DD2520 Applied Cryptography
Lecture 3

Douglas Wikström
KTH Royal Institute of Technology
`dog@kth.se`

January 21, 2024

Non-cryptographic Hash Functions

Recall how a hash table is constructed.

- An array D indexed by keys K.
- A huge set T of potential objects that may be stored in D.
- A **hash function** $h : T \to K$ that computes the key $h(t)$ of any object t to be stored.

Non-cryptographic Hash Functions

Recall that we need the following for a hash table to work as intended.

- **The function *h* must be exceptionally simple.** Thus, it may be easy to find a collision, and *h* is nothing like a randomly chosen function.

Non-cryptographic Hash Functions

Recall that we need the following for a hash table to work as intended.

- **The function h must be exceptionally simple.** Thus, it may be easy to find a collision, and h is nothing like a randomly chosen function.
- **The distribution of stored objects is not malicious.** Then h distributes objects “smoothly” over K.

Non-cryptographic Hash Functions

Recall that we need the following for a hash table to work as intended.

- **The function h must be exceptionally simple.** Thus, it may be easy to find a collision, and h is nothing like a randomly chosen function.
- **The distribution of stored objects is not malicious.** Then h distributes objects "smoothly" over K.
- **The size $|K|$ of the table is not large.** Thus, due to the birthday paradox we cannot expect to avoid collisions, but we do not care as long as they are evenly distributed.

If assumptions are violated we go from $O(1)$ to $O(\log n)$ lookups or memory is wasted.

Cryptographic Hash Function

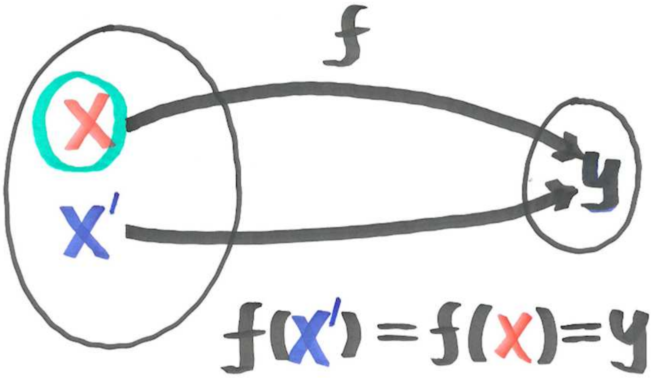
A **(cryptographic) hash function** maps arbitrarily long bit strings into bit strings of fixed length.

The output of a hash function should be “unpredictable”.

Wish List

- Finding a pre-image of an output should be hard.
- Finding two inputs giving the same output should be hard.
- The output of the function should be “random”.

Pre-image Problem



RANDOM	SECRET	COMPUTED	PUBLIC

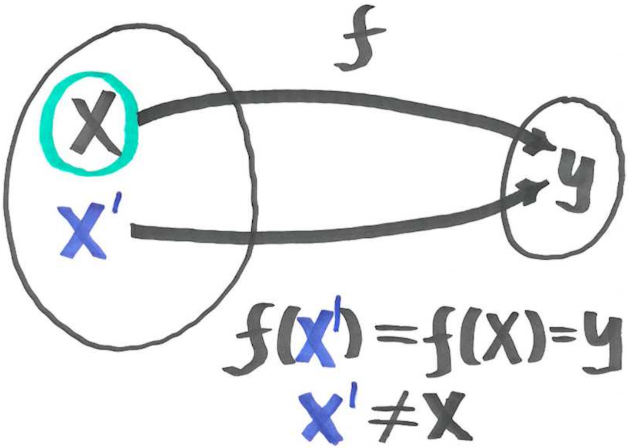
One-Wayness

Definition. A function f on bit strings is said to be **one-way**[1] if given $f(x)$ for a random x it is infeasible[2] to compute x' such that $f(x) = f(x')$.

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

[2] This means that we are convinced that it is impossible in practice.

Second Pre-image Problem

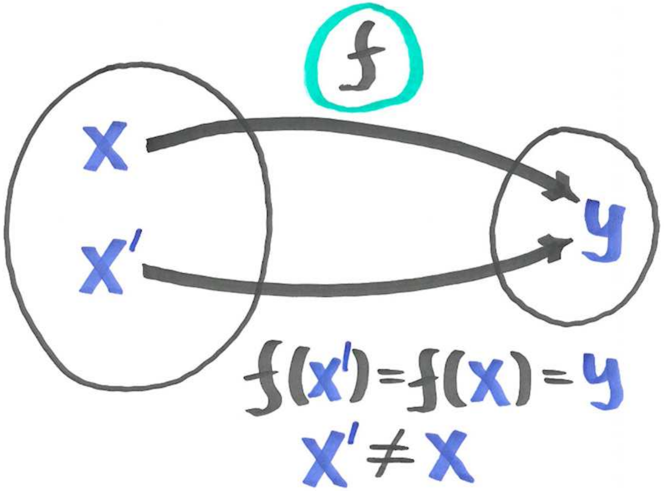


RANDOM	COMPUTED	PUBLIC

Second Pre-image Resistance

Definition. A function h on bit strings is said to be **second pre-image resistant** if given a random x it is infeasible to compute $x' \neq x$ such that $f(x') = f(x)$.

