*

The theorem as stated above shows that ElGamal is IND-CPA secure as long as the underlying group $\mathbb{G}$ is DDH-secure.

CS 388H Introduction to Cryptography 7-Oct-2009

Lecture 12: One-Way Functions As Cryptographic Primitives

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Announcements and Course Roadmap

The in-class midterm is tentatively scheduled for November 4.

So far, the course has focused on cryptosystems and their security under number theory assumptions such as discrete log, RSA, DDH, etc. Now we will start to build cryptographic primitives with properties that are not derived from number theory assumptions. This will enable us to build cryptosystems from arbitrary instantiations of these general primitives that will remain secure in general even if a particular number theory assumption is disproven.

Today we will focus on the significance of one-way functions in this sense. We'll start to see how crypto assumptions (even very weak assumptions) can be used to build OWFs and how OWFs can be used to build cryptosystems. To justify the cryptographic importance of OWFs, we'll look at why the security of many crypto applications actually imply the existence of OWFs.

2 Definitions

Definition 2.1 *A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is a **one-way function** if and only if*

1. *evaluating f is easy (polynomial in the size of the input), and*
2. *inverting f is hard (for any efficient $\mathcal{A}$, $Pr[\mathcal{A}(y = f(x)) = x' \text{ st } f(x') = y] \in negl$, where the probability is taken over $\mathcal{A}$'s randomness and all inputs x).*

Definition 2.2 *A **length-preserving OWF** is a OWF f such that $|x| = |f(x)|$.*

Definition 2.3 *A **one-way permutation** is a OWF f such that the function f_n, denoting f with restricted domain $\{0,1\}^n$, is a bijection. (A bijection is a function whose domain and range have the same cardinality and every element in its range is mapped to by exactly one element in its domain. Because of this, one-way permutations have no collisions.)*

Definition 2.4 *A **family/collection of OWFs** is specified by 3 algorithms as follows:*

- *Gen(λ) $\rightarrow I$ indexing OWF $f_I : D_I \rightarrow R_I$*
- *Sample(I)$\rightarrow x \in D_I$*
- *Evaluate($I, x \in D_I$) $\rightarrow f_I(x) \in R_I$*

3 OWFs From Secure Signature Schemes

Claim 3.1 *If there exist signature schemes that are secure (existentially unforgeable under chosen message attack), then one-way functions exist.*

Proof. The approach for this proof will be to first define a function from the abstract descriptions of the setup, sign, and verify algorithms specifying all signature schemes. Then we will assume for contradiction that the function is not a OWF and use a black-box inversion algorithm to construct an algorithm that attacks the signature scheme.

We'll work through a first proposed function and show why it will not prove the above claim. A test question could be of the form "Illustrate why the following proposition fails":

Proposition 3.2 *For any secure signature scheme, Sign(SK, m) = σ is a OWF.*

Say we have some secure signature scheme. Modify the sign algorithm so there exists one degenerate SK* (eg 0^n) where Sign(SK$^*, m$) $= m$ for all $m \in \mathcal{M}$. This modification maintains the security of the signature scheme as long as Gen produces SK* with only negligible probability. Then an efficient inversion algorithm could produce SK$^*, m$ as a possible preimage for any OWF challenge m. This shows that Sign(SK, m) is not a OWF, so we have failed to prove the claim.

We will instead use the OWF construction suggested by the following lemma:

Lemma 3.3 *Define $f(x)$ to be the VK produced by a signature scheme's setup algorithm on random input x (a natural corresponding security parameter would be $\lambda = |x|$). If the signature scheme is secure, then f is a one-way function.*

Proof. Assume for contradiction that there exists some algorithm $\mathcal{A}$ that efficiently inverts f. Construct $\mathcal{B}$ to attack the signature scheme:

1. $\mathcal{B}$ receives VK from the signature scheme challenger and gives VK to $\mathcal{A}$ as the postimage of f for $\mathcal{A}$ to invert
2. $\mathcal{A}$ outputs a randomness parameter x satisfying $f(x) = \text{VK}$
3. $\mathcal{B}$ runs Setup($\lambda = |x|, r = x$) = VK$'$, SK$'$
4. $\mathcal{B}$ forges a signature for any message m^* with $\sigma^* = \text{Sign}(\text{SK}', m^*)$

It may be the case that step 3 produces VK$' \neq$ VK, but the VK$'$, SK$'$ pair produced is a valid VK, SK pair for the signature scheme simulated by $\mathcal{B}$'s challenger. If the signature scheme is secure, this construction shows that $\mathcal{A}$ cannot exist so f is indeed a OWF. □

The lemma shows how to construct a OWF from a secure signature scheme. This proves the claim that the existence of secure signature schemes implies the existence of OWFs. □

4 More Conjectured OWFs

4.1 OWF based on the hardness of factoring

It is conjectured that $f_{mult}(x, y) = x \cdot y, |x|, |y|$ is a OWF.

4.2 OWF based on NP-hard problem

$f_{subset-sum}(x_1, \ldots, x_n, J \in 2^{\{x_1,\ldots,x_n\}}) = \sum_{j \in J} x_j, x_1, \ldots, x_n$.

4.3 RSA collection of OWFs

- Gen(λ) $\rightarrow$ Choose p, q primes, $N = pq$, $e \in \mathbb{Z}_N^*$; set $I = N, e$
- Sample(I) $\rightarrow$ Output random element $x \in \mathbb{Z}_N$
- Evaluate($I, x \in D_I$) $\rightarrow x^e$

Notice that these f_I are one-way permutations.

5 Exercise

Suppose we f is a OWF. Let $g(x) = f(1, x_2, \ldots, x_n)$. Is g a OWF?

Assume there exists an attacker $\mathcal{A}$ that inverts g with probability ε. Consider an f-inverter $\mathcal{B}$ that passes an f inversion challenge y to $\mathcal{A}$ and returns $\mathcal{A}$'s answer x such that $g(x) = y$. If $x_1 = 1$, then g's definition implies $f(x) = y$ and $\mathcal{B}$ succeeds. $\mathcal{B}$ inverts f with probability $\varepsilon/2$. Since f is given to be a OWF, no such $\mathcal{A}$ exists, and we conclude that g is a OWF.

CS 388H Introduction to Cryptography 12-Oct-2009

Lecture 13: Hardcore Bits

Instructor: Brent Waters *TA: Sara Krehbiel*

The midterm is scheduled for November 4 (same time and place as lecture), and it will be closed-book and closed-notes.

1 Hardcore Bits

Last class we talked about how a OWF f is such that, given $y = f(x)$ it is hard to discover some x' such that $f(x') = y$. However, this makes no guarantees about how hard it is to find individual bits of x.

Definition 1.1 *A function $h : \{0,1\}^* \to \{0,1\}$ is **hardcore** for OWF f if and only if*

1. *evaluating $h(x)$ is easy (polynomial in the size of x), and*
2. *predicting $h(x)$ given $f(x)$ is hard.*

The second condition can be formalized in either of two ways:

- For any efficient $\mathcal{A}$ that predicts $h(x)$ given $f(x)$,

$$Pr[\mathcal{A}(f(x)) = h(x)] \leq 1/2 + negl(n).$$

- For any efficient $\mathcal{A}$ that distinguishes $h(x)$ from a random bit given $f(x)$,

$$Pr[\mathcal{A}(f(x), h(x)) = 1] - Pr[\mathcal{A}(f(x), b \in_u \{0,1\}) = 1] = negl(n).$$

In both cases, probability is taken over $\mathcal{A}$'s randomness and all possible inputs $x \in \{0,1\}^n$.

1.1 Sample hardcore bit exam question

Show a counterexample to the claim that the first bit of *any* OWF is hardcore.

Approach: Assume f is a OWF. Construct g from f such that g is a OWF and the first bit is not hardcore. (Ie $h(x) = x_1$ is not hardcore for g.) Example: Take $g(x) = x_1 | f(x_2 \ldots x_n)$. Exposing one bit of the pre-image of $g(x)$ does not break the one-wayness provided by f of of the remaining bits, but it allows the first bit to be read, so g constitutes a OWF in which the first bit is not hardcore.

2 Goldreich-Levin Theorem: Hardcore Predicates from OWFs

Theorem 2.1 *Suppose f is a OWF. For x, r st $|x| = |r|$, define $g(x, r) = f(x), r$. Then the following two conditions hold:*

1. *g is a OWF.*
2. *$h(x, r) =< x, r >= \oplus_{i=1}^{r} x_i \cdot r_i$ is hardcore for g.*

The first part of the proof is straightforward: construct a simple reduction from the problem of inverting g to the problem of inverting f to show that g must be a OWF.

The approach for the second condition is similar: assume that h is not hardcore (ie there exists an efficient attacker $\mathcal{A}$ with non-negligible advantage in predicting $h(x)$ given $f(x)$), and use this to invert f. (Note that after proving the first part of the theorem, inverting g using $\mathcal{A}$ would be an acceptable proof, but we will find it more direct to invert f.)

