Cryptocomplexity II

Kryptokomplexität II

Sommersemester 2024

Chapter 6: Other Protocols

Dozent: Prof. Dr. J. Rothe

Merkle and Hellman's Public-Key Cryptosystem

- Merkle and Hellman proposed their public-key cryptosystem in 1978.
- Although it was broken by Shamir in the early 1980s, it is still worth studying it, since it is simple and elegant and particularly suitable for illustrating the basic design principles of public-key cryptography.
- However, there are variants of this cryptosystem that are still unbroken to this date.
- All public-key cryptosystems considered so far are based on the idea of *trapdoor one-way functions*.

Merkle and Hellman's Public-Key Cryptosystem

- For example, the RSA public-key cryptosystem employs the fact that
  - modular exponentiation can be performed efficiently using "square-and-multiply" (so both encryption and authorized decryption are easy with the prime factors of n as trapdoor information), but
  - unauthorized decryption seems to be hard, since computing m from e, n, and $c = m^e \bmod n$ (i.e., extracting the e^{th} root of c modulo n) and factoring the RSA modul n both are considered to be computationally infeasible tasks.

- Similarly, in ElGamal's public-key cryptosystem
  - both encryption and authorized decryption are easy, where the latter uses Bob's private key b as trapdoor information, but
  - unauthorized decryption appears to be hard, since computing discrete logarithms is considered infeasible.

Merkle and Hellman's Public-Key Cryptosystem

- Both computing discrete logarithms and extracting roots modulo an integer can be viewed as inverses of the modular exponentiation function.

- The difference between these two inverse functions is that
  - root extraction for

    $$\alpha = \beta^a \bmod n$$

    means computing the base β from α, a, and n, whereas
  - the discrete logarithm of α requires computing the exponent a given α, β, and n.

Merkle and Hellman's Public-Key Cryptosystem

- The Merkle–Hellman cryptosystem is also based on the idea of trapdoor one-way functions. In particular, its security rests on the hardness of the *subset-of-sums problem*, denoted SOS.
- SOS is a restricted variant of the *knapsack problem*, and we will show that SOS is NP-complete.
- Thus, there is no known deterministic or randomized polynomial-time algorithm solving this problem.
- The trapdoor information used here is that certain instances of SOS are nonetheless easy to solve.

Subset-of-Sums Problem

Definition

The *subset-of-sums problem*, SOS, is defined as follows: Given a sequence $s_1, s_2, \ldots, s_n, T$ of positive integers (encoded in binary), does there exist a boolean vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ in $\{0,1\}^n$ such that

$$\sum_{i=1}^{n} x_i s_i = T\,?$$

The numbers s_i are called the *sizes*, and T is the *target sum*. Formalized as a set of yes-instances, this decision problem has the following form:

$$\mathrm{SOS} = \left\{ \langle s_1, s_2, \ldots, s_n, T \rangle \;\middle|\; \begin{array}{l} s_1, s_2, \ldots, s_n, T \in \mathbb{N} - \{0\}, \text{ and there is} \\ \text{some } \vec{x} \in \{0,1\}^n \text{ such that } \sum_{i=1}^{n} x_i s_i = T \end{array} \right\}.$$

Subset-of-Sums Problem

Theorem

SOS *is* NP*-complete.*

Proof:

- The proof that SOS belongs to NP is easy and omitted.
- To prove NP-hardness, we reduce from the NP-complete problem X-3-Cover. We are given:
  - a set U of cardinality $3m$ for some $m \in \mathbb{N}$ and
  - a collection $\mathscr{S} \subseteq \mathfrak{P}(U)$ of subsets of U such that every set in $\mathscr{S}$ has exactly three elements.

  Question: Do there exist m pairwise disjoint sets $S_1, S_2, \ldots, S_m \in \mathscr{S}$ such that $U = \bigcup_{i=1}^{m} S_i$?

