Katz, Lindell
Introduction to Modern Cryptrography
Slides Chapter 5

Markus Bläser, Saarland University

Hash functions

- Hash functions map elements from a large set into a small set.
- Good hash functions produce few collissions.

Definition (5.1)

A *hash functions* is a pair of ppt algorithms $(\mathrm{Gen}, \mathrm{H})$ such that

1. Gen on input 1^n outputs some key s.
2. H on input s and $x \in \{0,1\}^*$ outputs a value $\mathrm{H}^s(x) \in \{0,1\}^{\ell(n)}$.

If H is only defined on inputs $x \in \{0,1\}^{\ell'(n)}$, then H is a *fixed length hash function* for inputs of length $\ell'(n) > \ell(n)$.

Collision resistance

The collision finding experiment $\text{Hash-Coll}_{\mathcal{A},\Pi}(n)$:

1. A key s is generated using $\text{Gen}(1^n)$.
2. $\mathcal{A}$ gets s and outputs two output x, x'.
   (x, x' have length $\ell'(n)$ if H is fixed-length.)
3. The output of the experiment is 1 if $x \neq x'$ but $H^s(x) = H^s(x')$. Otherwise, it is 0

Definition (5.2)

$\Pi = (\text{Gen}, H)$ is *collision resistant* if for all ppt adversaries $\mathcal{A}$,

$$\Pr[\text{Hash-Coll}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Merkle-Damgård transform

Construction (5.3)

(Gen, h) *is a fixed-length hash function for inputs of length* $2n$ *and with output length* n*. Construct a hash function* (Gen, H) *as follows:*

- H *on key* s *and input* $x \in \{0,1\}^*$ *with* $|x| =: L < 2^n$ *does the following:*
  1. *Set* $B := \lceil L/n \rceil$*. Break* x *input blocks* $x_1, \ldots, x_B$ *of length* n *each (padding the last block with 0s). Set* $x_{B+1} = L$ *(encoded in binary).*
  2. *Set* $z_0 := 0^n$ *("initialisation vector")*
  3. *For* $i = 1, \ldots, B+1$ *compute* $z_i := h^s(z_{i-1} \| x_i)$*.*
  4. *Output* z_{B+1}*.*

Theorem (5.4)

If (Gen, h) *is collision resistant, so is* (Gen, H)*.*

Hash-and-MAC

Construction (5.5)

Let $\Pi = (\mathsf{Mac}, \mathsf{Vrfy})$ *be a MAC for length* $\ell(n)$.
Let $\Pi_H = (\mathsf{Gen}_H, H)$ *be a hash function with output length* $\ell(n)$.
Construct $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ *as follows:*

- Gen': *Choose* $k \in \{0,1\}^n$ *uniformly at random and let* $s \leftarrow \mathsf{Gen}_H(1^n)$. *Return* $\langle k, s \rangle$.
- Mac': *given message* m, *output* $\mathsf{Mac}_k(H^s(m))$.
- Vrfy': *given message* m *and tag* t, *output* 1 *if* $\mathsf{Vrfy}_k(H^s(m), t) = 1$. *Otherwise, output* 0.

Theorem (5.6)

If Π *is a secure MAC for messages of length* ℓ *and* Π_H *is collision resistant, then Construction 5.5 is a secure MAC for arbitary length messages.*

Hash-and-MAC (2)

Collision-detector $\mathcal{C}$:

Input: key s

1. Choose k uniformly at random from $\{0,1\}^n$.
2. Run $\mathcal{A}'(1^n)$. When $\mathcal{A}'$ requests a tag for m_i, then compute $t_i \leftarrow \mathrm{Mac}_k(H^s(m_i))$ and give it to $\mathcal{A}$'.
3. When $\mathcal{A}'$ outputs (m^*, t^*), then if there is an i for which $H^s(m^*) = H^s(m_i)$, output (m^*, m_i).

Hash-and-Mac (3)

Adversary $\mathcal{A}$:

Input: oracle access to $\mathrm{Mac}_k(.)$

1. $s \leftarrow \mathrm{Gen}_H(1^n)$
2. Run $\mathcal{A}'(1^n)$. When $\mathcal{A}'$ requests a tag on the ith message m_i, then compute $\hat{m}_i := H^s(m_i)$ and obtain a tag $t_i \leftarrow \mathrm{Mac}_k(\hat{m}_i)$. Give t_i to $\mathcal{A}'$.
3. When $\mathcal{A}'$ outputs (m^*, t^*), then output $(H^s(m^*), t^*)$.

Birthday attacks

Algorithm (5.9)

Input: *Hash function* $H : \{0,1\}^* \to \{0,1\}^\ell$.
Output: *Distinct* x, x' *with* $H(x) = H'(x)$.

1. $x_0 \leftarrow \{0,1\}^{\ell+1}$; $x' := x := x_0$;
2. *for* $i := 1, 2, \ldots$ *do*
3. $\quad x := H(x)$; $x' := H(H(x'))$
4. $\quad$ *if* $x = x'$ *then break*
5. $x' := x$; $x := x_0$;
6. *for* $j := 1, \ldots, i$ *do*
7. $\quad$ *if* $H(x) = H(x')$ *then return* x, x'
8. $\quad$ *else* $x := H(x)$; $x' := H(x')$

Random oracle model

Some constructions using hash functions can only be proven secure in a stronger model...

$\longrightarrow$ *random oracle model*

- blackbox containing a truly random function H
- H can only be evaluated, we cannot inspect the "code" or anything like this

Two step approach:

1. Security proof in the random oracle model
2. In reality, instantiate random oracle by appropriate some hash function.

!!!! Security proof becomes invalid, since

- we e.g. now have the code of the function.
- H now is deterministic

The random oracle model (2)

- **Standard model:** Probability taken over internal randomness of Π and the internal randomness of adversary $\mathcal{A}$.
- **Random oracle model:** Probability taken in addition over H. H is part of the construction!!!

