Lecture 1
Course Description and Introduction to Hashfunctions

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

January 17, 2011

- Course Description
- Oral Presentation
- Hashfunctions
- Random Oracles

Quote of the Day

Cryptography is concerned with the conceptualization, definition, and construction of computing systems that address security concerns.

– Oded Goldreich, Foundations of Cryptography, 1997

Applications of Cryptography

Historically.

- Military and diplomatic secret communication.
- Communication between banks, e.g., credit card transactions.

Modern.

- Protecting satellite TV from leaching.
- Secrecy and authenticity on the Internet, mobile phones, etc.
- Credit cards.

Applications of Cryptography

Today.

- Distributed file systems, authenticity of blocks in bit torrents, anonymous remailers, Tor-network, etc.
- RFID tags, Internet banking, Försäkringskassan, Skatteverket, "e-legitimation".

Future.

- Secure distributed computing (multiparty computation): election schemes, auctions, secure cloud computing, etc.
- Variations of signatures, cryptosystem, and other primitives with special properties, e.g., group signatures, identity based encryption, etc.

Course Description

Goal

The goal of the course is to

- give an overview of modern cryptography

in order that students should

- know how to evaluate and, to some extent, create cryptographic constructions, and
- to be able to read and to extract useful information from research papers in cryptography.

Prerequisites

- *DD1352 Algorithms, data structures and complexity*, or *DD2354 Algorithms and complexity*.
- Knowledge of mathematics and theory of algorithms corresponding to the required courses of the D or F-programmes at KTH.

Tentative Plan of Lectures (1/2)

L1-L2. Administration, introduction, classical cryptography, security notions of hashfunctions, random oracles, iterated constructions, SHA-1, universal hashfunctions.

L3-L4. Symmetric ciphers, perfect secrecy, entropy, substitution-permutation networks, linear cryptanalysis.

L5-L6. AES, Feistel networks, Luby-Rackoff, DES, modes of operations, DES-variants.

L7-L8. Elementary number theory, public-key cryptography, RSA, primality testing, textbook RSA, semantic security.

Tentative Plan of Lectures (2/2)

L9-L10. RSA in ROM, Rabin, discrete logarithms, Diffie-Hellman, El Gamal.

L11-L12. Message authentication codes, identification schemes, signature schemes, PKI, elliptic curve cryptography.

L13. Pseudorandom generators.

L14. Guest lecture of Mats Näslund, Senior Specialist, Ericsson Research and Tekn Dr, NADA KTH 1998.

L15. Make-up time, special topic.

Course Requirements

- **Know the Rules.** Read and understand:
  `http://www.kth.se/csc/student/hederskodex`
  `http://www.csc.kth.se/DD2448/krypto11/rules`
- **Oral Presentations. a)** Choose a SHA-3 candidate. **b)** summarize the paper in a 12-min oral presentation. Gives P-points ($P = 0$ or $30 \leq P \leq 80$).
- **Homework 1-2.** Each homework is a set of problems giving B and H-points ($B \geq 40$ and $B + H \geq 100$).
- **Oral Exam.** Purpose is to give a fair grade. Discussion starting from submitted solutions. Gives (possibly negative) B and H-points and a single O-point if passed.

Deadlines

The deadlines in this course are strict. Late solutions are awarded zero points.

- **Oral Presentation.** January 31 – February 4 (there will be times to book on the homepage).
- **Homework 1.** Thursday, February 17, 10:00.
- **Homework 2.** Thursday, March 10, 10:00.
- **Oral Exam.** March 22 – March 25 (there will be times to book on the homepage).

Grading

To earn a grade the requirements of all lower grades must be satisfied as well, with $T = B + H + P + O$.

Grade	Requirements
E	$B \geq 60$, $P \geq 30$, and $O \geq 1$.
D	$T \geq 120$.
C	$T \geq 140$ and $P \geq 50$.
B	$T \geq 170$.
A	$T \geq 200$ and $P \geq 70$.

Register For the Course

Send an email to `dog@csc.kth.se` with subject `Krypto11_Register` and a body using the following format.