Subset-of-Sums Problem

- Our goal is to construct from X-3-Cover instance $\langle U, \mathscr{S} \rangle$ an instance $\langle s_1, s_2, \ldots, s_n, T \rangle$ of SOS such that
  - some of the sizes s_i sum up to exactly T

    if and only if
  - U can be partitioned into m pairwise disjoint sets from $\mathscr{S}$.

- It is convenient to view the elements of the universe U as positive integers, i.e.,

$$U = \{1, 2, \ldots, 3m\}.$$

- Let n be the number of sets in the given collection $\mathscr{S}$, i.e.,

$$\mathscr{S} = \{S_1, S_2, \ldots, S_n\}.$$

- Think of each set S_i in $\mathscr{S}$ as a bit vector $\vec{s}_i$ of dimension $3m$.

Subset-of-Sums Problem

- For example, let $U = \{1, 2, \ldots, 6\}$, and consider the collection $\mathscr{S} = \{S_1, S_2, S_3\}$ with three sets, where

$$S_1 = \{1,3,6\} \quad \text{corresponds to} \quad \vec{s}_1 = (1,0,1,0,0,1);$$
$$S_2 = \{3,4,6\} \quad \text{corresponds to} \quad \vec{s}_2 = (0,0,1,1,0,1);$$
$$S_3 = \{2,4,5\} \quad \text{corresponds to} \quad \vec{s}_3 = (0,1,0,1,1,0).$$

- Interpret the bit vectors $\vec{s}_i$ as positive integers s_i in $(n+1)$-ary representation.

- The base $n+1$ is chosen in order to avoid problems with the carry in the addition of the integers represented by the $\vec{s}_i$.

Subset-of-Sums Problem

- That is, for each i with $1 \leq i \leq n$, the integer s_i corresponding to the set S_i is defined by

$$s_i = \sum_{j \in S_i} (n+1)^{3m-j}.$$

- In the above example, we have

$$\begin{aligned} s_1 &= 4^5 + 4^3 + 4^0 = 1089; \\ s_2 &= 4^3 + 4^2 + 4^0 = 81; \\ s_3 &= 4^4 + 4^2 + 4^1 = 276. \end{aligned}$$

Subset-of-Sums Problem

- The universe U, which contains all integers j with $1 \leq j \leq 3m$, thus corresponds to the vector $\vec{1} = (1, 1, \ldots, 1)$ of dimension $3m$.

- Hence, defining the target sum T to be the integer represented by this vector in base $n+1$:

$$T = \sum_{j=0}^{3m-1} (n+1)^j,$$

it follows that

- U can be partitioned into m pairwise disjoint sets from $\mathscr{S}$ if and only if
- $\sum_{i=1}^{n} x_i s_i = T$ for suitably chosen coefficients $x_i \in \{0, 1\}$.

Subset-of-Sums Problem

- In particular, $x_i = 1$ if and only if
  S_i is one of the sets participating in the partition of U.

- In the above example,
  - $U = S_1 \cup S_3$ with S_1 and S_3 being disjoint sets from $\mathscr{S}$, and
  - for the boolean coefficient vector $\vec{x} = (1,0,1)$, we have

$$s_1 + s_3 = 1365 = T.$$

This completes the proof. ❑

Subset-of-Sums Problem

Definition

Let $\langle \vec{s}, T \rangle$ be a given SOS instance, where $\vec{s} = (s_1, s_2, \ldots, s_n)$ is a sequence of positive integers (the sizes) and T is a positive integer (the target). The sequence $\vec{s}$ of sizes is said to be *superincreasing* if for each i with $2 \leq i \leq n$,

$$s_i > \sum_{j=1}^{i-1} s_j.$$

Fact

For instances $\langle \vec{s}, T \rangle$ with a superincreasing sequence $\vec{s}$ of sizes, the problem SOS can be solved in deterministic polynomial time.