Collision Problem



RANDOM	COMPUTED	PUBLIC

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if given a random α it is infeasible to compute $x \neq x'$ such that $f_\alpha(x') = f_\alpha(x)$.

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if given a random α it is infeasible to compute $x \neq x'$ such that $f_\alpha(x') = f_\alpha(x)$.

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.

Birthday Paradox and Hash Functions

Suppose that the range of a function f is R and let $r = |R|$. Then the probability that there is at least one collision for k random inputs is equal to

$$1 - \left(1 - \frac{1}{r}\right)\left(1 - \frac{2}{r}\right)\cdots\left(1 - \frac{k-1}{r}\right) \approx 1 - e^{-k^2/r} \quad .$$

Birthday Paradox and Hash Functions

Suppose that the range of a function f is R and let $r = |R|$. Then the probability that there is at least one collision for k random inputs is equal to

$$1 - \left(1 - \frac{1}{r}\right)\left(1 - \frac{2}{r}\right)\cdots\left(1 - \frac{k-1}{r}\right) \approx 1 - e^{-k^2/r} \quad .$$

When $k = \Omega(\sqrt{r})$ we should **expect** a collision for **any** function!

Relations for Compressing Hash Functions

- If a function is not second pre-image resistant, then it is not collision-resistant.

Relations for Compressing Hash Functions

- If a function is not second pre-image resistant, then it is not collision-resistant.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

Relations for Compressing Hash Functions

- If a function is not second pre-image resistant, then it is not collision-resistant.
    1. Pick random x.
    2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
    3. Output x' and x.

- If a function is not one-way, then it is not second pre-image resistant.

Relations for Compressing Hash Functions

- If a function is not second pre-image resistant, then it is not collision-resistant.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- If a function is not one-way, then it is not second pre-image resistant.
  1. Given random x, compute $y = f(x)$.
  2. Request pre-image x' of y.
  3. Repeat until $x' \neq x$, and output x'.

Random Oracles

Random Oracle As Hash Function

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

A random oracle is the **perfect** hash function. 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 hash functions.

Pre-Image of Random Oracle

We assume with little loss that an adversary always “knows” if it has found a pre-image, 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 algorithm A making q oracle queries

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

Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a pre-image, 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 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 Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second pre-image, 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

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

Second Pre-Image of Random Oracle

We assume with little loss that an adversary always “knows” if it has found a second pre-image, 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

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

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

Collision Resistance of Random Oracles

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 Resistance of Random Oracles

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.

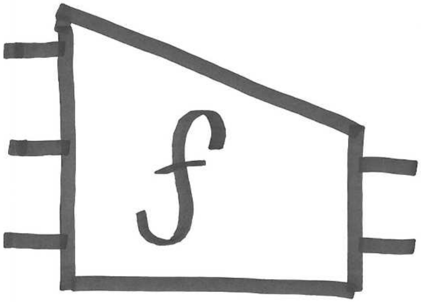
Theorem. Let $H : X \to Y$ be a randomly chosen function. Then for every such algorithm A making q oracle queries

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

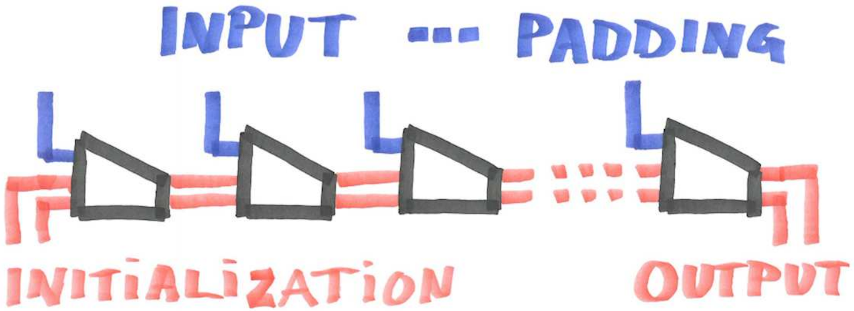
Proof. $1 - \frac{i-1}{|Y|}$ bounds the probability that the ith query does not give a collision for any of the $i-1$ previous queries, conditioned on no previous collision.

Iterated Hash Functions

Compression Function



Merkle-Damgård (1/4)



Merkle-Damgård (2/4)

Suppose that we are given a collision resistant hash function

$$f : \{0,1\}^{n+t} \to \{0,1\}^n \ .$$

How can we construct a collision resistant hash function

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

mapping any length inputs?

Merkle-Damgård (3/4)

Construction.

1. Let $x = (x_1, \ldots, x_k)$ with $|x_i| = t$ and $0 < |x_k| \leq t$.
2. Let x_{k+1} be the total number of bits in x.
3. Pad x_k with zeros until it has length t.
4. $y_0 = 0^n$, $y_i = f(y_{i-1}, x_i)$ for $i = 1, \ldots, k+1$.
5. Output y_{k+1}

Here the total number of bits is bounded by $2^t - 1$, but this can be relaxed.

Merkle-Damgård (4/4)

Suppose A finds collisions in Merkle-Damgård.

- If the number of bits differ in a collision, then we can derive a collision from the last invocation of f.
- If not, then we move backwards until we get a collision. Since both inputs have the same length, we are guaranteed to find a collision.