Format	Example
`<Firstname(s)>`	`Eva Stina`
`<Lastname(s)>`	`Guvendiren Olsson`
`<Email address(s)>`	`eva@yagoo.se eva@guvendiren.tr`
`<Personal number>`	`830121-1234`
`<Study status>`	`F-07`

Latex

- Latex is the standard typesetting tool for mathematics.
- It is the fastest way to produce mathematical writing. We recommend using it.
- The best way to learn it is to read `http://tobi.oetiker.ch/lshort/lshort.pdf`

Oral Presentation

Hashfunction

A hashfunction maps arbitrarily long bit strings into bit strings of fixed length.

The output of a hashfunction should be "unpredictable".

Wish List

- Finding a preimage of an output should be hard.
- Finding two inputs giving the same output should be hard.
- The output of the function should be random.
- Truncating should only reduce the output size.

etc

Standardized Hashfunctions

Despite that theory says it is impossible, in practice people simply live with **fixed** hashfunctions and use them as if they are randomly chosen functions.

SHA

- Secure Hash Algorithm (SHA-0,1, and the SHA-2 family) are hashfunctions standardized by NIST to be used in, e.g., signature schemes and random number generation.
- SHA-0 was **weak** and withdrawn by NSA. SHA-1 was **withdrawn** 2010. SHA-2 family is based on similar ideas but seems safe so far...
- All are **iterated** hashfunctions, starting from a basic **compression function**.

SHA-3

- NIST is running an open competition for the next hashfunction, named SHA-3. Several groups of famous researchers have submitted proposals.
- Call for SHA-3 explicitly asked for "different" hashfunctions. You will tell us all about the candidates!
- It might be a good idea to read about SHA-1 for comparison...

Oral Presentation

- Every student gives a 12 min talk.
- Up to 6 talks in 2 hour-sessions. Book a time on the homepage (soon).
- You listen to the other talks of your session.
- Timing, audience, tools,... Plan and rehearse your talk!
- `http://www.csc.kth.se/utbildning/kth/kurser/DD2448/krypto11/rules`

Hashfunctions

Negligible Functions

Definition. A function $\epsilon(n)$ is negligible if for every constant $c > 0$, there exists a constant n_0, such that

$$\epsilon(n) < \frac{1}{n^c}$$

for all $n \geq n_0$.

Motivation. Events happening with negligible probability can not be exploited by polynomial time algorithms! (they "never" happen)

Definitions

- $\{0,1\}^*$ denotes the set of all finite bit strings, i.e., the set
$$\cup_{n=1}^{\infty}\{0,1\}^n \ .$$
- A function $f : \{0,1\}^* \to \{0,1\}^*$ is polynomial time computable, if there exists an algorithm A with running time bounded by a polynomial $p(\cdot)$, that given $x \in \{0,1\}^*$ computes $f(x)$ in time $p(|x|)$.
- A Turing machine M with an oracle $O(\cdot)$ is a Turing machine which may do a special query transition. Before the transition it writes a query x in a designated area of its tape and after the transition $O(x)$ appears on another designated area of its tape. This is written $M^{O(\cdot)}$ or M^{O}.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial time computable function.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

$$f_n : \{0,1\}^n \rightarrow \{0,1\}^*$$

by setting $f_n(x) = f(x)$.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

$$f_n : \{0,1\}^n \to \{0,1\}^*$$

  by setting $f_n(x) = f(x)$.
- Note that we may recover f from the ensemble by $f(x) = f_{|x|}(x)$.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

$$f_n : \{0,1\}^n \rightarrow \{0,1\}^*$$

  by setting $f_n(x) = f(x)$.
- Note that we may recover f from the ensemble by $f(x) = f_{|x|}(x)$.
- When convenient we give definitions for a function, but it can be turned into a definition for an ensemble.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}$$

for some polynomials $l(n)$ and $l'(n)$.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \rightarrow \{0,1\}^{l'(n)}$$

  for some polynomials $l(n)$ and $l'(n)$.

- Here n is the security parameter and α is a "key" that is chosen randomly.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}$$

  for some polynomials $l(n)$ and $l'(n)$.

- Here n is the security parameter and α is a "key" that is chosen randomly.