Proof: Exercise. ❑

Merkle and Hellman's Public-Key Cryptosystem

Step	Alice	Erich	Bob
1			chooses • a superincreasing sequence of sizes, $\vec{s} = (s_1, s_2, \ldots, s_n)$, • a prime $p > \sum_{i=1}^{n} s_i$, and • a multiplier $b \in \mathbb{Z}_p^*$, and computes the vector $\vec{t} = (t_1, t_2, \ldots, t_n)$ by the linear modular transformation $t_i = bs_i \bmod p$; $\vec{t}$ is public and $\vec{s}$, p, and b are private
2		$\Leftarrow \vec{t}$	

Merkle and Hellman's Public-Key Cryptosystem

Step	Alice	Erich	Bob
3	encrypts the message $\vec{m} = (m_1, m_2, \ldots, m_n)$ as $c = \sum_{i=1}^{n} m_i t_i$		
4		$c \Rightarrow$	
5			decrypts c by defining $T = b^{-1} c \bmod p$ and solving the SOS problem for the instance $\langle \vec{s}, T \rangle$

Table: Merkle and Hellman's public-key cryptosystem (continued)

Merkle and Hellman's Public-Key Cryptosystem

1. **Key Generation.** The legitimate receiver Bob chooses
   - a superincreasing sequence $\vec{s} = (s_1, s_2, \ldots, s_n)$ of sizes,
   - a prime number $p > \sum_{i=1}^{n} s_i$, and
   - a multiplier $b \in \mathbb{Z}_p^*$ (chosen so that $\vec{t}$ as defined below is *not* superincreasing).

   These values are his private key, i.e., his trapdoor information.

   He then determines his public key, which is a new vector

   $$\vec{t} = (t_1, t_2, \ldots, t_n)$$

   obtained by the following linear modular transformation:

   $$t_i = bs_i \bmod p.$$

2. **Communication.** Bob's public key $\vec{t}$ is now known to Alice.

Merkle and Hellman's Public-Key Cryptosystem

3. **Encryption.** If $\vec{m} = (m_1, m_2, \ldots, m_n) \in \{0,1\}^n$ is the message Alice wishes to encrypt, she computes

$$c = \sum_{i=1}^{n} m_i t_i.$$

4. **Communication.** Alice sends the ciphertext c to Bob.
5. **Decryption.** When Bob receives the ciphertext c, $0 \leq c \leq n(p-1)$, he uses his trapdoor information to invert the linear modular transformation previously applied. In particular, he computes the target sum

$$T = b^{-1} c \bmod p,$$

which can be used to recover the original message. Since $\vec{s}$ is a superincreasing sequence of sizes, Bob can recover $\vec{m}$ by efficiently solving SOS for the instance $\langle \vec{s}, T \rangle$.

Merkle and Hellman's Public-Key Cryptosystem

Example (Merkle–Hellman PKCS)

Bob chooses his public key consisting of

- the superincreasing vector $\vec{s} = (2, 3, 6, 12, 25, 51, 101, 203, 415)$,
- the prime number $p = 821$ satisfying $p > \sum_{i=1}^{n} s_i$, and
- the multiplier $b = 444$.

Next, he computes his public key via $t_i = bs_i \bmod p = 444s_i \bmod 821$:

$$\vec{t} = (67, 511, 201, 402, 427, 477, 510, 643, 356).$$

Note that, unlike $\vec{s}$, $\vec{t}$ is *not* superincreasing.

Suppose Alice's message is $m = (1, 0, 1, 1, 0, 1, 0, 0, 1)$.

Merkle and Hellman's Public-Key Cryptosystem

Example (Merkle–Hellman PKCS: continued)

Using Bob's public key $\vec{t}$, she computes the ciphertext

$$c = \sum_{i=1}^{n} m_i t_i = 67 + 201 + 402 + 477 + 356 = 1503$$

and sends c to Bob.

Using the extended Euclidean algorithm, Bob computes

$$b^{-1} = 444^{-1} \bmod 821 = 723$$

and thus the target sum

$$T = b^{-1} c \bmod p = 723 \cdot 1503 \bmod 821 = 486.$$

Now, he applies the SOS algorithm for superincreasing sequences of sizes to $\langle \vec{s}, T \rangle$ to decrypt c and to obtain the plaintext m again.