For the rest of lecture, we will prove that h is at least semi-hardcore. More precisely, we first show that there cannot exist a perfect attacker $\mathcal{A}$ that predict $h(x)$ with 100% success, and then we show a stronger result that there is no $\mathcal{A}$ that predicts $h(x)$ with at least $3/4 + 1/p(n)$ success (for some polynomial p).

2.1 Proof that h cannot be perfectly predicted

Lemma 2.2 *Assume there exists an efficient $\mathcal{A}$ such that $Pr[\mathcal{A}(g(x, r)) = h(x, r)] = 1$. Then there exists an efficient $\mathcal{B}$ that inverts f with non-negligible probability.*

Proof. First, some additional assumptions and notation:

- Assume for now that f is injective. Then we can assume $|f(x)| = |x|$ and $\mathcal{B}$ doesn't have to worry about guessing the length of x given $f(x)$.
- Let $e^{(i)} \in \mathbb{R}^n$ denote the vector with 1 in the ith position and 0 everywhere else.
- Note that $h(x, e^{(i)}) = x_i$.

Now consider the following construction of $\mathcal{B}$:

1. $\mathcal{B}$ is given $f(x)$ from the f inversion challenger.
2. For $i = 1 \dots n$, $\mathcal{B}$ challenges $\mathcal{A}$ to find the hardcore bit of $g(x, e^{(i)}) = f(x), e^{(i)}$ and sets x_i to be $\mathcal{A}$'s output.
3. $\mathcal{B}$ produces $x = x_1 \dots x_n$ as an inversion of $f(x)$.

If $\mathcal{A}$ perfectly predicts $h(x)$, then $\mathcal{B}$ inverts f by the observation that $h(x, e^{(i)}) = x_i$, contradicting that f is a OWF. Therefore, no perfect predictor of h exists. $\square$

2.2 Proof that h cannot be predicted with probability greater than 3/4

Lemma 2.3 *Assume $\exists$ efficient $\mathcal{A}$ with $Pr_{x,r\in\{0,1\}^n}[\mathcal{A}(g(x,r)) = h(x,r)] = 3/4 + 1/p(n)$. Then $\exists$ an efficient $\mathcal{B}$ that inverts f with non-negligible probability.*

How much of our previous construction of $\mathcal{B}$ do we have to change? All we know about $\mathcal{A}$ is that it succeeds with probability at least 3/4 over all the inputs and its randomness. We don't know whether there are certain inputs on which it always fails. Since the $e^{(i)}$ constitute only n of the 2^n possible inputs for r and $n/2^n << 1/4$, it may be the case that $\mathcal{A}$ fails on all of the inputs the previous construction gives it. We must consider alternate ways to determine the x_i from queries to $\mathcal{A}$ that guarantee some non-negligible probability of a successful prediction of the hardcore bit.

Fact 2.4 *$a(b \oplus c) = ab \oplus ac$.*

Corollary 2.5 *$h(x,r) \oplus h(x,e^{(i)} \oplus r) = h(x,r) \oplus h(x,r) \oplus h(x,e^{(i)}) = x_i$.*

Notice that while r and $e^{(i)} \oplus r$ are clearly related, they are independently random in $\{0,1\}^n$.

Next, we consider that $\mathcal{A}$'s probability of success is taken over all $x, r \in \{0,1\}^n$. This gives no guarantees about the probability of success taken over all r for a *given* x. If $\mathcal{A}$ succeeded sufficiently frequently on only a negligible amount of x, we would not be able to use it to invert $f(x)$ for random x with non-negligible probability. Instead, we want it to be the case that for some non-negligible fraction of x, $Pr_r[\mathcal{A}(f(x),r) = h(x,r)]$ is sufficiently high. For these x, we should be able to use $\mathcal{A}$ to invert f by feeding it inputs specified by the corollary.

Claim 2.6 *Let $S = \{x \text{ st } Pr_r[\mathcal{A}(f(x),r) = h(x,r)] \geq 3/4 + 1/2p(n)\}$. Then $|S| \geq \dfrac{2^n}{2p(n)}$.*

Proof. We prove the claim be contradiction: Suppose $|S| < 2^n/2p(n)$. Now we want to show that even if $\mathcal{A}$ always guesses the hardcore bit for $x \in S$, S is too small to satisfy that the total $Pr_{x,r}[\mathcal{A}$ guesses $h(x,r)] \geq 3/4 + 1/p(n)$ as the initial assumption about $\mathcal{A}$ requires. For notational simplicity, let G denote $\mathcal{A}(f(x),r) = h(x,r)$, the event that $\mathcal{A}$ succeeds.

$$\begin{aligned} Pr_{x,r}[G] &= Pr_r[G|x \in S]Pr[x \in S] + Pr_r[G|x \notin S]Pr[x \notin S] \\ &< 1 \cdot \frac{1}{2p(n)} + (\frac{3}{4} + \frac{1}{2p(n)}) \cdot 1 \\ &= \frac{3}{4} + \frac{1}{2p(n)} \end{aligned}$$

This completes the proof by contradiction of the lower bound on the size of S. □

To be able to do anything useful with the corollary that shows how to extract x_i from a specific pair of hardcore bits, we need a lower bound on the probability that $\mathcal{A}$ will produce the correct bits for both of those queries.

Claim 2.7 *For all $x \in S$ and $i = 1 \ldots n$,*

$$Pr_r[\mathcal{A}(f(x),r) = h(x,r) \cap \mathcal{A}(f(x),e^{(i)} \oplus r) = h(x,e^{(i)} \oplus r)] \geq 1/2 + 1/p(n).$$

Proof. We have no guarantee that the probability of these two events is independent, so we cannot simply multiply the separate probabilities. Instead, we use the **union bound**:

Fact 2.8 $Pr[X \cup Y] \leq Pr[X] + Pr[Y]$.

Let $G_{r'}$ denote $\mathcal{A}(f(x), r') = h(x, r')$. Then

$$\begin{aligned} 1 - Pr_r[G_r \cap G_{e^{(i)} \oplus r}] &= Pr_r[\neg G_r \cup \neg G_{e^{(i)} \oplus r}] \\ &\leq Pr_r[\neg G_r] + Pr_r[\neg G_{e^{(i)} \oplus r}] \\ &\leq 1 - (\frac{3}{4} + \frac{1}{2p(n)}) + 1 - (\frac{3}{4} + \frac{1}{2p(n)}) \\ &= \frac{1}{2} - \frac{1}{p(n)} \end{aligned}$$

so $Pr_r[G_r \cap G_{e^{(i)} \oplus r}] \geq 1 - (\frac{1}{2} - \frac{1}{p(n)}) = \frac{1}{2} + \frac{1}{p(n)}$ as desired. □

2.2.1 Sketch of the rest of the proof...

Construct $\mathcal{B}$ as follows:

1. $\mathcal{B}$ is given $f(x)$ from the f inversion challenger. We hope that $x \in S$ as defined above. The claim ensures this will happen a non-negligible fraction of the time.

2. For $i = 1 \ldots n$, $\mathcal{B}$ challenges $\mathcal{A}$ several times with random $r, r \oplus e^{(i)}$ pairs. Set x_i to be the bit corresponding to the majority result of this sample. (The main part of this proof we didn't covered is how to use Chernoff bounds to determine how many random samples - it will be poly(n) - we have to take to guarantee that the majority vote will correctly produce x_i with high probability.)

3. $\mathcal{B}$ produces $x = x_1 \ldots x_n$ as an inversion of $f(x)$.

This construction shows that there cannot be any efficient $\mathcal{A}$ that predicts $h(x)$ with probability $> 3/4$. The proof that there is no efficient $\mathcal{A}$ that predicts $h(x)$ with any non-negligible advantage is known but harder.

CS 388H Introduction to Cryptography 14-Oct-2009

Lecture 14: Pseudorandom Generators

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Recap: Constructing a Hardcore Predicate from a OWF (Goldreich-Levin Theorem)

Our goal last class was to show that no attacker can guess the hardcore predicate $h(x,r)$ for modified OWF g with probability at least $3/4+1/p(n)$. If we had shown this for probability $1/2+1/p(n)$ (which can be done), that would prove that $h(x,r)$ is hardcore for g. Here is a summary of the lemmas and proof strategies we used:

1. Corollary 2.5 in the previous lecture notes shows how $\mathcal{B}$ can determine x_i given $\mathcal{A}$'s (correct) predictions of $h(x,r)$ and $h(x,e^{(i)}\oplus r)$.
2. To ensure that $\mathcal{B}$ was non-negligibly successful, in Claim 2.6, we defined a set S of inputs x on which $\mathcal{A}$ was successful $>3/4$ of the time and showed that it contained a non-negligible fraction of all possible inputs x.
3. Claim 2.7 used the union bound to show that $\mathcal{A}$ correctly predicts both $h(x,r)$ and $h(x,e^{(i)}\oplus r)$ non-negligibly greater than half the time for $x \in S$.
4. Because $\mathcal{B}$ hopes $x \in S$ and $\mathcal{A}$ succeeds on non-negligibly greater than 1/2 of the random inputs $h(x,r)$ and $h(x,e^{(i)}\oplus r)$, $\mathcal{B}$ has non-negligible advantage.