Standardized Hash Functions

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

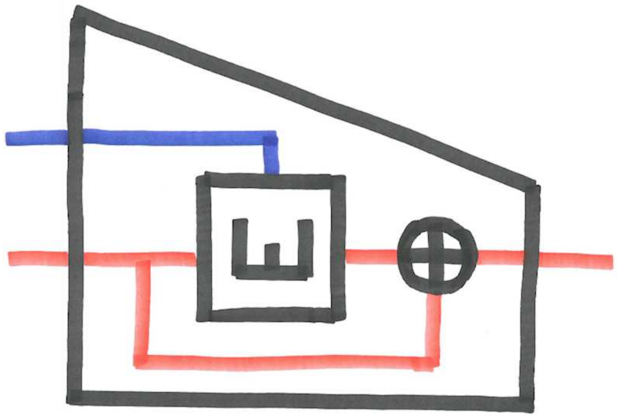
SHA-0,1,2

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

$$f(y_{i-1}, x_i) = \mathsf{E}_{x_i}(y_{i-1}) \oplus y_{i-1}$$

and you know how to build a cipher :-)

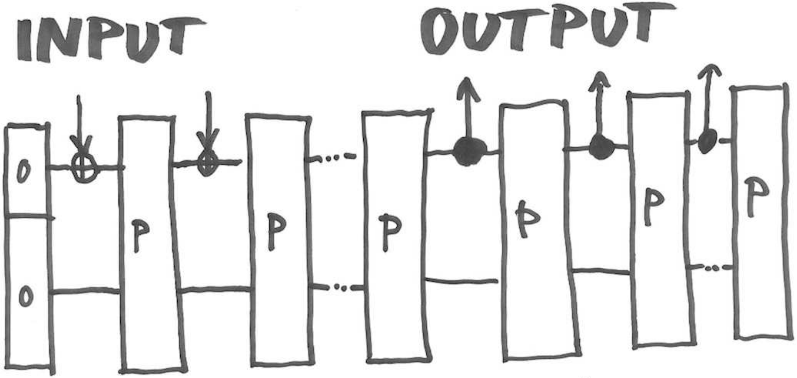
SHA-2



SHA-3

- NIST ran an open competition for the next hash function, named SHA-3. Several groups of famous researchers submitted proposals.
- Call for SHA-3 explicitly asked for “different” hash functions.
- It might be a good idea to read about SHA-1 for comparison.
- The competition ended October 2, 2012, and the hash function **Keccak was selected as the winner**.
- This was constructed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche,

SHA-3 Is a Sponge Function



$P \approx E_k$ WITH FIXED KEY k

DD2520 Applied Cryptography
Lecture 4

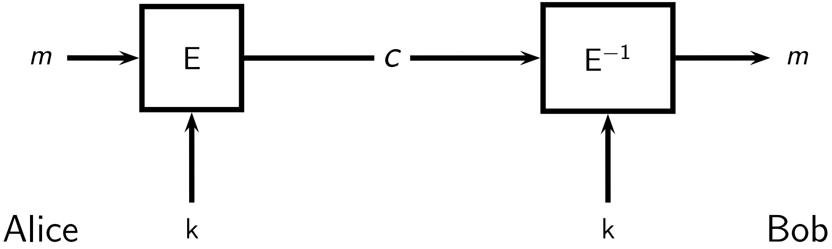
Douglas Wikström
KTH Royal Institute of Technology
dog@kth.se

Jan 23, 2024

Public Key Cryptosystems

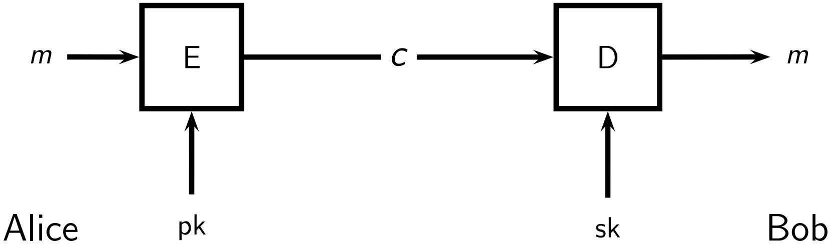
Cipher (Symmetric Cryptosystem)

$c = \mathsf{E}_{\mathsf{k}}(m)$ $m = \mathsf{E}_{\mathsf{k}}^{-1}(c)$



Public-Key Cryptosystem

$c = \mathsf{E}_{\mathsf{pk}}(m)$ $m = \mathsf{D}_{\mathsf{sk}}(c)$



Real-world Analogues of the Functionality

“Anybody can send a private message to Alice.”

- Mailbox (physical security).
- Couriers (physical security and trust in others).
- Separate envelopes (physical security and infeasibility).

Think of additional examples around you!

Merkle's Puzzles

Alice generates a public key.

1. She buys N solved jigsaw puzzles and paints them white.
2. For the ith puzzle she:
    2.1 generates a random cipher key k_i,
    2.2 writes (i, k_i) on the puzzle,
    2.3 puts it back in its box, and
    2.4 shakes the box to break up the puzzle.
3. She randomly re-orders the boxes and ships them to Bob.

Bob encrypts a message m for Alice.

1. He solves a randomly chosen puzzle and find (j, k_j),
2. computes $c = \mathsf{E}_{k_j}(m)$, and
3. sends (j, c) to Alice.

Alice decrypts c and recovers m.

She receives (j, c) and computes $m = \mathsf{E}_{k_j}(c)$.