Rabi, Rivest, and Sherman's Protocols

- In 1984, Rivest and Sherman proposed a protocol for secret-key agreement that is based on a *"strongly noninvertible, associative one-way function"*

$$\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*.$$

- In 1997, Rabi and Sherman proposed
  - a related digital signature scheme and
  - related multi-party extensions of these protocols, assuming σ to be *commutative* in addition.
- These types of one-way functions (in a complexity-theoretic worst-case model) have been further studied by
  - Hemaspaandra and Rothe in 1999,
  - Hemaspaandra, Pasanen, and Rothe in 2001, and
  - Hemaspaandra, Rothe, and Saxena in 2008.

Rabi, Rivest, and Sherman's Protocols

Step	Alice	Erich	Bob
1	chooses two large random strings, x and y, keeps x secret and computes $x\sigma y$		
2		$\langle y, x\sigma y\rangle \Rightarrow$	
3			chooses a large random string, z, keeps z secret and computes $y\sigma z$
4		$\Leftarrow y\sigma z$	
5	computes her key $k_A = x\sigma(y\sigma z)$		computes his key $k_B = (x\sigma y)\sigma z$

Table: Rivest–Sherman secret-key agreement protocol

Complexity-Theoretic (Worst-Case) One-Way Function

Definition (one-way function)

Let $f : \Sigma^* \to \Sigma^*$ be any function. Let D_f denote the domain of f and let R_f denote the range of f.

1. f said to be *honest* if there exists a polynomial p such that for each $y \in R_f$ there exists some $x \in D_f$ such that $y = f(x)$ and $|x| \leq p(|y|)$.

2. We say that f is FP-*invertible* if there exists a function $g \in \mathrm{FP}$ such that for each $y \in R_f$,
$$f(g(y)) = y.$$

3. We say that f is a *one-way function* if f is honest, $f \in \mathrm{FP}$, and f is not FP-invertible.

Do There Exist One-Way Functions?

Remark: For example, the function

$$f(x) = 1^{\lceil \log |x| \rceil}$$

is *dishonest* because the preimage of any string 1^m in the range of f has exponential size in m.

Thus, trivially, no FP function can invert f (although, clearly, $f \in \mathrm{FP}$).

Fact

There exist one-way functions if and only if $\mathrm{P} \neq \mathrm{NP}$.

Proof: We show: $\mathrm{P} = \mathrm{NP} \iff$ every honest (partial) FP function can be inverted in polynomial time.

Do There Exist One-Way Functions?

($\Rightarrow$) Assume $\mathrm{P} = \mathrm{NP}$ and let f by an honest, partial function in FP.

Let p be the polynomial witnessing the honesty of f.

Define the set

$$\text{Prefix-Inv}_f = \{\langle x, y\rangle \,\big|\, x \text{ is prefix of a string } z,\ |z| \leq p(|y|), \text{ and } f(z) = y\}.$$

Since $f \in \mathrm{FP}$, $\text{Prefix-Inv}_f \in \mathrm{NP}$; an NP algorithm, on input $\langle x, y\rangle$:

- guesses z with $|z| \leq p(|y|)$;
- checks whether x is a prefix of z;
- checks whether $f(z) = y$
- accepts if and only if both tests are successful.

Do There Exist One-Way Functions?

Since $\mathrm{P} = \mathrm{NP}$ by our assumption, $\text{Prefix-Inv}_f \in \mathrm{P}$.

Hence, f can be inverted with oracle Prefix-Inv_f (or, with a hypothetical polynomial-time algorithm for it) by the following prefix search:

$\textsc{Construct-Inverse}^{\text{Prefix-Inv}_f}(y)$ {
  $x := \varepsilon$; *(* ε *is the empty string* *)*
  `loop` {
    `if` $(\langle x0, y\rangle \in \text{Prefix-Inv}_f)$ $x := x0$
    `else if` $(\langle x1, y\rangle \in \text{Prefix-Inv}_f)$ $x := x1$
          `else exit` };
  `if` $(f(x) = y)$ `return` x
  `else reject`; }

Do There Exist One-Way Functions?