We skipped the explanation of how to use Chernoff bounds to bound the probability that $>1/2$ of $p(n)$ trials of $h(x,r)$ and $h(x,e^{(i)}\oplus r)$ will be wrong. This may come up on a subsequent problem set.

2 Pseudorandom Generators: Preliminaries

Having shown roughly how to construct hardcore predicates, we next discuss pseudorandom generators.

Intuition. We want to be able to take a random seed and amplify it in a way that attackers that don't see the seed can't tell whether the expanded randomness is amplified from some seed or true randomness.

Usefulness. Computers build up entropy from events like keystrokes, network interrupts, etc. The ability to amplify this randomness is important for situations in which a computer has a small amount of truly random bits and a running program needs many more random bits.

Definition 2.1 *$G : \{0,1\}^n \to \{0,1\}^l$ (where l is a function of n with $l > n$) is a pseudorandom generator (PRG) iff any efficient $\mathcal{A}$ (which may know the code for and be able to evaluate G) is such that $Pr_{x \in \{0,1\}^n}[\mathcal{A}(G(x)) = 1] - Pr[\mathcal{A}(U_l) = 1] \in negl(n)$, where U_l denotes an l-bit string of bits chosen uniformly at random.*

3 Proof of Existence of PRGs from OWPs

We take a 2-step approach to proving the existence of PRGs with output size $l = p(n)$ for an arbitrary polynomial p from OWPs:

1. Show that the existence of an OWP with domain and range $\{0,1\}^n$ implies the existence of a PRG that amplifies n bits of randomness by one bit.
2. Show that the existence of a PRG that amplifies a single bit of randomness implies the existence of a PRG that amplifies n bits to l bits.

Question: We've shown how to construct OWFs from secure signature schemes and seen examples of functions believed to be one-way based on hardness assumptions, but how many OWPs do we actually know? Ie is proving PRGs from OWPs even a useful exercise? Hill '89 shows the stronger result that OWFs imply PRGs. The proof approach is more subtle so we won't cover it, but a lot of the OWP $\implies$ PRG techniques we are about to see help.

3.1 $\exists$ OWP $f : \{0,1\}^n \to \{0,1\}^n \implies \exists$ PRG $g : \{0,1\}^n \to \{0,1\}^{n+1}$

Let $f : \{0,1\}^n \to \{0,1\}^n$ by a OWP with hardcore predicate h (h needn't be the GL HP we saw last time, but that construction establishes that some HP h exists), and define $g(x) = f(x), h(x)$.

Lemma 3.1 *If there exists an efficient algorithm $\mathcal{A}$ that defeats the PRG game with g, then there exists an efficient $\mathcal{B}$ that guesses $h(x)$ given $f(x)$.*

Proof. Assume there exists some $\mathcal{A}$ that successfully distinguishes $g(x)$ from random bits. Formally,

$$Pr_x[\mathcal{A}(f(x), h(x)) = 1)] - Pr[\mathcal{A}(U_{n+1}) = 1)] \geq \epsilon \notin negl(n)$$

Note that $x \sim U_n$ implies $f(x) \sim U_n$ since f is a permutation, so $U_{n+1} \sim f(x), U_1$, where U_1 is $h(x)$ half the time and $\bar{h(x)}$ the rest of the time. The second term of the above equation is therefore equivalent to $\frac{1}{2}(Pr_x[\mathcal{A}(f(x), h(x)) = 1] + Pr_x[\mathcal{A}(f(x), \bar{h(x)}) = 1])$ and we can regroup the terms in the equation to get:

$$\frac{1}{2}(Pr_x[\mathcal{A}(f(x), h(x)) = 1] - Pr_x[\mathcal{A}(f(x), \bar{h(x)}) = 1]) \notin negl(n)$$

This shows that a PRG attack algorithm $\mathcal{A}$ can also serve as an algorithm that guesses a hardcore bit of a OWF. By definition of OWFs and hardcore bits, this cannot happen, so g must be a PRG. □

3.2 $\exists$ PRG $g : \{0,1\}^n \to \{0,1\}^{n+1} \implies \exists$ PRG $g' : \{0,1\}^n \to \{0,1\}^l$

We present an algorithm for constructing $g' : \{0,1\}^n \to \{0,1\}^l$ using PRG $g : \{0,1\}^n \to \{0,1\}^{n+1}$. Note that we require only that g is a PRG; it needn't be $f(x), h(x)$ as in the proof of the previous lemma.

Algorithm 1 $g'(x \in \{0,1\}^n)$

```
  $s_0 := x$
  for $i = 1, \ldots, l$ do
    $s_i, \sigma_i := g(s_{i-1})$
  end for
  return $\sigma_1, \sigma_2, \ldots, \sigma_l$
```

Lemma 3.2 *g' defined by Algorithm 1 is a PRG. Formally, no efficient algorithm can distinguish $g'(x) = \sigma_1, \ldots, \sigma_l$ from U_l with non-negligible advantage for random, secret x.*

Proof. The proof uses a hybrid argument based on a set of experiments $\text{Expt}_1, \ldots, \text{Expt}_l$, where Expt_i generates l bits as follows:

- Choose $\sigma_1, \ldots, \sigma_{i-1}$ at random.
- Choose s_{i-1} uniformly at random from $\{0,1\}^n$.
- Run the g' algorithm starting at the ith iteration of the for loop to determine $\sigma_i, \ldots, \sigma_l$.

Then Expt_1 gives $g'(x)$ for random x and Expt_{l+1} is U_l.

We show that $Pr[\mathcal{A}(\text{Expt}_1) = 1] - Pr[\mathcal{A}(\text{Expt}_{l+1}) = 1] \in negl(n)$ (ie g' is a PRG) by showing the following claim.

Claim 3.3 *$Pr[\mathcal{A}(Expt_i) = 1] - Pr[\mathcal{A}(Expt_{i+1}) = 1] \in negl(n)$ for $i = 1, \ldots, l$.*

Assume for contradiction that the previous expression is not negligible for some i. Then we can construct $\mathcal{B}$ as follows to break PRG g:

- $\mathcal{B}$ gets $y = \{g(x), U_{n+1}\}$ from the PRG challenge for g.
- $\mathcal{B}$ chooses $\sigma_1, \ldots, \sigma_{i-1}$ at random.
- $\mathcal{B}$ sets $\sigma_i = y_{n+1}$.
- $\mathcal{B}$ runs the g' algorithm starting at the $(i+1)$th iteration of the for loop using $s_i = y_1, \ldots, y_n$ to produce $\sigma_{i+1}, \ldots, \sigma_l$.
- $\mathcal{B}$ gives $\sigma_1, \ldots, \sigma_l$ to $\mathcal{A}$ and uses $\mathcal{A}$'s output as its output to the PRG challenger for g.

Note that if $y = g(x)$, then $\mathcal{A}$ receives Expt_i with $s_{i-1} = x \sim U_l$. If $y \sim U_{l+1}$, then $\mathcal{A}$ receives Expt_{i+1} with $s_i = y_1, \ldots, y_n \sim U_l$. $\mathcal{B}$'s advantage therefore is exactly $\mathcal{A}$'s advantage. Since g is a PRG by assumption, this proves the claim. The lemma follows from the fact that there are a polynomial number of experiments, all of which are individually close. $\square$

CS 388H Introduction to Cryptography 19-Oct-2009

Lecture 15: Pseudorandom Functions

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Lecture 14 Recap: PRGs from OWPs

Last lecture we wanted a PRG $g : \{0,1\}^n \rightarrow \{0,1\}^l$ from a OWP $f : \{0,1\}^n \rightarrow \{0,1\}^n$. We did this in two steps:

1. We showed that for OWF f, $g(x) = f(x), h(x)$ was a PRG by reducing a PRG attack algorithm to an algorithm that guessed the $h(x)$ given $f(x)$, which is not possible for OWP f and HP h.

2. We showed how to construct $g' : \{0,1\}^n \rightarrow \{0,1\}^l$ using PRG $g : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ and established that g' was a PRG by defining a series of hybrid experiments and showing that each pair of consecutive experiments was indistinguishable based on the assumption that g is a PRG.