Analysis of Merkle's Puzzles

We assume that the lth puzzle must be solved completely to recover its payload (l, k_l).

- Alice deals with solved puzzles and Bob solves **one** puzzle.

  This is efficient!
- Eve must solve **half** of the puzzles on average to find (j, k_j).

  This takes time!

This computational asymmetry is what gives asymmetry between the encryptor-decryptor pair, and the adversary, respectively.

Is constant vs. linear asymmetry enough?

History of Public-Key Cryptography

Public-key cryptography was discovered:

- By Ellis, Cocks, and Williamson at the Government Communications Headquarters (GCHQ) in the UK in the early 1970s (not public until 1997).
- Independently by Merkle in 1974 (Merkle's puzzles).
- Independently in its discrete-logarithm based form by Diffie and Hellman in 1977, and instantiated in 1978 (key-exchange).
- Independently in its factoring-based form by Rivest, Shamir and Adleman in 1977.

Preliminaries for RSA

Greatest Common Divisors

Definition. A common divisor of two integers m and n is an integer d such that $d \mid m$ and $d \mid n$.

Definition. A greatest common divisor (GCD) of two integers m and n is a common divisor d such that every common divisor d' divides d.

Packets of milk. Two customers orders m and n packets of milk, respectively. A fixed box size must be used to ship all packets of milk such that each box is completely full and each packet of milk is contained in a box.

What is the greatest size of box that can be used?

Properties

- $\gcd(m, n) = \gcd(n, m)$
- $\gcd(m, n) = \gcd(m - n, n)$ if $m \geq n$
- $\gcd(m, n) = \gcd(m \bmod n, n)$

Packets of milk. The order of the customers does not matter. Subtraction work on completely full boxes. Modular reduction is repeated subtraction.

Euclidean Algorithm

We wish to compute gcd(m, n) for integers m and n such that $m \geq n$.

```
EUCLIDEAN($m$, $n$)
(1)    while $n \neq 0$
(2)        $t \leftarrow n$
(3)        $n \leftarrow m \bmod n$
(4)        $m \leftarrow t$
(5)    return $m$
```

Packets of milk. Euclid's algorithm works on **boxes** of milk and stops when there is a single box left, even if we did not put the packets of milk in boxes to start with!

Bezout's Lemma

Lemma. There exists integers a and b such that

$$\gcd(m, n) = am + bn \ .$$

The extended Euclidian algorithm computes a and b.

Example. If p and q are distinct primes, then there exists integers a and b such that $ap + bq = 1$.

Packets of milk. Execute Euclid's algorithm and keep track of the number of boxes subtracted in each step! The totals give a and b.

Coprimality (Relative Primality)

Definition. Two integers m and n are coprime if their greatest common divisor is 1.

Fact. If a and n are coprime, then there exists a b such that $ab = 1 \bmod n$.

Why is this the case?

The Multiplicative Group

The set $\mathbb{Z}_n^* = \{0 \leq a < n \mid \gcd(a, n) = 1\}$ forms a group, since:

- **Closure.** It is closed under multiplication modulo n.
- **Associativity.** For $x, y, z \in \mathbb{Z}_n^*$:
$$(xy)z = x(yz) \bmod n \ .$$
- **Identity.** For every $x \in \mathbb{Z}_n^*$:
$$1 \cdot x = x \cdot 1 = x \ .$$
- **Inverse.** For every $a \in \mathbb{Z}_n^*$ exists $b \in \mathbb{Z}_n^*$ such that:
$$ab = 1 \bmod n \ .$$

Euler's Phi-Function (Totient Function)

Definition. Euler's Phi-function $\phi(n)$ counts the number of integers $0 < a < n$ relatively prime to n.

Euler's Phi-Function (Totient Function)

Definition. Euler's Phi-function $\phi(n)$ counts the number of integers $0 < a < n$ relatively prime to n.

- Clearly: $\phi(p) = p - 1$ when p is prime.
- Similarly: $\phi(p^k) = p^k - p^{k-1}$ when p is prime and $k > 1$.
- In general: $\phi\left(\prod_i p_i^{k_i}\right) = \prod_i \left(p_i^k - p_i^{k-1}\right)$.

Euler's Phi-Function (Totient Function)

Definition. Euler's Phi-function $\phi(n)$ counts the number of integers $0 < a < n$ relatively prime to n.

- Clearly: $\phi(p) = p - 1$ when p is prime.
- Similarly: $\phi(p^k) = p^k - p^{k-1}$ when p is prime and $k > 1$.
- In general: $\phi\left(\prod_i p_i^{k_i}\right) = \prod_i \left(p_i^k - p_i^{k-1}\right)$.

Example. For a product $N = pq$, where p and q are prime and $p \neq q$, we have $\phi(N) = (p-1)(q-1)$.

Fermat's and Euler's Theorems

Theorem. (Fermat) If $b \in \mathbb{Z}_p^*$ and p is prime, then $b^{p-1} = 1 \bmod p$.

Theorem. (Euler) If $b \in \mathbb{Z}_n^*$, then $b^{\phi(n)} = 1 \bmod n$.

Recall. If $b \in G$, where G is an abelian group, then $b^{|G|} = 1$ due to Lagrange's theorem.

“Prime Number Theorem”

The primes are relatively dense!