($\Rightarrow$) Let M be an arbitrary NPTM. Let p be a polynomial such that paths of $M(y)$ can be encoded by strings in $\{0,1\}^*$ of length at most $p(|y|)$.

We show: The language of M, $L(M)$, is in P.

Define the set

$$\mathrm{Comp}_M = \{\langle x, y\rangle \,\big|\, x \text{ with } |x| \leq p(|y|) \text{ encodes an accepting path of } M(y)\}.$$

Define the function

$$f(\langle x, y\rangle) = \begin{cases} y & \text{if } \langle x, y\rangle \in \mathrm{Comp}_M \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Clearly, $\mathrm{Comp}_M \in \mathrm{P}$, thus $f \in \mathrm{FP}$.

Do There Exist One-Way Functions?

If $y \in R_f$, then there exists a pair $\langle x, y \rangle$ such that

- $f(\langle x, y \rangle) = y$ and
- $|\langle x, y \rangle|$ is in $\mathscr{O}(p(|y|) + |y|)$, which is polynomial in $|y|$.

Hence, f is honest.

By assumption, f (as an honest, partial FP function) can be inverted in polynomial time: $(\exists g \in \mathrm{FP})(\forall y \in R_f)[f(g(y)) = y]$.
Since $f, g \in \mathrm{FP}$, we have $R_f \in \mathrm{P}$. Further, it holds that

$$\begin{aligned} y \in R_f &\iff f(\langle x, y \rangle) = y \text{ for some } \langle x, y \rangle \in \mathrm{Comp}_M \\ &\iff M \text{ accepts } y \text{ on some path } x \\ &\iff M \text{ accepts } y \\ &\iff y \in L(M). \end{aligned}$$

Hence, $L(M) = R_f \in \mathrm{P}$, so $\mathrm{P} = \mathrm{NP}$. ❑

Strong One-Way Function

Definition

Let $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ be any partial one-way function.

1. We say σ is *strong* if σ is
   - polynomial-time computable,
   - s-honest, and
   - strongly noninvertible.

2. We say that σ is *s-honest* if there exists a polynomial p such that both (a) and (b) are true:
   (a) For each $x, z \in \Sigma^*$ with $x\sigma y = z$ for some $y \in \Sigma^*$, there exists some string $\tilde{y} \in \Sigma^*$ such that $x\sigma\tilde{y} = z$ and $|\tilde{y}| \leq p(|x| + |z|)$.

   (b) For each $y, z \in \Sigma^*$ with $x\sigma y = z$ for some $x \in \Sigma^*$, there exists some string $\tilde{x} \in \Sigma^*$ such that $\tilde{x}\sigma y = z$ and $|\tilde{x}| \leq p(|y| + |z|)$.

Strong One-Way Function

3. We say that σ is *(polynomial-time) invertible with respect to the first argument* if there exists an inverter $g_1 \in \mathrm{FP}$ such that for each $z \in R_\sigma$ and for all $x, y \in \Sigma^*$ with $(x, y) \in D_\sigma$ and $x \sigma y = z$, we have

$$x \sigma g_1(\langle x, z \rangle) = z.$$

4. We say that σ is *(polynomial-time) invertible with respect to the second argument* if there exists an inverter $g_2 \in \mathrm{FP}$ such that for each $z \in R_\sigma$ and for all $x, y \in \Sigma^*$ with $(x, y) \in D_\sigma$ and $x \sigma y = z$, we have

$$g_2(\langle y, z \rangle) \sigma y = z.$$

5. We say that σ is *strongly noninvertible* if σ is
   - neither invertible with respect to the first argument
   - nor invertible with respect to the second argument.