2 Pseudorandom Functions

Definition 2.1 *A* ***random function*** $F : \{0,1\}^k \rightarrow \{0,1\}^l$ *is specified by a table of* $2^k \times l$ *random bits.*

Consider a family of functions $F : \{0,1\}^k \rightarrow \{0,1\}^l$ where each is specified uniquely by a random key K. Efficiency essentially requires $k = p(\lambda)$. Usually, $|K| \in O(\lambda)$. In any case, $|K| << 2^k \cdot l$, so the amount of randomness required in such a set of functions is polynomial in λ compared to the exponential amount of randomness needed to define a truly random function. We want a good notion of pseudorandom functions for a couple reasons:

1. They can be used to create symmetric key cryptography. We will show today that PRFs can be built from PRGs, which we know exist if OWFs exist. This gives symmetric key cryptography an advantage over public key cryptography in that it is based on the very weak and general assumption of the existence of OWFs, so its security is potentially stronger.

2. Next lecture, we'll see how to use PRFs to build many-message signature schemes that avoid the state explosion of the tree-based scheme from lecture 7.

Definition 2.2 *F as above is a* ***secure pseudorandom function family*** *iff all efficient $\mathcal{A}$ are such that $Pr_f[\mathcal{A}^{f()} = 1] - Pr_K[\mathcal{A}^{F_K()} = 1]$ is negligible.*

The notation above essentially means $\mathcal{A}$ is not significantly more likely to detect true randomness when given oracle access to a truly random function f than when it is given oracle access to F_K from family F (where K is hidden from $\mathcal{A}$).

Why is this setup different from (and maybe more legitimate than) the random oracle model? In the random oracle model, we assume that H behaves as a random function despite the fact that an attacker in reality could have access to all of the code specifying H. Here, we give the attacker the description of function *family* F but *explicitly* withhold K.

3 Constructing PRFs from PRGs

We now present the Goldreich, Goldwasser, and Micali construction of a secure PRF family $F : \{0,1\}^k \to \{0,1\}^n$ with security parameter λ (ie $|K| = O(\lambda)$) from a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$.

Given $s \in \{0,1\}^n$, define $G_0, G_1 : \{0,1\}^n \to \{0,1\}^n$ so that $G(x) = G_0(s)|G_1(s)$. Define F_K ($|K| = n$) with the following algorithm:

Algorithm 1 $F_K(x \in \{0,1\}^k)$

```
{Here, we label the s with 0, ..., k. In the proof, we will label them with prefixes of x.}
$s_0 := K$
for $i = 1 \ldots k$ do
    $s_i := G_{x_i}(s_{i-1})$
end for
return $s_k$
```

We want to prove that F is a secure PRF family if G is a PRG. Our approach will be a series of double-hybrid experiments.

We are no longer concerned with a single s_i specified at each level $i = 0 \ldots k$, but with each possible $\tilde{s}_{x^{(i)}}$, where $x^{(i)} = x_1, \ldots, x_i$ is the ith prefix of x.

Define hybrid H_i to be the experiment where we choose $\tilde{s}_y$ uniformly at random for all values of y st $|y| \leq i$. Then compute $\tilde{s}_{x^{(j)}} := G_{x_j}(\tilde{s}_{x^{(j-1)}})$ for $j = i+1 \ldots k$ and return $\tilde{s}_x$.

H_0 is F_K and H_k is a random function, so if we show that $Pr[\mathcal{A}(H_i) = 1] - Pr[\mathcal{A}(H_{i+1}) = 1]$ is negligible, we can conclude that F is a PRF family.

In H_i, the s in the first i levels of the tree are generated randomly and subsequent nodes are generated with G_0 or G_1 (half of the PRG) seeded with the value of the parent node, specified by $x^{(i)}$. In H_{i+1}, the s in the first $i+1$ levels are all random. To prove indistinguishability, we define another set of hybrid experiments where $H_{i,0} = H_i$ and $H_{i,last} = H_{i+1}$ and $H_{i,j}$ is indistinguishable from $H_{i,j+1}$.

One idea: Successive sub-experiments could make two more of the level-$(i+1)$ nodes random from left to right. What is the problem with this? Even if it is the case that successive sub-experiments are only negligibly distinguishable, there are 2^i experiments, so this will not show that H_i an H_{i+1} are only negligibly distinguishable.

Better idea: Assume wlog that $\mathcal{A}$ makes at most $t = poly(n)$ oracle queries. Use these queries to define intra-level hybrid experiments. (We'll pursue this next lecture.)

CS 388H Introduction to Cryptography 21-Oct-2009

Lecture 16: PRFs Continued and Symmetric Key Cryptography

Instructor: Brent Waters *TA: Sara Krehbiel*

1 GGM PRFs: Lecture 15 Recap and Security Reduction

Algorithm 1 $F_K(x \in \{0,1\}^k)$

```
{Here, we label the $s$ with $0, \dots, k$. In the proof, we will label them with prefixes of $x$.}
$s_0 := K$
for $i = 1 \dots k$ do
    $s_{x^{(i)}} := G_{x_i}(s_{x^{(i-1)}})$
end for
return $s_x$
```

Last lecture we presented the GGM construction of PRF family $F : \{0,1\}^k \to \{0,1\}^n$ from a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$ and began to outline the proof that F_K is indeed a PRF. The algorithm above is the same as from lecture 15 with the seeds/nodes specified with the successive prefixes of input x rather than just the level. Here we complete the proof.

Theorem 1.1 *If G is a PRG, then F_K as defined by Algorithm 1 is a PRF.*

Proof. Our approach will be to illustrate if any efficient PRF attacker $\mathcal{A}$ can distinguish between any two of a polynomial number of 2-level hybrid experiments, then $\mathcal{A}$ can be used to defeat the PRG challenger on G. We suppose $\mathcal{A}$ makes $t \in poly(n)$ queries to the oracle it is provided (truly random f or pseudorandom F_K as above with K hidden).

Define H_i to be the experiment that on query x generates $s_{x^{(i)}}$ randomly and uses the F_K algorithm starting at the $(i+1)$th iteration to produce s_x as the oracle's output.

We can see that H_0 mirrors F_K and H_k mirrors a truly random function. As with other hybrid proofs, because there are a polynomial number (k) of experiments, if we can establish that $\mathcal{A}$ cannot distinguish any H_i from H_{i+1}, then we have shown that F is a secure pseudorandom function family.

To show $Pr[\mathcal{A}^{H_{i+1}} = 1] - Pr[\mathcal{A}^{H_i} = 1] \in negl(n)$, we define a series of hybrid experiments $H_{i,j}$ where $H_{i,0} = H_i$ and $H_{i,t} = H_{i+1}$ and show that $\mathcal{A}$ cannot distinguish between any of these on the assumption that G is a PRG.

Define $H_{i,j}$ to be the experiment that generates outputs for the first j queries requested by $\mathcal{A}$ by choosing random $s_{x^{(i)}|0}, s_{x^{(i)}|0}$ and then picking up at the $(i+2)$th iteration of F_K (seeded with $s_{x^{(i+1)}}$). The remaining queries are generated by choosing random $s_{x^{(i)}}$ and picking up at the $(i+1)$th iteration of F_K.

Now we are ready to show if $\mathcal{A}$ distinguishes between $H_{i,j}$ and $H_{i,j+1}$, we can construct a $\mathcal{B}$ that breaks the PRG challenge (r_0, r_1) for G by keeping track of $\mathcal{A}$'s t queries and producing outputs for them using the following algorithm:

Algorithm 2 $\mathcal{B}$ for $\mathcal{A}$ that distinguishes $H_{i,j}$ from $H_{i,j+1}$ with queries $x_1, \ldots, x_t$

```
for $\ell = 1 \ldots t$ do
  if $x^{(i)}$ has been seen before then
    use old values of $s_{x^{(i)}|0}$ and $s_{x^{(i)}|1}$
  else if $\ell \le j$ then
    choose $s_{x^{(i)}|0}$ and $s_{x^{(i)}|1}$ at random
  else if $\ell \ge j + 2$ then
    choose $s_{x^{(i)}|0} = G_0(S_{x^{(i)}})$ and $s_{x^{(i)}|1} = G_1(S_{x^{(i)}})$
  else
    $\{\ell = j + 1\}$ choose $s_{x^{(i)}|0} = r_0$ and $s_{x^{(i)}|1} = r_1$
  end if
  start running $F_K$ from the $i + 2$th iteration and give $s_k$ to $\mathcal{A}$
end for
return $\mathcal{A}$'s answer to PRF challenge as answer to PRG challenge $(r_0, r_1)$
```

If (r_0, r_1) are truly random, then $\mathcal{B}$ simulates $H_{i,j+1}$ for $\mathcal{A}$. If $(r_0, r_1) = G(s)$ for some s, then $\mathcal{B}$ simulated $H_{i,j}$. $\mathcal{B}$'s advantage in the PRG game is therefore exactly $\mathcal{A}$'s advantage in distinguishing $H_{i,j+1}$ from $H_{i,j}$. Because G is given to be a PRG, $\mathcal{A}$'s advantage must be negligible, and this completes the proof that F_K is a PRF. $\square$

2 More Space-Efficient Signature Schemes Using PRFs

Recall the tree-based many message signature scheme from lecture 7. Every time we reached a new node we had to not only generate but store two new public and private key pairs for its children in order to be able to sign messages along that path in the future. However, we could have instead stored the randomness that the key generation algorithm used and just regenerated the same pairs as needed.