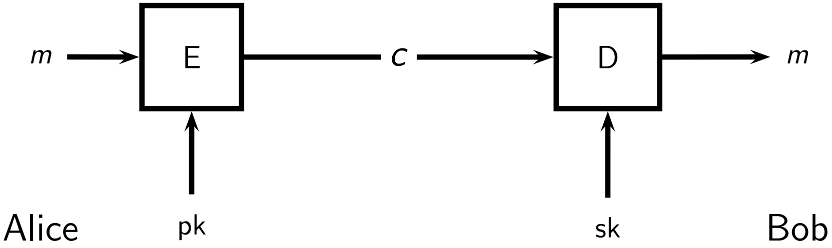
Theorem. A randomly chosen n-bit integer is prime with probability $\Theta(1/n)$.

Given a primality test we can sample random primes!

RSA Cryptosystem

Public-Key Cryptosystem

$c = \mathsf{E}_{\mathsf{pk}}(m)$ $m = \mathsf{D}_{\mathsf{sk}}(c)$



The RSA Cryptosystem (1/2)

Key Generation.

- Choose $n/2$-bit primes p and q randomly and define $N = pq$.
- Choose e in $\mathbb{Z}^*_{\phi(N)}$ and compute $d = e^{-1} \bmod \phi(N)$.

  This means that $ed = 1 \bmod \phi(N)$.
- Output the key pair $((N, e), (N, d))$, where (N, e) is the public key and (N, d) is the secret key.

The RSA Cryptosystem (2/2)

Encryption. Encrypt a plaintext $m \in \mathbb{Z}_N^*$ by computing

$$c = m^e \bmod N \ .$$

Decryption. Decrypt a ciphertext c by computing

$$m = c^d \bmod N \ .$$

Why Does It Work?

$$(m^e \bmod N)^d \bmod N = m^{ed} \bmod N$$

Why Does It Work?

$$\begin{aligned}(m^e \bmod N)^d \bmod N &= m^{ed} \bmod N \\ &= m^{1+t\phi(N)} \bmod N\end{aligned}$$

Why Does It Work?

$$\begin{aligned}(m^e \bmod N)^d \bmod N &= m^{ed} \bmod N \\ &= m^{1+t\phi(N)} \bmod N \\ &= m^1 \cdot \left(m^{\phi(N)}\right)^t \bmod N\end{aligned}$$

Why Does It Work?

$$\begin{aligned}
(m^e \bmod N)^d \bmod N &= m^{ed} \bmod N \\
&= m^{1+t\phi(N)} \bmod N \\
&= m^1 \cdot \left(m^{\phi(N)}\right)^t \bmod N \\
&= m \cdot 1^t \bmod N
\end{aligned}$$

Why Does It Work?

$$\begin{aligned}
(m^e \bmod N)^d \bmod N &= m^{ed} \bmod N \\
&= m^{1+t\phi(N)} \bmod N \\
&= m^1 \cdot \left(m^{\phi(N)}\right)^t \bmod N \\
&= m \cdot 1^t \bmod N \\
&= m \bmod N
\end{aligned}$$

Implementing RSA

- Source of randomness and primality test to find primes.
- Greatest common divisor to compute public key.
- Modular arithmetic for encryption and decryption.
- (Implementation of CRT for faster decryption.)

How Hard is Factoring?

The obvious way to break RSA is to factor the public modulus N and recover the prime factors p and q.

- The number field sieve factors N in time
$$O\left(e^{(1.92+o(1))\left((\ln N)^{1/3}+(\ln\ln N)^{2/3}\right)}\right) \quad .$$
- The elliptic curve method factors N in time
$$O\left(e^{(1+o(1))\sqrt{2\ln p\ln\ln p}}\right) \quad .$$

Note. The running times are subexponential:

$O(a^{n^c})$ with $c < 1$ for some $a > 1$.

Example of an Algebraic Weakness of RSA

Suppose that $e = 3$ is used by all parties as encryption exponent.

Small Message. If m is small, then $m^e < N$. Thus, **no reduction takes place**, and m can be computed in $\mathbb{Z}$ by taking the eth root.

Example of an Algebraic Weakness of RSA

Suppose that $e = 3$ is used by all parties as encryption exponent.

Small Message. If m is small, then $m^e < N$. Thus, **no reduction takes place**, and m can be computed in $\mathbb{Z}$ by taking the eth root.

There are less obvious weaknesses!

CPA Security

- RSA clearly provides some kind of “security”, but it is clear that we need to be careful.
- Intuitively, we want to leak no information of the encrypted plaintext.
- In other words, no function of the plaintext can efficiently be guessed notably better from its ciphertext than without it.

CPA Security

- RSA clearly provides some kind of “security”, but it is clear that we need to be careful.
- Intuitively, we want to leak no **knowledge** of the encrypted plaintext.
- In other words, no function of the plaintext can efficiently be guessed notably better from its ciphertext than without it.

CPA Security

- RSA clearly provides some kind of "security", but it is clear that we need to be careful.
- Intuitively, we want to leak no **knowledge** of the encrypted plaintext.
- In other words, no function of the plaintext can efficiently be guessed notably better from its ciphertext than without it.

Let us try to fix this and learn something about random oracles!

The RSA Assumption

Definition. The RSA assumption states that it is infeasible to recover the **complete** plaintext from a ciphertext provided that the plaintext is **randomly** chosen.

Wait a second!?