Security is not guaranteed for any particular choice of H!

Random oracle model (3)

Important properties:

- If x has not been queried so far, then $\mathrm{H}(\mathrm{x})$ is uniform. (If H is queried a second time, then the answer has to be consistent. Queries are however "private".)

In a proof by reduction, the reduction has to simulate the oracle H for $\mathcal{A}$.

- Extractability: If $\mathcal{A}$ queries H, the reduction can learn x.
- Programmability: The reduction can set the value $\mathrm{H}(\mathrm{x})$ as long as it is correctly (uniformly) distributed.

Prgs from a random oracle

$H : \{0,1\}^{\ell_{in}(n)} \to \{0,1\}^{\ell_{out}(n)}$, $\ell_{in}(n), \ell_{out}(n) > n$,
n security parameter.

If $\ell_{out}(n) > \ell_{in}(n)$, then H can be used as a prg:

$$\left|\Pr[\mathcal{D}^{H(.)}(y) = 1] - \Pr[\mathcal{D}^{H(.)}(H(x)) = 1]\right| \leq \mathsf{negl}(n).$$

left-hand $\Pr$: over randomness of $\mathcal{D}$, H, and y
right-hand $\Pr$: over randomness of $\mathcal{D}$, H, and x

- $H(x)$ does not look random to $\mathcal{D}$ iff he queried $H(x)$.
- x is chosen at random and $\mathcal{D}$ can query only a polynomial number of values $\longrightarrow$ negligible.

Hash functions from a random oracle

If $\ell_{\mathrm{out}}(n) < \ell_{\mathrm{in}}(n)$, then H is a collision resistant Hash function.

- $\mathcal{A}$ can query H at a polynomial number of points
- he can only win if he finds a collision
- H is random $\longrightarrow$ birthday paradox

Prfs from a random oracle

$\ell_{in}(n) = 2n$, $\ell_{out}(n) = n$

We define a prf as follows:

$$F_k(x) := H(k||x)$$

For every ppt distinguisher $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}^{H(.),H(k||.)}(1^n) = 1] - \Pr[\mathcal{D}^{H(.),f(.)}(1^n) = 1]\right| = \mathsf{negl}(n),$$

since unless you query $H(.)$ at a point of the form $k||z$, the two views are identical. The latter happens with probability $q/2^n$.

Further uses of Hash functions

- Virus fingerprinting
- Deduplication in multi-user file systems
- Authentication of stored data $\longrightarrow$ Merkle trees

Collision resistance is sufficient

Password hashing

- Instead of storing a password w, we store $\hbar w := \mathrm{H}(w)$.
- When a password w' is entered, we check $\hbar w = \mathrm{H}(w')$.
- w comes usually from a relatively small subset $\mathrm{D} \subset \{0,1\}^*$.

We would like to have that finding a word w' with $\mathrm{H}(w') = \hbar w$ is proportional to $|\mathrm{D}|$. (Brute force dictionary attack)

Only known in the random oracle model

Key derivation

In practice, two parties share a random password that is not uniformly distributed. They want to generate a uniform key from this.

Definition (5.12)

A probability distribution D has *min-entropy* m if for every fixed x: $\Pr[X = x] \leq 2^{-m}$ if X is distributed according to D.

Key derivation: Obtain a uniformly distributed string from a distribution with high min-entropy.

A random oracle is a good key derivation function

- Finding $H(x)$ with q queries has probability $q2^{-m}$.

Can be done without the random oracle by a *randomness extractor*.

Commitment schemes

- Gen: On input 1^n, generates appropriate parameters
- Com: gets the parameters p and a message $m \in \{0,1\}^n$, chooses some random string r and outputs a string $c := \mathrm{Com}(p, m, r)$.

Idea: c commits a sender to m without revealing it. "digital envelope"

Commitment hiding experiment

$\mathsf{Hiding}_{\mathcal{A},\mathsf{Com}}(n)$:

1. Parameters p are generated using Gen
2. $\mathcal{A}$ is given the parameters and it outputs two messages m_0, m_1.
3. b is chosen uniformly at random and $c \leftarrow \mathsf{Com}(p, m_b, r)$ is computed
4. c is given to $\mathcal{A}$ and he outputs a bit b'
5. The outcome of the experiment is 1 if $b = b'$ and 0 otherwise.

Commitment binding experiment

$\mathsf{Binding}_{\mathcal{A},\mathsf{Com}}(n)$

1. Parameters p are generated using Gen
2. $\mathcal{A}$ is given the parameters and it outputs (c, m, r, m', r')
3. The outcome of the experiment is 1, if $m \neq m'$ and $\mathsf{Com}(p, m, r) = c = \mathsf{Com}(p, m', r')$

Secure commitment schemes

Definition (5.13)

A commitment scheme Com is secure if for all ppt $\mathcal{A}$,

$$\Pr[\mathsf{Hiding}_{\mathcal{A},\mathsf{Com}}(n) = 1] \leq 1/2 + \mathsf{negl}(n),$$
$$\Pr[\mathsf{Binding}_{\mathcal{A},\mathsf{Com}}(n) = 1] \leq \mathsf{negl}(n).$$

- Commitment $c := H(m||r)$ for a uniformly chosen $r \in \{0,1\}^n$
- Hiding follows from the fact that $\mathcal{A}$ queries $H(*||r)$ only with negligible probability.
- Binding follows from collision resistance