So far, this still requires O(# messages signed) storage. However, now that we have established the existence of PRFs that deterministically produce random-looking outputs for a given input, we can use pseudorandom bits for the key generation at each node. Then we can store K and any time a key pair for the children of some node is required, we use F_K(node ID #) as the pseudorandom (but deterministically computed) input for the key gen algorithm. This reduces the space requirement to a $|K|$.

3 Symmetric Key Cryptography

Symmetric key cryptography assumes a single shared key SK required by the both the encrypting party and the decrypting party. This is less convenient than public key cryptography in that it requires transfer of a secret, but now that we know how to construct PRFs from OWFs, we will be able to argue the security of symmetric key cryptosystems on extremely weak assumptions.

3.1 Algorithms for symmetric key encryption

- $Gen(\lambda) \to SK$
- $Enc(SK, m \in \mathcal{M}) \to CT$
- $Dec(SK, CT) \to m \cup \perp$

3.2 Security definitions

We'll first give the definition for security under chosen plaintext attack, which is the symmetric key analogue to IND-CPA security of public key encryption. Then we will introduce a new, stronger form of security under chosen ciphertext attack.

3.2.1 CPA security for symmetric key encryption schemes

A symmetric key encryption scheme E is CPA secure if for any efficient attack algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible ($Adv_{\mathcal{A}} = Pr[b' = b] - 1/2$):

- Challenger computes $SK \leftarrow Gen_E(\lambda)$.
- Challenger gives $\mathcal{A}$ oracle access to $Enc_E(SK, \cdot)$. ($\mathcal{A}$ can request $CT = Enc_E(SK, m)$ for $poly(\lambda)$ messages $m \in \mathcal{M}$ throughout the course of the game. You can think of this as the symmetric key analogue to giving VK to $\mathcal{A}$ in the public key IND-CPA challenge, because it similarly gives $\mathcal{A}$ the ability to encrypt messages.)
- $\mathcal{A}$ is allowed to choose m_0, m_1 for the challenger to encrypt. (Assume $|m_i| = |m_j|$ $\forall m_i, m_j \in \mathcal{M}$.)
- Challenger picks $b \in \{0, 1\}$ and gives $CT^* = Enc_E(SK, m_b)$ to $\mathcal{A}$.
- $\mathcal{A}$ gives guess b'.

Note that $\mathcal{A}$ may query the oracle for one of the m_b it provides to the challenger for encryption, but $\mathcal{A}$ has no control over the randomness used in any of the algorithms.

3.2.2 CCA security for symmetric key encryption schemes

A symmetric key encryption scheme E is CCA secure if for any efficient $\mathcal{A}$, $\mathcal{A}$'s advantage is negligible in the previous game with the added stipulation that $\mathcal{A}$ also receives oracle access to $Dec_E(SK, \cdot)$ and can ask for decryptions of any $poly(\lambda)$ number of CT for $CT \neq CT^* = Enc_E(SK, m_b)$.

3.3 Construction of a CPA-secure symmetric key encryption scheme

Assume we have a PRF $F : \{0,1\}^\ell \to \{0,1\}^n$ with functions indexed by keys K. Define the algorithms of an encryption scheme as follows:

- $Gen(\lambda) \to$ Pick PRF key K at random; set $SK := K$.
- $Enc(SK, m \in \mathcal{M} = \{0,1\}^n) \to$ Choose random $r \in \{0,1\}^\ell$; set $CT := (r, F_{SK}(r) \oplus m)$.
- $Dec(SK, CT = (c_1, c_2)) \to$ Decrypt $m = c_2 \oplus F_{SK}(c_1)$.

It is easy to check that this scheme is correct. Its CPA security will be shown next lecture using a reduction showing that $\mathcal{A}$'s advantage in this scheme is close to as in a scheme using truly random f instead of PRF F_{SK}. The security of the truly random scheme is straightforward to establish.

CS 388H Introduction to Cryptography 26-Oct-2009

Lecture 17: Message Authentication Codes

Instructor: Brent Waters *TA: Sara Krehbiel*

1 From Last Lecture: Finishing Proof of Security of PRF-Based Symmetric Key Encryption Scheme

Consider the algorithms for the symmetric key encryption scheme that was proposed last lecture as CPA-secure. It assumes the existence of a PRF family $F : \{0,1\}^\ell \to \{0,1\}^n$.

- $Gen(\lambda) \to$ Pick PRF key K at random; set $SK := K$.
- $Enc(SK, m \in \mathcal{M} = \{0,1\}^n) \to$ Choose random $r \in \{0,1\}^\ell$; set $CT := (r, F_{SK}(r) \oplus m)$.
- $Dec(SK, CT = (c_1, c_2)) \to$ Decrypt $m = c_2 \oplus F_{SK}(c_1)$.

1.1 Size of ℓ

Last lecture, the question of how large ℓ needed to be for security was posed. Because attacker $\mathcal{A}$ is allowed oracle access to the encrypt algorithm, a polynomial number of queries could potentially give $\mathcal{A}$ non-negligible information about F_{SK} if the size of F's domain (2^ℓ) is not exponential in λ. To avoid this, choose $\ell \approx \lambda$.

1.2 Security proof approach

Let GameReal be the IND-CPA challenge for the above encryption scheme. Let GameRandom be the IND-CPA challenge for the encryption scheme with F replaced by a truly random function f.

If we can show that any algorithm's advantage in GameReal is at most negligibly different than its advantage in GameRand and that its advantage in GameRand is at most negligible, then we will have established the CPA-security of the encryption scheme.

Lemma 1.1 *For all efficient $\mathcal{A}$, $|Adv_{\mathcal{A},Real} - Adv_{\mathcal{A},Rand}| \in negl(\lambda)$.*

Proof. Assume for contradiction that there exists some $\mathcal{A}$ for which the lemma does not hold. We construct $\mathcal{B}$ that breaks the PRF property of F.

- PRF challenger gives $\mathcal{B}$ oracle access to either f or F_K (K hidden).
- $\mathcal{B}$ simulates an IND-CPA challenger for the encryption scheme for attacker $\mathcal{A}$, replacing computation of $F_K(r)$ with oracle queries through the PRF challenger.
- $\mathcal{A}$ gives $\mathcal{B}$ two messages, m_0, m_1.
- $\mathcal{B}$ chooses random bit b and passes encryption of m_b to $\mathcal{A}$.

- $\mathcal{A}$ makes guess b'.
- $\mathcal{B}$ outputs 1 to the PRF challenger if $b' = b$ and 0 otherwise.

$\mathcal{B}$ simulates GameReal for $\mathcal{A}$ if the PRF challenger gives him oracle access to F_K and GameRand if his oracle access is to f. By forwarding the correctness of $\mathcal{A}$'s guess, he effectively harnesses $\mathcal{A}$'s de facto ability to distinguish between the games to achieve a non-negligible advantage against the PRF challenger. F is given to be a OWF, so the lemma is proven. □

Lemma 1.2 *For all efficient $\mathcal{A}$, $Adv_{\mathcal{A},Rand} \in negl(\lambda)$.*

Proof. We may note that $f(r) \oplus m_b \sim U_\ell$ and infer that $Adv_{\mathcal{A},Rand} = 0$. However, we have to consider that the attacker is allowed $Q \in poly(\lambda)$ encryption queries and therefore may know $f(r)$ for up to Q values of r. If the encryption algorithm randomly chooses one of these values for r when it encrypts m_b, the attacker can figure this out from c_1 and decrypt the message perfectly before guessing. However, there are 2^ℓ possible values for r, so we can use the union bound to establish that the attacker only can guess perfectly with at most negligible probability $Q/2^\ell$ and in all other cases $f(r) \oplus m_b \sim U_\ell$ implies zero advantage for $\mathcal{A}$. □

We have shown that no efficient polynomial $\mathcal{A}$ breaks IND-CPA and the proposed OWF-based symmetric key encryption scheme is therefore IND-CPA secure.

2 Exercises

Here we present two exercises using the IND-CPA' game, which we define to be the same as the IND-CPA game except that in addition to encryption oracle access, the attacker can specify what randomness is used in oracle queries.