Associativity for Total One-Way Functions

- Associativity ensures that the Rivest–Sherman protocol works.
  Suppose for the moment that the function σ were total.
  Then, associativity of σ means

$$(x\sigma y)\sigma z = x\sigma(y\sigma z)$$

  for all $x, y, z \in \Sigma^*$.

- This property guarantees that Alice and Bob indeed compute the same secret key:

$$k_A = x\sigma(y\sigma z) = (x\sigma y)\sigma z = k_B.$$

- This notion of associativity is meaningful for total functions, yet it is not meaningful for nontotal two-argument functions.

Weak Associativity for Nontotal One-Way Functions

- Consider the following attempt to capture associativity for nontotal functions that is due to Rabi and Sherman (1997):
  σ is *weakly associative* if

$$(x\sigma y)\sigma z \quad = \quad x\sigma(y\sigma z) \tag{1}$$

  for all $x, y, z \in \Sigma^*$ such that each of the following are defined:
  - $x\sigma y$,
  - $y\sigma z$,
  - $(x\sigma y)\sigma z$, and
  - $x\sigma(y\sigma z)$
- However, this definition attempt fails to do the job for nontotal functions. **What is wrong with it?**

Weak Associativity for Nontotal One-Way Functions

- Consider, for example, a function σ such that
  - $0\sigma 1 = 0$ and $1\sigma 0 = 1$,
  - yet σ is not defined on the pair $(0,0)$.

  Then $0\sigma(1\sigma 0) = 0\sigma 1 = 0$, yet σ is not defined on $((0\sigma 1),0) = (0,0)$.
- Thus, (1) has the form "undefined $= 0$."
- But weak associativity fails to evaluate (1) as being false for these values of $x = 0$, $y = 1$, and $z = 0$.
- When defining associativity for partial (including both total and nontotal) functions, it seems more natural to require that both sides of (1) stand or fall together:
  - Either both sides of (1) should be defined and equal,
  - or each side should be undefined.

Kleene's Distinction Between Complete and Weak Equality

- This observation is related to the distinction between "complete equality" and "weak equality" of partial functions (Kleene 1952, pp. 327–328):

  We now introduce "$\psi(x_1,\ldots,x_n) =_c \chi(x_1,\ldots,x_n)$" to express, for particular $x_1,\ldots,x_n$, that if either of $\psi(x_1,\ldots,x_n)$ and $\chi(x_1,\ldots,x_n)$ is defined, so is the other and the values are the same (and hence if either of $\psi(x_1,\ldots,x_n)$ and $\chi(x_1,\ldots,x_n)$ is undefined, so is the other). The difference in the meaning of (i) "$\psi(x_1,\ldots,x_n) =_w \chi(x_1,\ldots,x_n)$" and (ii) "$\psi(x_1,\ldots,x_n) =_c \chi(x_1,\ldots,x_n)$" comes when one of $\psi(x_1,\ldots,x_n)$ and $\chi(x_1,\ldots,x_n)$ is undefined.

  Then (i) is undefined, while (ii) is true or false according as the other is or is not undefined.

Associativity for Partial One-Way Functions

Definition

Let $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ be any partial function. Let $\perp$ be a symbol indicating, by "$x\sigma y = \perp$," that σ is undefined on (x, y). Let $\Gamma = \Sigma^* \cup \{\perp\}$ be an extension of Σ^*. Define an extension $\widehat{\sigma} : \Gamma \times \Gamma \to \Gamma$ of σ by

$$x\widehat{\sigma}y = \begin{cases} x\sigma y & \text{if } x \neq \perp \neq y \text{ and } (x,y) \in D_\sigma \\ \perp & \text{otherwise.} \end{cases} \tag{2}$$

1. We say that σ is *associative* if for each $x, y, z \in \Sigma^*$,

$$(x\widehat{\sigma}y)\widehat{\sigma}z = x\widehat{\sigma}(y\widehat{\sigma}z).$$

2. We say that σ is *commutative* if for each $x, y \in \Sigma^*$,

$$x\widehat{\sigma}y = y\widehat{\sigma}x.$$

Do Strong, Total, Associative, Commutative OWFs Exist?