- We may also view F as an ensemble $\{f_\alpha\}$, where $f_\alpha = \{f_{n,\alpha_n}\}_{n\in\mathbb{N}}$ and $\alpha = \{\alpha_n\}_{n\in\mathbb{N}}$.

Ensembles of Functions (3/3)

These conventions allow us to talk about a "random function" f in several convenient ways.

Now you can forget that and assume that everything works!

One-Wayness

Definition. An function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is said to be **one-way**[1] if for every polynomial time algorithm A and a random x

$$\Pr[A(f(x)) = x' \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Normally f is computable in polynomial time in its input size.

[1] "Enkelriktad" på svenska **inte** "enväg".

One-Wayness

Definition. An function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be **one-way**[1] if for every polynomial time algorithm A and a random x

$$\Pr[A(1^n, f(x)) = x' \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Normally f is computable in polynomial time in its input size.

[1] "Enkelriktad" på svenska **inte** "enväg".

Second Pre-Image Resistance

Definition. A function $h : \{0,1\}^* \rightarrow \{0,1\}^*$ is said to be **second pre-image resistant** if for every polynomial time algorithm A and a random x

$$\Pr[A(x) = x' \wedge x' \neq x \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Note that A is given not only the output of f, but also the **input** x, but it must find **another** preimage.

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if for every polynomial time algorithm A and randomly chosen α

$$\Pr[A(\alpha) = (x, x') \wedge x \neq x' \wedge f_\alpha(x') = f_\alpha(x)] < \epsilon(n)$$

for a negligible function ϵ.

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if for every polynomial time algorithm A and randomly chosen α

$$\Pr[A(\alpha) = (x, x') \wedge x \neq x' \wedge f_\alpha(x') = f_\alpha(x)] < \epsilon(n)$$

for a negligible function ϵ.

An algorithm that gets a small "advice string" for each security parameter can easily hardcode a collision for a fixed function f, which explains the random index α.

Relations Between Notions

- Collision resistance implies pre-image resistance.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.
- Second pre-image resistance implies one-wayness.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- Second pre-image resistance implies one-wayness.
  1. Given random x, compute $y = f(x)$.
  2. Request preimage x' of y.
  3. Repeat until $x' \neq x$, and output x'.

Random Oracle As Hashfunction

A random oracle is simply a randomly chosen function with appropriate domain and range.

A random oracle is the **perfect** hashfunction. Every input is mapped **independently** and **uniformly** in the range.

Let us consider how a random oracle behaves with respect to our notions of security of hashfunctions.

Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a preimage, i.e., it queries the random oracle on its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(H(x)) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^q \quad .$$

Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a preimage, i.e., it queries the random oracle on its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(H(x)) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^q \quad .$$

Proof. Each query x' satisfies $H(x') \neq H(x)$ independently with probability $1 - \frac{1}{|Y|}$.

Second Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second preimage, i.e., it queries the random oracle on the input and its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(x) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^{q-1} \quad .$$

Second Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second preimage, i.e., it queries the random oracle on the input and its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(x) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^{q-1} .$$

Proof. Same as preimage case, except we must waste one query on the input value to get the target in Y.

Collision of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a collision, i.e., it queries the random oracle on its outputs.

Theorem. Let $H : X \to Y$ be a randomly chosen function. Then for every such algorithm A making q oracle queries

$$\begin{aligned}\Pr[A^{H(\cdot)} = (x, x') \wedge x \neq x' \wedge H(x) = H(x')] &\leq 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{|Y|}\right) \\ &\leq \frac{q(q-1)}{2|Y|} \enspace .\end{aligned}$$

Collision of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a collision, i.e., it queries the random oracle on its outputs.

Theorem. Let $H : X \to Y$ be a randomly chosen function. Then for every such algorithm A making q oracle queries

$$\begin{aligned}\Pr[A^{H(\cdot)} = (x, x') \wedge x \neq x' \wedge H(x) = H(x')] &\leq 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{|Y|}\right) \\ &\leq \frac{q(q-1)}{2|Y|} \enspace .\end{aligned}$$

Proof. $1 - \frac{i-1}{|Y|}$ bounds the probability that the ith query does not give a collision for any of the $i - 1$ previous queries, conditioned on no previous collision.