2.1 Security of modification to previous encryption scheme

Argue whether, given the security of the previous encryption scheme and OWF $g : \{0,1\}^\ell \to \{0,1\}^\ell$, the following encryption scheme is IND-CPA' secure:

- $Gen(\lambda) \to$ Pick PRF key K at random; set $SK := K$.
- $Enc(SK, m \in \mathcal{M} = \{0,1\}^n) \to$ Choose random $r \in \{0,1\}^\ell$ or specific r specified by attacker; $CT := (g(r), F_{SK}(g(r)) \oplus m)$.
- $Dec(SK, CT = (c_1, c_2)) \to$ Decrypt $m = c_2 \oplus F_{SK}(c_1)$.

2.2 Security of modification to arbitrary encryption scheme

Argue whether IND-CPA' security is implied by IND-CPA security for general encryption schemes.

3 Message Authentication Codes (MACs)

MACs are symmetric key analogues of signature schemes. The motivation for symmetric vs public key signatures is that they may be more efficient.

3.1 Algorithms

A MAC scheme is characterized by the following algorithms:

- $Gen(\lambda) \rightarrow SK$
- $MAC(SK, m) \rightarrow$ tag σ
- $Verify(SK, m, \sigma) \rightarrow \{0, 1\}$

Correctness is essentially the same as with public key signature schemes: For $SK \leftarrow Gen(\lambda)$, we should have $Verify(SK, m, MAC(SK, m)) = 1$.

3.2 MAC security

3.2.1 Standard unforgeability

A MAC scheme is said to be existentially unforgeable under chosen message attack if any attack algorithm has at most negligible advantage in the following game:

- Challenger gets $SK \leftarrow Gen(\lambda)$.
- Attacker gets oracle access to $MAC(SK, \cdot)$.

  Let $S = \{m_1, \ldots, m_Q\}$ be the attacker's oracle queries ($Q \in poly(\lambda)$).
- Attacker outputs m^*, σ^*.
- Attacker wins if $Verify(SK, m^*, \sigma^*) = 1$.

Note that the attacker doesn't really benefit from oracle access to $Verify(SK, \cdot, \cdot)$ because the MAC oracle is guaranteed to produce correct results and checking a polynomial number of attempted forgeries before submitting them will not increase a negligible advantage to a non-negligible one.

3.2.2 Strong unforgeability

The setup of the strong unforgeability of the game is the same, but instead of S consider $T = \{(m_1, \sigma_1), \ldots, (m_Q, \sigma_Q)\}$, the series of pairs of the attacker's oracle queries and the tags they returned. To win, the attacker must submit any forgery $(m^*, \sigma^*) \notin T$.

Note that this is a stronger notion of security because for forgery (m^*, σ^*), $m^* \notin S \implies (m^*, \sigma^*) \notin T$, so any attacker of the standard unforgeability game can attack the strong unforgeability game.

3.3 Construction of a fast MAC using a PRF

Given $F : \{0,1\}^\ell \to \{0,1\}^n$, build a MAC for messages in $\mathcal{M} = \{0,1\}^\ell$:

- $Gen(\lambda) \to$ Choose random PRF key K
- $MAC(K, m) \to \sigma := F_K(m)$
- $Verify(K, m, \sigma) \to 1$ if $\sigma = F_K(m)$, 0 otherwise

3.4 Structure of proof of security

Use a strategy similar to the proof of security of the PRF-based encryption scheme from the beginning of lecture:

1. Define GameReal to be the unforgeability game with the encryption scheme above; define GameRand to be the unforgeability game with an encryption scheme as above but with random f replaced for PRF F.
2. Show that the advantage of any attacker in GameRand is at most $1/2^n$, or $Q/2^n$ if the attacker is given a $Verify(SK, \cdot, \cdot)$ oracle, but negligible in any case.
3. Show that the advantage of any attacker is not significantly different in GameReal and GameRand by constructing an algorithm $\mathcal{B}$ that harnesses a non-negligible advantage of such an attacker to break the PRF.

CS 388H Introduction to Cryptography 28-Oct-2009

Lecture 18: Constructing a CCA-Secure Encryption Scheme

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Rigor Guidelines

Consider the algorithms for the symmetric key encryption scheme that was proposed last lecture as CPA-secure assuming that its function F was a PRF. Establish that it is IND-CPA-secure based on the PRF property of F using the following structure:

1. State the theorem: eg If F is a PRF family, then the scheme is CPA-secure.
2. Define the games/hybrid experiments your proof will use, if applicable. With games, it's often enough to clearly describe them in terms of their difference from a well-established game: eg Define GameRand to be the CPA challenge with random f used in place of PRF F_K.
3. Provide intuition about how the proof will proceed: eg We will argue that the difference in advantage in the two games is negligible and then that the advantage in GameRand is negligible, thereby showing that the attacker has negligible advantage in GameReal.
4. State your lemma(s): eg For any efficient $\mathcal{A}$, the $\mathcal{A}$'s advantage in GameReal is negligibly different than $\mathcal{A}$'s advantage in GameRand.
5. Proof: eg Assume for counterexample that an $\mathcal{A}$ with non-negligibly different advantages exists. Construct $\mathcal{B}$ to break PRF F. Give your construction/description of $\mathcal{B}$. Analyze $\mathcal{B}$'s advantage. Things like the factor of two converting between $Pr[b' = 1|b = 1] - Pr[b' = 1|b = 0]$ and $Pr[b' = b] - 1/2$ are not super important, but terms like $Q/2^n$ are important to call out explicitly before establishing them as negligible (and subsequently ignoring them, as appropriate).

2 Constructing a CCA-Secure Encryption Scheme

2.1 Re-examine previous symmetric encryption scheme (Lectures 16-17)

Given OWF family $F : \{0,1\}^\ell \to \{0,1\}^n$, define:

- $Gen(\lambda) \to$ Choose random K
- $Enc(K, m \in \{0,1\}^n) \to$ Choose random $r \in \{0,1\}^\ell$; $CT := (r, F_K(r) \oplus m)$.
- $Dec(K, (c_1, c_2)) \to$ Decrypt $m = c_2 \oplus F_K(c_1)$.

We showed this was CPA-secure by using a CPA attacker to break F; is it CCA-secure? CCA-security is a valid real-world concern because attackers can often get a lot of information just from seeing whether a decrypt algorithm fails on certain ciphertexts.

CCA-secure systems allow attackers to query the decryption oracle on any ciphertexts other than their challenge ciphertext. Suppose an attacker is given an encryption $CT^* = (r, F_K(r) \oplus m_b$. The attacker can request decryption on $CT' = CT^*$ with the last bit flipped. Flip the last bit of the returned message, and that gives m_b, making it easy for the attacker to guess b.

2.2 Using MACs for CCA-Secure Encryption

Implicit in the definition of the algorithms comprising a symmetric key encryption scheme is the idea that only trusted parties with access to the secret key should be allowed to produce ciphertexts. It seems reasonable that we would want these ciphertexts to be MACed so decrypt will reduce the possibility of bogus decryption oracle queries by explicitly failing on anything that is not MACed properly.

2.2.1 First proposal: Encrypt and MAC

Given an unforgeable MAC scheme $(Gen_{MAC}, MAC, Verify)$ and a CPA-secure symmetric encryption scheme $(Gen_{CPA}, Enc_{CPA}, Dec_{CPA})$, define another symmetric key encryption scheme as follows:

- $Gen(\lambda) \rightarrow SK := (K_M \leftarrow Gen_{MAC}, K_E \leftarrow Gen_{CPA})$
- $Enc(SK, m) \rightarrow CT := (c_1 \leftarrow Enc_{CPA}(K_E, m), c_2 \leftarrow MAC(K_M, m))$
- $Dec(SK, CT) \rightarrow$ If $Verify(K_M, Dec_{CPA}(K_E, c_1), c_2)$ then output m, else output $\perp$

Is this CCA-secure? Note that both CPA and CCA-security of encryption schemes deal with preventing an attacker from getting information about the message, but MAC unforgeability ensures only that attackers can't trick the Verify algorithm, making no claims about the ease of distinguishing between the tags of two messages.

To see this more concretely, consider a secure scheme MAC$'$. Build MAC as a modified version of MAC$'$ that adds the message to the end of the tag, and the Verify algorithm ignores those bits. The new scheme is a legitimate MAC scheme, but the tags clearly do not hide the message.

Since the above encryption scheme exposes the tag of m, it would not even be CPA secure with the MAC scheme described above.

2.3 Second proposal: MAC *then* Encrypt

This time we try a scheme that hides the MAC tag so we don't have to worry that CPA/CCA attackers will be able to read the message from the ciphertext.

- $Gen(\lambda) \to SK := (K_M \leftarrow Gen_{MAC}, K_E \leftarrow Gen_{CPA})$
- $Enc(SK, m) \to$

  Let $x \leftarrow MAC(K_M, m)$

  Set $CT := Enc_{CPA}(K_E, (x, m))$
- $Dec(SK, CT) \to$

  Let $(x, m) \leftarrow Dec_{CPA}(K_E, CT)$

  If $MAC(K_M, m) = x$ then output m, else output $\perp$

This looks CPA secure because x, which we previously saw could potentially give away m, is hidden. Is there a way to manipulate CT while keeping (x, m) in tact so decrypt will actually work?