Are we assuming what we wish to prove is true?

Wait a second!?

Are we assuming what we wish to prove is true?

No!

We do assume something which we cannot prove, but

this assumption seems **weaker** than the security of all bits for every possible plaintext.

This type of reductionist proof of security is very common in cryptography and computer science.

Modified RSA with Random Oracle (1/2)

Suppose that $f : \{0,1\}^n \to \{0,1\}^n$ is a randomly chosen function (a random oracle).

- **Key Generation.** Choose a random RSA key pair $((N,e),(N,d))$, with $\log_2 N = n$.
- **Encryption.** Encrypt a plaintext $m \in \{0,1\}^n$ by choosing $r \in \mathbb{Z}_N^*$ **randomly** and computing

$$(u,v) = (r^e \bmod N, f(r) \oplus m) \ .$$

- **Decryption.** Decrypt a ciphertext (u,v) by

$$m = v \oplus f(u^d) \ .$$

Modified RSA with Random Oracle (2/2)

Security. In the random oracle model, recovering any bit of plaintext requires solving the RSA assumption.

Problems

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

Modified RSA with Random Oracle (2/2)

Security. In the random oracle model, recovering any bit of plaintext requires solving the RSA assumption.

Problems

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

Solutions.

- Using a "optimal" padding the first problem can be reduced. See standard OAEP+.

Modified RSA with Random Oracle (2/2)

Security. In the random oracle model, recovering any bit of plaintext requires solving the RSA assumption.

Problems

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

Solutions.

- Using a “optimal” padding the first problem can be reduced. See standard OAEP+.
- Using a scheme with an impractical rate, the second problem can be removed.

Signature Schemes

Digital Signature

- A digital signature is the **public-key** equivalent of a MAC; the receiver verifies the integrity and authenticity of a message.
- Does a digital signature replace a real handwritten one?
- How do you verify a written signature?

Textbook RSA Signature (1/2)

- Generate RSA keys $((N, e), (N, d))$.
- To sign a message $m \in \mathbb{Z}_N$, compute $\sigma = m^d \bmod N$.
- To verify a signature σ of a message m, verify that $\sigma^e = m \bmod N$.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2 \bmod N$ is a signature of $m_1 m_2 \bmod N$

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2 \bmod N$ is a signature of $m_1 m_2 \bmod N$
- We can also pick a signature σ and compute the message it is a signature of by $m = \sigma^e \bmod N$.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2 \bmod N$ is a signature of $m_1 m_2 \bmod N$
- We can also pick a signature σ and compute the message it is a signature of by $m = \sigma^e \bmod N$.

We must be more careful!

RSA with Full Domain Hash (1/2)

Let $H_N : \{0,1\}^* \to \mathbb{Z}_N^*$ be a random oracle for every N.

- Generate RSA keys $((N, e), (N, d))$.
- To sign a message $m \in \{0,1\}^*$, compute $\sigma = H_N(m)^d \bmod N$.
- To verify a signature σ of a message m, verify that $\sigma^e = H_N(m) \bmod N$.

RSA with Full Domain Hash (2/2)

Theorem. RSA-FDH is provably secure in the random oracle model under the RSA assumption.

Problems.

- The hash function H_N has an inconvenient range.
- Our analysis is in the random oracle model, **which is unsound!**.

RSA with Full Domain Hash (2/2)

Theorem. RSA-FDH is provably secure in the random oracle model under the RSA assumption.

Problems.

- The hash function H_N has an inconvenient range.
- Our analysis is in the random oracle model, **which is unsound!**.

Solutions.

- The hash function can be replaced by a standard one or theoretical results can avoid it.

RSA with Full Domain Hash (2/2)

Theorem. RSA-FDH is provably secure in the random oracle model under the RSA assumption.

Problems.

- The hash function H_N has an inconvenient range.
- Our analysis is in the random oracle model, **which is unsound!**.

Solutions.

- The hash function can be replaced by a standard one or theoretical results can avoid it.
- Using the strong RSA assumption a signature scheme without random oracles or other additional assumptions can be constructed.

PKIs

Problem

- We have constructed public-key cryptosystems and signature schemes.
- Only the holder of the secret key can decrypt ciphertexts and sign messages.
- How do we **know** who holds the secret key corresponding to a public key?

Signing Public Keys of Others

- Suppose that Alice computes a signature $\sigma_{A,B} = \mathsf{Sig}_{\mathsf{sk}_A}(\mathsf{pk}_B, Bob)$ of Bob's public key pk_B and his identity and hands it to Bob.
- Suppose that Eve holds Alice's public key pk_A.
- Then **anybody** can hand $(\mathsf{pk}_B, \sigma_{A,B})$ **directly** to Eve, and Eve will be convinced that pk_B is Bob's key (assuming she trusts Alice).

Certificate

- A **certificate** is a signature of a public key along with some information on how the key may be used, e.g., it may allow the holder to issue certificates.
- A certificate is valid for a given setting if the signature is valid and the usage information in the certificate matches that of the setting.
- Some parties must be trusted to issue certificates. These parties are called Certificate Authorities (CA).

Certificate Chains

A CA may be “distributed” using in certificate chains.

- Suppose that Bob holds valid certificates

$$\sigma_{0,1}, \sigma_{1,2}, \ldots, \sigma_{n-1,n}$$

  where $\sigma_{i-1,i}$ is a certificate of pk_{P_i} by P_{i-1}.

