One-way functions

Daniel Bosk

School of Computer Science and Communication,
KTH Royal Institute of Technology, Stockholm

Department of Information and Communication Systems,
Mid Sweden University, Sundsvall

6th April 2020

1 One-way functions
- Hash functions
- Message-authentication codes

Idea

- We want a function which we can efficiently compute.
- However, it shouldn't be possible to find its inverse.

Example

Easy $f(x) = y$

Hard $f^{-1}(y) = x$

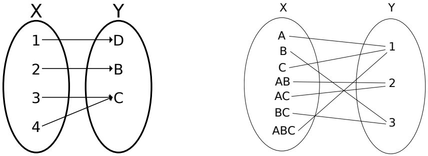
Idea

- We want a function which we can efficiently compute.
- However, it shouldn't be possible to find its inverse.

Example

Easy $f(x) = y$

Hard $f^{-1}(y) = x$



(a) $h \colon X \to Y$ (b) $h' \colon X \to Y$

Figure: Two non-injective, surjective functions h and h'.

Exercise

Could either of these two functions be one-way functions?

Definition (One-way function[1])

- Let $h\colon \{0,1\}^* \to \{0,1\}^*$.
- h is *one-way* if
  1. there exists an efficient algorithm A such that $A(x) = h(x)$;
  2. for every efficient algorithm A', every positive polynomial $p(\cdot)$ and all sufficiently large n's

$$\Pr\left[A'(h(x), 1^n) \in h^{-1}(h(x))\right] < \frac{1}{p(n)}$$

[1] **GoldreichFOC-1**.

Definition (Preimage resistance)

Input hash function H, value y.

Output Any x such that $H(x) = y$.

Definition (Second preimage resistance)

Input hash function H, value x.

Output Any value x' such that $H(x) = H(x')$.

Definition (Collision resistance)

Input hash function H.

Output Any two x, x' such that $H(x) = H(x')$.

Example (Implementations you might've heard of)

- MD5
- SHA1
- SHA256 (SHA-2)
- SHA-3

Example (Applications)

- Verifying file content integrity
- Digital signatures
- Protect passwords

Note

- One-wayness returns as a useful property in many situations.
- Encryption also has the one-wayness property:
  - Easy: Given k, m, compute $c \leftarrow \text{Enc}_k(m)$.
  - Hard: Given c, compute either of k, m.
- However, encryption is bijective, hash functions are generally not.

Example

- Let $\text{Enc}_k(\cdot) = \text{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\text{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\text{Dec}_k(c') = \text{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Example

- Let $\text{Enc}_k(\cdot) = \text{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\text{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\text{Dec}_k(c') = \text{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Example

- Let $\mathrm{Enc}_k(\cdot) = \mathrm{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\mathrm{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\mathrm{Dec}_k(c') = \mathrm{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Example

- Let $\mathrm{Enc}_k(\cdot) = \mathrm{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\mathrm{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\mathrm{Dec}_k(c') = \mathrm{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Example

- Let $\text{Enc}_k(\cdot) = \text{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\text{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\text{Dec}_k(c') = \text{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Example

- Let $\text{Enc}_k(\cdot) = \text{Dec}_k(\cdot) = \cdot \oplus k \bmod 2$.
- Alice and Bob share k.
- Alice sends $\text{Enc}_k(m) = c$ to Bob.
- Eve intercepts c, she cannot get to m.
- Eve computes $c' = c \oplus m_E$ and passes c' to Bob.
- Bob computes $\text{Dec}_k(c') = \text{Dec}_k(c \oplus m_E) = m \oplus k \oplus m_E \oplus k = m \oplus m_E$.

Exercise

How can we solve this? Bob needs to know that Eve modified the message!

Idea: MACs

- Alice and Bob need something that Eve doesn't know how to modify.
- If that something is tied to the message, then a modified message would be detectable.

Exercise

Any ideas on how we can construct such a thing?

Idea: MACs

- Alice and Bob need something that Eve doesn't know how to modify.
- If that something is tied to the message, then a modified message would be detectable.

Exercise

Any ideas on how we can construct such a thing?

Idea: MACs

- Alice and Bob need something that Eve doesn't know how to modify.
- If that something is tied to the message, then a modified message would be detectable.

Exercise

Any ideas on how we can construct such a thing?

Example

- Let h be a one-way function.
- If we use $h(c) = t$, then Eve can also compute the hash function: $h(c') = t'$.
- A secret hash function would violate Kerckhoff's principle, so that's not an option.
- If we instead use the message, rather than the ciphertext.
- Then $h(m) = t$ and
  - $\text{Dec}(k)c' = m' = m \oplus m_E, h(m') \neq t$.
  - $\text{Dec}(k)c = m, h(m) = t$.
- Eve makes up m', she can compute $t' = h(m')$.