Consider a secure scheme CPA$'$ whose encrypt algorithm outputs some CT. Define scheme CPA where its algorithm takes the CPA$'$ encryption of a message and tacks on a random bit. Decrypt just ignores the last bit and otherwise works like the CPA$'$ decryption algorithm.

If we use this CPA scheme in our scheme above, then we could just flip the random bit at the end and query the decrypt oracle to get m.

2.4 Third proposal: Encrypt then MAC

- $Gen(\lambda) \to SK := (K_M \leftarrow Gen_{MAC}, K_E \leftarrow Gen_{CPA})$
- $Enc(SK, m) \to CT := (c_1 \leftarrow Enc_{CPA}(K_E, m), c_2 \leftarrow MAC(K_M, c_1))$
- $Dec(SK, CT) \to$ If $Verify(K_M, c_1, c_2)$ output $m \leftarrow Dec_{CPA}(K_E, c_1)$, else output $\perp$

This proposal is CCA-secure. To prove this, define Game CCA to be the standard IND-CCA game and define Game SetReply to be a simulation of the CCA game where the decrypt oracle given to the attacker is modified. The simulation keeps track of $S =$ the set of (c_1, c_2) pairs produced by the attacker's queries to the encrypt oracle. If the decrypt oracle is queried on $(c_1, c_2) \in S$, it replies with the message that was used to produce the encryption. Otherwise, it returns a fail.

If we prove the security of the SetReply game, how will this relate to the security of the real game?

Claim 2.1 *For all efficient $\mathcal{A}$ that make at most Q queries, if MAC is strongly unforgeable, then $|Adv_{\mathcal{A},CCA} - Adv_{\mathcal{A},SetReply}| \in negl(\lambda)$.*

To prove this, we construct an algorithm $\mathcal{B}$ from an attacker $\mathcal{A}$ that violates the above claim and show that $\mathcal{B}$ forges a MAC tag.

1. $\mathcal{B}$ is given MAC oracle access.
2. $\mathcal{B}$ chooses K_E for the CCA encryption simulation.

3. $\mathcal{A}$ queries the encrypt and decrypt oracles, and $\mathcal{B}$ handles them as follows:

   When $\mathcal{A}$ queries for the encryption of m, $\mathcal{B}$ gets c_1 by running $Enc_{CPA}(K_E, m)$ and gets c_2 as result of query to MAC oracle for tag of c_1

   When $\mathcal{A}$ queries for the decryption of (c_1', c_2'), $\mathcal{B}$ returns the previous value of m if $(c_1', c_2') \in S$. Otherwise, ask the verify oracle if c_2 is a valid tag of c_1. If not, return a fail. If it does, then submit this as the MAC-breaking forgery.

Unless $\mathcal{A}$ queries on $(c_1', c_2') \notin S$, $\mathcal{B}$ simulates the SetReply game. But $\mathcal{A}$ is given to have a non-negligibly better advantage in CCA than in SetReply, so it must make such a query with non-negligible probability.

To show the second claim, that $\mathcal{A}$'s advantage in the SetReply game is negligible, we note that the SetReply game basically reduces to the CPA game with some additional book-keeping. Intuitively, this should work out since the encryption scheme is based on a known CPA-secure encryption scheme.

CS 388H Introduction to Cryptography 2-Nov-2009

Lecture 19: Midterm Review

Instructor: Brent Waters *TA: Sara Krehbiel*

There are no scribe notes for the lecture that was devoted to the midterm review, when we discussed topics for the midterms and practice problems.

CS 388H Introduction to Cryptography 9-Nov-2009

Lecture 20: Bit Commitment

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Agenda

- Today: Bit commitment
- Wednesday: Lecture by Vitaly Shmatikov
- Next week: Zero knowledge
- Later: Lossy trapdoor functions

2 Bit Commitment: Intuition and Protocols

Basic idea:

1. One party chooses a bit, puts it in a lock box, keeps the key.
2. Other party receives the box but not the key.
3. Box can be unlocked with the key at a later point.

Applications:

- Two people can remotely simulate flipping a coin by each choosing a bit, sending them in locked boxes, and simultaneously revealing and XORing them.
- This can be extended to encompass more general knowledge commitment: put a prediction about a future event in a lock box.

Bit commitment schemes are characterized by two protocols:

1. $S_C(b, \lambda) \leftrightarrow R_C(\lambda)$

   Commitment c is a common output; witness w is output to sender only.

2. $S_R(c, w, b) \leftrightarrow R_R(c)$

   Receiver outputs accept and b or reject.

The protocols are said to be correct iff the receiver accepts the behavior of an honest sender.

3 Bit Commitment Security Properties

3.1 Hiding: Sender's security

Hiding is maintained if for all efficient malicious receivers R_C^* that outputs a guess of the bit chosen during the commit protocol,

$$Pr[S_C(b=1) \leftrightarrow R_C^* = 1] - Pr[S_C(b=0) \leftrightarrow R_C^* = 1] = negl(\lambda)$$

3.2 Soundness/binding: Receiver's security

A protocol is sound if all efficient malicious attackers S_R^* have a negligible chance of winning the following game:

1. Challenger runs R_C with S_C^*; common output c is revealed and attacker can save w.
2. S_C^* executes the reveal protocol with R_C twice using the same c and both values of b; wins if both values of b are accepted.

3.3 Perfect hiding and binding

Statistically secure hiding. Statistically secure hiding is when unbounded adversaries have at most negligible advantage.

Perfect hiding. In a perfect hiding scheme, all unbounded receiver adversaries have zero advantage in determining b.

Perfect binding. In a perfectly binding scheme, all unbounded sender adversaries have zero advantage in getting both values of b accepted.

Claim 3.1 *A bit commitment protocol cannot simultaneously be perfectly hiding and perfectly binding.*

Proof. Assume perfect binding. Then for an honest run of the commit protocol, it is only possible to convince the receiver to accept a reveal of either 0 or 1 (not both). An unbounded receiver could iterate through all possibilities for randomness associated with either input and determine which input generated the commitment. $\square$

4 Pederson Commitment

Commit $S_C(b, \lambda) \leftrightarrow R_C(\lambda)$

1. Receiver chooses group G of prime order p with $|p| \approx \lambda$ and two generators g and h. Receiver cares that $g \neq 1$ (which would destroy binding). Give (G, g, h) to sender.
2. Sender checks $h \neq 1$ (this would destroy hiding).
3. Sender chooses $r \in \mathbb{Z}_p$. Computes $\sigma = g^b h^r$ (r is the witness). Sends to receiver.

4. Commitment $c = ((G, g, h), \sigma)$.

Reveal $S_R(b, c, r) \leftrightarrow R_R(c)$

1. S_R gives receiver $r \in \mathbb{Z}_p, b \in \{0, 1\}$.
2. Receiver checks that $\sigma = g^b h^r$. If so, accept b; else, reject.

The Pederson bit commitment protocol is perfectly hiding. Proof: With r random and $h \neq 1$, the distribution of σ is independent of b. Formally:

$$\forall x \in G, Pr[\sigma = x | b = 0] = Pr[\sigma = x | b = 1]$$

. This means that b is information theoretically secure. (Additionally, all values of x are equally likely, but this isn't required for perfect hiding.)

The Pederson bit commitment protocol is binding. Proof: Assume there exists some efficient malicious sender $\mathcal{A}$ that breaks the commitment, and build $\mathcal{B}$ to break DL:

1. $\mathcal{B}$ gets DL challenge $G, g, h(= g^a)$.
2. $\mathcal{B}$ starts commit by giving G, g, h.
3. $\mathcal{A}$ gives σ to $\mathcal{B}$.
4. $\mathcal{B}$ checks that $\mathcal{A}$ produces two effective reveals $(b = 0, r), (b = 1, r')$.
5. Then $\sigma = h^r = gh^{r'} \rightarrow g = h^{r-r'}$, so $\mathcal{B}$ breaks DL with $a = (r - r')^{-1} \bmod p$.

Note the similarities to the argument we used earlier for proving security of CRHF families.

CS 388H Introduction to Cryptography 11-Nov-2009

Lecture 21: Guest Lecture

Instructor: Brent Waters *TA: Sara Krehbiel*

Lecture 21 was a guest lecture by Professor Vitaly Shmatikov on security.

CS 388H Introduction to Cryptography 16-Nov-2009

Lecture 22: Interactive Proof Systems and Zero Knowledge

Instructor: Brent Waters *TA: Sara Krehbiel*

1 Interactive Proof Systems

1.1 Motivation for zero knowledge systems

One situation in which zero knowledge schemes are useful is when you want to prove the effectiveness of an algorithm without revealing it. For example, Netflix recently offered a \$1M prize to the person that could produce the best preference-predicting algorithm. To participate in this competition, you may have wanted a way to prove that your algorithm works without exposing it before getting your prize.