- Who does Bob trust?

DD2520 Applied Cryptography
Lecture 5

Douglas Wikström
KTH Royal Institute of Technology
`dog@kth.se`

Jan 27, 2024

Zero-Knowledge Proofs

Interaction

- A student claims to know the content of a course.

  Why does passing the written exam lead to passing the course?

- A new friend claims that they are a snowboarder like you.

  How can they convince you outside the slope?

- An attentative suspect claims watching the news as an alibi.

  How can the suspect convince the Police quickly?

- You use a VPN and Amazon is unhappy about that. They require you to unlock your account.

  How do they authenticate you?

Real-world Properties

- **Completeness.** Honest parties mostly agree on any claims after interacting.
- **Knowledge extraction.** Claims of knowledge are accepted only if the prover actually do know most of what is claimed.
- **Soundness.** Truth claims are accepted only if they are likely to be true.
- **Zero knowledge.** Interactions do not leak too much of the evidence.

What is a claim?

Consider NP relations of the form $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^*$ and corresponding languages

$$\mathcal{L} = \{x \mid (x,w) \in \mathcal{R}\} \ .$$

Let $x \in \{0,1\}^*$ be a string and $\mathcal{R}$ a relation.

Knowledge claim about x:

- I know w such that $(x,w) \in \mathcal{R}$.

Truth claim about x:

- $x \in \mathcal{L}$, or equivalently
- there exists a w such that $(x,w) \in \mathcal{R}$.

Examples

Example. RSA moduli and their factorizations

$$\mathcal{R}_{RSA} = \big\{\big(N,(p,q)\big) \mid p \text{ and } q \text{ are distinct primes and } N = pq\big\}$$

- We can efficiently determine that N is a composite integer!
- Proving knowledge of (p, q) such that $N = pq$ still make sense.

Examples

Example. Let G by a cyclic group of order q with generator g

$$\mathcal{R}_{DL} = \left\{(y, x) \mid x \in \mathcal{Z}_q \text{ and } y = g^x\right\}$$

- For every $y \in G$ there exists an $x \in \mathcal{Z}_q$ such that $y = g^x$.
- Proving knowledge of x such that $y = g^x$ still make sense.

Examples

Example. Let G by a cyclic group of order q with generators g and h

$$\mathcal{R}_{DL} = \big\{\big((y,z),x\big) \mid x \in \mathcal{Z}_{\boldsymbol{q}} \text{ and } (y,z) = (g^x, h^x)\big\}$$

- Most pairs (y,z) do not have this property and it is infeasible to check!
- Proving that there exists an x such that $(y,z) = (g^x, h^x)$ makes sense.
- Proving knowledge of x such that $(y,z) = (g^x, h^x)$ makes sense.

Properties

- **Knowledge extraction.** If Alice convinces Bob, then we could extract w from Alice: "Alice knows w".
- **Soundness.** If $x \notin \mathcal{L}$, then Alice convinces Bob with small probability no matter what she does: "Alice cannot lie".
- **Completeness.** If Alice and Bob are honest, then Bob accepts Alice's claim: "it works".
- **Zero knowledge.** Bob can simulate an interaction with Alice on his own: "Bob does not learn anything".

Example: Written Exam

Example: Written Exam

- Some **knowledge extraction.** If Alice passes the course, then we could rewind time and ask different questions until we have extracted everything that Alice knows.

Example: Written Exam

- Some **knowledge extraction.** If Alice passes the course, then we could rewind time and ask different questions until we have extracted everything that Alice knows.
- Decent **soundness.** If Alice committed to everything she knows in a bunch of envelopes, she could prove that the content of her commitment corresponded to a given grade by opening a few. (Proof claim.)

Example: Written Exam

- Some **knowledge extraction.** If Alice passes the course, then we could rewind time and ask different questions until we have extracted everything that Alice knows.

- Decent **soundness.** If Alice committed to everything she knows in a bunch of envelopes, she could prove that the content of her commitment corresponded to a given grade by opening a few. (Proof claim.)

- Resonable **completeness.** If Alice knows the course content and the teacher is honest, then she will most likely pass the course.

Example: Written Exam

- Some **knowledge extraction.** If Alice passes the course, then we could rewind time and ask different questions until we have extracted everything that Alice knows.
- Decent **soundness.** If Alice committed to everything she knows in a bunch of envelopes, she could prove that the content of her commitment corresponded to a given grade by opening a few. (Proof claim.)
- Resonable **completeness.** If Alice knows the course content and the teacher is honest, then she will most likely pass the course.
- Almost **zero knowledge.** Bob could look up some random answers and prepare an exam for the corresponding questions.

Example: Written Exam

The teacher must be experienced to write a good exam, but they need to know very little of the content of the course to assess knowledge precisely.

Teaching is much harder than assessing!

Example: Written Exam

The teacher must be experienced to write a good exam, but they need to know very little of the content of the course to assess knowledge precisely.

Teaching is much harder than assessing!

Computing is much harder than verifying!

Prove knowledge of pre-image

Let q be a prime and let G be a group of order q. Let $\phi : \mathbb{Z}_q \to G$ be a homomorphism, i.e., $\phi(a)\phi(b) = \phi(ab)$ for any $a, b \in \mathbb{Z}_q$.