Example

- Let h be a one-way function.
- If we use $h(c) = t$, then Eve can also compute the hash function: $h(c') = t'$.
- A secret hash function would violate Kerckhoff's principle, so that's not an option.
- If we instead use the message, rather than the ciphertext.
- Then $h(m) = t$ and
  - $\mathrm{Dec}(k)c' = m' = m \oplus m_E, h(m') \neq t$.
  - $\mathrm{Dec}(k)c = m, h(m) = t$.
- Eve makes up m', she can compute $t' = h(m')$.

Example

- Let h be a one-way function.
- If we use $h(c) = t$, then Eve can also compute the hash function: $h(c') = t'$.
- A secret hash function would violate Kerckhoff's principle, so that's not an option.
- If we instead use the message, rather than the ciphertext.
- Then $h(m) = t$ and
  - $\mathrm{Dec}(k)c' = m' = m \oplus m_E, h(m') \neq t$.
  - $\mathrm{Dec}(k)c = m, h(m) = t$.
- Eve makes up m', she can compute $t' = h(m')$.

Example

- Let h be a one-way function.
- If we use $h(c) = t$, then Eve can also compute the hash function: $h(c') = t'$.
- A secret hash function would violate Kerckhoff's principle, so that's not an option.
- If we instead use the message, rather than the ciphertext.
- Then $h(m) = t$ and
  - $\mathrm{Dec}(k)c' = m' = m \oplus m_E, h(m') \neq t$.
  - $\mathrm{Dec}(k)c = m, h(m) = t$.
- Eve makes up m', she can compute $t' = h(m')$.

Example

- Let h be a one-way function.
- If we use $h(c) = t$, then Eve can also compute the hash function: $h(c') = t'$.
- A secret hash function would violate Kerckhoff's principle, so that's not an option.
- If we instead use the message, rather than the ciphertext.
- Then $h(m) = t$ and
  - $\mathrm{Dec}(k)c' = m' = m \oplus m_E, h(m') \neq t$.
  - $\mathrm{Dec}(k)c = m, h(m) = t$.
- Eve makes up m', she can compute $t' = h(m')$.

Solution

- *Let s be a secret shared between Alice and Bob.*
- $h(c \parallel s) = t$, *Eve doesn't know* s.
- *Bob can immediately check* $h(c' \parallel s) \neq t$.

Note

- It requires even a bit more than this!
- But the idea is correct.

Solution

- *Let s be a secret shared between Alice and Bob.*
- $h(c \parallel s) = t$, *Eve doesn't know* s.
- *Bob can immediately check* $h(c' \parallel s) \neq t$.

Note

- It requires even a bit more than this!
- But the idea is correct.

Solution

- *Let s be a secret shared between Alice and Bob.*
- $h(c \parallel s) = t$, *Eve doesn't know* s.
- *Bob can immediately check* $h(c' \parallel s) \neq t$.

Note

- It requires even a bit more than this!
- But the idea is correct.

Solution (Hash-based message-authentication code, HMAC[2])

- *Let h be a one-way function.*
- *Let c be the ciphertext, s our MA secret.*
- *Then tag $t = \mathrm{HMAC}_s(c)$, where*

$$\mathrm{HMAC}_s(c) = h\left[(s \oplus p_o) \parallel h\left[(s \oplus p_i) \parallel c\right]\right],$$

and p_i, p_o are inner and outer pads, respectively.

Note

This is proven secure by **HMAC**!

[2] **HMAC**.

Solution (Hash-based message-authentication code, HMAC[2])

- *Let h be a one-way function.*
- *Let c be the ciphertext, s our MA secret.*
- *Then tag $t = \mathrm{HMAC}_s(c)$, where*

$$\mathrm{HMAC}_s(c) = h\left[(s \oplus p_o) \parallel h\left[(s \oplus p_i) \parallel c\right]\right],$$

and p_i, p_o are inner and outer pads, respectively.

Note

This is proven secure by **HMAC**!

[2] **HMAC**.

Solution (Hash-based message-authentication code, HMAC[2])

- *Let h be a one-way function.*
- *Let c be the ciphertext, s our MA secret.*
- *Then tag $t = \mathrm{HMAC}_s(c)$, where*

$$\mathrm{HMAC}_s(c) = h\left[(s \oplus p_o) \parallel h\left[(s \oplus p_i) \parallel c\right]\right],$$

and p_i, p_o are inner and outer pads, respectively.

Note

This is proven secure by **HMAC**!

[2] **HMAC**.

Message-authentication codes