1.2 Interactive proof systems

An interactive proof system is a protocol between an unbounded prover P (sometimes called a wizard) and a poly-time verifier V that accepts or rejects inputs x as members of a given language L.

1.3 Completeness

$< P, V >$ is complete if for all inputs $x \in L$, $Pr[< P, V > (x) = accept] = 1 - negl(k)$.

1.4 Soundness

$< P, V >$ is sound if for every potentially malicious prover P^* and any input $x \notin L$, $Pr[< P^*, V > (x) = accept] = negl(k)$.

1.5 IP languages

IP is the class of languages L that have interactive proof systems.

2 Example: Graph Isomorphism

Definition 2.1 *Let $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$. G_0 and G_1 are **isomorphic** if and only if there exists a permutation $\pi : [1 \ldots n] \rightarrow [1 \ldots n]$ such that for all $i, j \in \{1, \ldots, n\}$, $(i, j) \in E_0 \leftrightarrow (\pi(i), \pi(j)) \in E_1$.*

Consider the language of pairs of isomorphic graphs: $L = \{(G_0, G_1) | G_0 \text{ and } G_1 \text{ are isomorphic}\}$. How could an unbounded prover P prove to an efficient verifier V that $(G_0, G_1) \in L$?

2.1 Interactive proof system for graph isomorphism

P enumerates through all permutations and provides the π such that $\pi(G_0) = G_1$. V checks that for all $i, j \in \{1, \ldots, n\}, (i, j) \in E_0 \leftrightarrow (\pi(i), \pi(j)) \in E_1$ (in time linear with the number of edges) . If so, accept, else reject.

Completeness holds because if the graphs are isomorphic, a permutation exists so P will find it in an exhaustive search of the $n!$ permutations.

Soundness holds because if the graphs are not isomorphic, no verifiable permutation exists for P to choose from.

2.2 Graph non-isomorphism

How could a prover convince a verifier with high probability that a two graphs are not isomorphic? Two observations:

- G_0, G_1 not isomorphic means that for any G', G' is isomorphic to at most 1 of G_0, G_1.
- If an unbounded prover is shown $G' = \pi(G_\sigma)$ for a random permutation π and bit σ, it will not be able to guess σ with probability greater than 1/2. This is because there exists permutation π' such that $G' = \pi'(G_{1-\sigma})$, and this permutation-bit combination was equally likely to have been chosen. This argument is information theoretic (the distribution of G' is not affected by the value of σ), so P's unboundedness can't help.

Now consider a protocol where V challenges P k times as follows:

1. V chooses random σ, π and sends $\pi(G_\sigma)$ to P
2. P checks whether $\pi(G_\sigma)$ is isomorphic to G_0 or G_1 and returns σ corresponding to the isomorphic graph.
3. If correct, V continues with a fresh challenge. If incorrect, V rejects G_0, G_1 as not in the language of non-isomorphic graphs.

The protocol is complete because if the graphs are not isomorphic, the unbounded prover will always determine which was permuted by comparing G' to all permutations of each graph. Only one of the graphs will have a match, revealing σ. The verifier will accept the correct σ all k times.

The protocol is sound because the second observation that σ is information theoretically hidden from the prover if the graphs are isomorphic ensures that the prover will be right just 1/2 the time. The probability of being right k times is $1/2^k \in negl(k)$.

2.3 Witnesses

Note that in system checking whether graphs are isomorphic, we could have required P to be bounded but given the witness π so P didn't have to iterate through the $n!$ possibilities. In graph non-isomorphism, we rely on iterated experiments because we don't know of a similarly short witness to establish non-isomorphism.

3 Zero Knowledge

A zero knowledge system is an interactive proof system in which P and V must both be efficient, but P is additionally given a witness w. For example, say P is supposed to provide V with the prime factorization of an integer N. If we give P witness $w = p, q$ both prime with $N = pq$, then P can provide a verifiable answer efficiently.

Completeness and soundness are as before. Note that soundness protects the verifier against a cheating prover. In zero knowledge schemes we require an additional property that ensures a prover that a nosy verifier will not learn anything other than what the prover gives it explicitly.

3.1 Zero knowledge property (Goldwasser, Micali, Rackoff definition)

A proof system $< P(w), V >$ satisfies the **zero knowledge** property if and only if for every efficient cheating verifier V^* and any $x \in L$, there exists an efficient M that simulates the protocol using access to the code of V^* and the transcript of the $< P(w), V^* >$ protocol but does not have access to the witness or P's code such that the distribution of the view (transcript of interactions and coins of V^*) of $< P(w), V^* > (x)$ is indistinguishable from $M(x)$'s outputs.

This is the same as saying that for every poly time distinguisher $\mathcal{D}$ that outputs 1 when it thinks it has seen the transcript of a protocol (where the prover had the witness) and 0 otherwise, it must be that $Pr[\mathcal{D}(< P(w), V^* > (x)) = 1] - Pr[\mathcal{D}(M(x)) = 1] = negl$.

4 3-Coloring

Later we will see a zero-knowledge protocol for 3-coloring. This is important because 3-coloring is a known NP-complete problem, so it implicitly gives us a zero-knowledge proof system for every NP problem, including factoring, etc.

CS 388H Introduction to Cryptography 18-Nov-2009

Lecture 23: Zero Knowledge for Graph Isomorphism

Instructor: Brent Waters *TA: Sara Krehbiel*

Last time we gave a protocol for deciding whether graphs were isomorphic. It involved a prover P providing a witness π to V, which gave the verifier knowledge. This time, we will come up with a zero-knowledge system for graph isomorphism.

1 Zero Knowledge for Graph Isomorphism

Recall the definition of zero knowledge for a language L: for all V^* and $x \in L$ there exists an M such that $M(x)$ is indistinguishable from the view of $< P(w), V^* > (x)$.

Define protocol $< P(\phi), V > (G_0, G_1)$ with $\phi(G_0) = G_1$ to be the following done k times:

1. P chooses random π and sends $F = \pi(G_1)$ to V.

   Note that now P can provide a permutation from either G_0 or G_1 to F:

   $$G_1 \xrightarrow{\pi} F \qquad G_0 \xrightarrow{\phi} G_1 \xrightarrow{\pi} F$$

2. V chooses random $b \in \{0, 1\}$.
3. If $b = 1$, P gives π. If $b = 0$, P gives $\phi \circ \pi$.
4. V checks that the received permutation takes G_b to F. If yes, continue, else quit.

The protocol is clearly complete because G_0, G_1 isomorphic means P can always give a correct permutation. Soundness holds because if G_0, G_1 are not isomorphic, P will be able to provide permutation π when V chooses $b = 1$, but no permutation exists for $b = 0$ because F and G_0 are not isomorphic. The probability of winning k times is the probability of V randomly picking $b = 1$ k times in a row which is $1/2^k \in negl(k)$.

2 Proof that Graph Isomorphism Protocol is Zero Knowledge

For the zero knowledge proof, describe a simulator M for V^*. For each round:

1. Record the state of V^* (you can think of this as the configuration of the turing machine representing V^* or a snapshot of a virtual machine)
2. Choose a random b', π' and send $F = \pi'(G_{b'})$ to V^*.
3. Get request b from V^*.
4. If $b = b'$, give π' to V^*. Else restart V^* at its previous state and go back to step 1.

We can say this runs in expected poly-time or we can stipulate that we quit (with negligible probability $1/2^k$ if we fail k times in a row.

2.1 Analysis

We've established that $(G_0, G_1) \in L$ means F is distributed independently of b', so b' is information theoretically secure. Thus, at each try, we have has exactly 1/2 chance of success (when V^* picks $b = b'$). Then in the entire simulation, there is a negligible chance of quitting, and when M doesn't quit, it outputs permutations distributed exactly like the permutations given by P in the real protocol. In other words,

$$Pr[\mathcal{D}(< P(\phi), V^* > (x)) = 1] - Pr[\mathcal{D}(M(x)) = 1 | M \text{ doesn't quit}] = 0$$

. We want that for every $\mathcal{D}$,

$$Pr[\mathcal{D}(< P(\phi), V^* > (x)) = 1] - Pr[\mathcal{D}(M(x)) = 1] = negl(k)$$

Note that

$$\begin{aligned} Pr[\mathcal{D}(M(x)) = 1] &= Pr[\mathcal{D}(M(x)) = 1 | \text{no quit}] Pr[\text{no quit}] \\ &+ Pr[\mathcal{D}(M(x)) = 1 | \text{quit}] Pr[\text{quit}] \end{aligned}$$

Pr[no quit] is negligibly close to 1 and Pr[quit] is negligible, so $\mathcal{D}$ has negligible advantage in distinguishing M from the view of P, V^*.