Alice holds a such that $A = \phi(a)$ and Bob holds A.

1. Alice chooses $r \in \mathbb{Z}_q$ randomly and hands $R = \phi(r)$ to Bob.
2. Bob chooses $c \in \mathbb{Z}_q$ randomly and hands c to Alice.
3. Alice computes $d = ca + r \bmod q$ and hands d to Bob.
4. Bob accepts if and only if $A^c R = \phi(d)$.

Completeness

If Alice and Bob are honest, then Bob is convinced since

$$\phi(d) = \phi(ca + r) = \phi(a)^c \phi(r) = A^c R \ .$$

Knowledge Extraction

Given **two related accepting** executions (R, c, d) and (R, c', d') such that

$$\begin{gathered} c \neq c' \\ A^c R = \phi(d) \\ A^{c'} R = \phi(d') \end{gathered}$$

Knowledge Extraction

Given **two related accepting** executions (R, c, d) and (R, c', d') such that

$$c \neq c'$$
$$A^c R = \phi(d)$$
$$A^{c'} R = \phi(d')$$

we have

$$\frac{A^c R}{A^{c'} R} = \frac{\phi(d)}{\phi(d')}$$

Knowledge Extraction

Given **two related accepting** executions (R, c, d) and (R, c', d') such that

$$c \neq c'$$
$$A^c R = \phi(d)$$
$$A^{c'} R = \phi(d')$$

we have

$$A^{c-c'} = \frac{A^c R}{A^{c'} R} = \frac{\phi(d)}{\phi(d')} = \phi(d - d')$$

Knowledge Extraction

Given **two related accepting** executions (R, c, d) and (R, c', d') such that

$$c \neq c'$$
$$A^c R = \phi(d)$$
$$A^{c'} R = \phi(d')$$

we have

$$A^{c-c'} = \frac{A^c R}{A^{c'} R} = \frac{\phi(d)}{\phi(d')} = \phi(d - d')$$

from which we can derive a preimage

$$x = (d - d')(c - c')^{-1}$$

such that $A = \phi(x)$.

Zero Knowledge

Bob can choose $c, d \in \mathbb{Z}_q$ randomly and compute

$$R = \phi(d)A^{-c}$$

to form a transcript (R, c, d) such that $A^c R = \phi(d)$.

This type of protocol is very common in practice. It is an **honest** verifier zero knowledge protocol, more precisely a Sigma protocol, or even more precisely a Schnorr protocol.

Fiat-Shamir Transform

Interaction is incredibly powerful, but also impractical.

Note that the verifier only flips coins in the protocol.

How do we reduce the protocol to a single message from Alice to Bob, which Bob verifies?

If we manage to do this, then anybody can post a their (only) message online and prove whatever they like!!!

Fiat-Shamir Transform

$H : \{0,1\}^* \to \mathbb{Z}_q$ is tweaked-SHA-3 and $m \in \{0,1\}^*$.

1. Alice picks $r \in \mathbb{Z}_q$ and computes $R = \phi(r)$ ~~and hands R to Bob~~.
2. Alice computes $c = H(A, R, m)$.
3. Alice computes $d = ca + r \bmod q$ and hands (R, d) to Bob.
4. Bob accepts if and only if $A^{H(A,R,m)} R = \phi(d)$.

Fiat-Shamir Transform

$H : \{0,1\}^* \to \mathbb{Z}_q$ is tweaked-SHA-3 and $m \in \{0,1\}^*$.

1. Alice picks $r \in \mathbb{Z}_q$ and computes $R = \phi(r)$ ~~and hands R to Bob~~.

2. Alice computes $c = H(A, R, m)$.

3. Alice computes $d = ca + r \bmod q$ and hands (R, d) to Bob.

4. Bob accepts if and only if $A^{H(A,R,m)} R = \phi(d)$.

Non-interactive!

Alice has signed m **using the public key** A **:-)**

Real-world Multiparty Computation

- Multiple keys held by different people needed to open a safe.
- Multiple persons partition a set $x_1, \ldots, x_N$, sum the integers in their part, and add the results.
- Counting votes in an election.

Average Height

Your height is h_i. We wish to compute $\frac{1}{N}\sum_{i\in[N]} h_i$.

Choose a random integer $r_i \in [0, 1000]$ and compute $a_i = h_i + r_i$.

Set $s_0 = 0$.

For $i = 1, \ldots, N$: compute $s_i = s_{i-1} + a_i$.

Set $t_0 = s_N$.

For $i = 1, \ldots, N$: compute $t_i = t_{i-1} - r_i$.

Average Height

Your height is h_i. We wish to compute $\frac{1}{N}\sum_{i\in[N]} h_i$.

Choose a random integer $r_i \in [0, 1000]$ and compute $a_i = h_i + r_i$.

Set $s_0 = 0$.

For $i = 1, \dots, N$: compute $s_i = s_{i-1} + a_i$.

Set $t_0 = s_N$.

For $i = 1, \dots, N$: compute $t_i = t_{i-1} - r_i$.

Who learns what?

DD2520 Applied Cryptography
Lecture 7

Douglas Wikström
KTH Royal Institute of Technology
`dog@kth.se`

February 3, 2024

Paper-based Voting Systems

- People identify themselves.
- Mark their ballots.
- Votes are counted.

...but, **how** exactly?

Good and bad arguments pro and con electronic voting?