Recall:

Fact

There exist one-way functions if and only if $\mathrm{P} \neq \mathrm{NP}$.

Question: Can we characterize the existence of one-way functions that are

- strongly noninvertible,
- total,
- associative, and
- commutative

by a plausible complexity-theoretic condition?

Do Strong, Total, Associative, Commutative OWFs Exist?

Answer:

Theorem (Hemaspaandra and Rothe (1999))

There exist total, strong, commutative, associative one-way functions if and only if $\mathrm{P} \neq \mathrm{NP}$.

Do Strong, Total, Associative, Commutative OWFs Exist?

Proof:

- By the above fact, it remains to prove that $\mathrm{P} \neq \mathrm{NP}$ implies the existence of such one-way functions. So assume that $\mathrm{P} \neq \mathrm{NP}$.
- Let L be some set in NP such that $L \notin \mathrm{P}$, and let M be some given NPTM accepting L.
- Recall that a *witness for "$x \in L$"* is any string $w \in \Sigma^*$ encoding an accepting path of M on input x. For instance, recall the example in the proof showing NP-completeness of SOS, where the yes-instance $\langle s_1, s_2, s_3, T \rangle = \langle 1089, 81, 276, 1365 \rangle$ of SOS was constructed. This is *witnessed* by the vector $\vec{x} = (1, 0, 1)$, since

$$s_1 + s_3 = 1089 + 276 = 1365 = T.$$

Do Strong, Total, Associative, Commutative OWFs Exist?

- For each $x \in L$, the *set of witnesses for "$x \in L$"* is defined by

$$\mathrm{Wit}_M(x) = \{w \in \Sigma^* \mid w \text{ is a witness for "}x \in L\text{"}\}.$$

- Note that $\mathrm{Wit}_M(x)$ is empty if and only if $x \notin L$.

- **Technical detail:** We assume that, for each $x \in L$,
  - each witness w for "$x \in L$" is of length $p(|x|)$ for some strictly increasing polynomial p, and
  - the length of w is strictly larger than the length of x.

  This assumption allows us to tell input strings in L apart from their witnesses.

Do Strong, Total, Associative, Commutative OWFs Exist?

Our Construction proceeds in two stages:

1. We first construct a *nontotal*, strong, commutative, associative one-way function σ from L.
   For illustration, we ask Officer σ for help:

Officer σ

2. We then extend σ to a *total* function $\hat{\sigma}$ that inherits all the other properties of σ (but the formal proof is left as an exercise).
   This is illustrated by Officer $\hat{\sigma}$:

Officer $\hat{\sigma}$

Do Strong, Total, Associative, Commutative OWFs Exist?

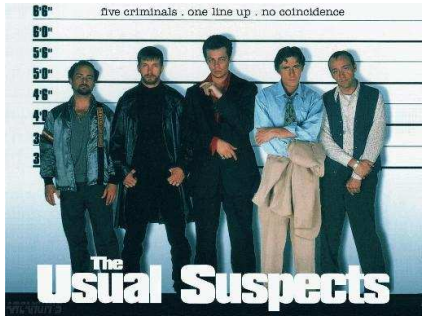
For the first stage, let $a, b \in \Sigma^*$ be any two given input strings. Define σ by

$$a\sigma b \;=\; \begin{cases} \langle x, \min(w_1, w_2)\rangle & \text{if } a = \langle x, w_1\rangle \text{ and } b = \langle x, w_2\rangle \\ & \text{for some } x \in \Sigma^* \text{ and } w_1, w_2 \in \mathrm{Wit}_M(x) \\ \langle x, x\rangle & \text{if } (a = \langle x, x\rangle \text{ and } b = \langle x, w\rangle) \\ & \text{or } (a = \langle x, w\rangle \text{ and } b = \langle x, x\rangle) \\ & \text{for some } x \in \Sigma^* \text{ and } w \in \mathrm{Wit}_M(x) \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $\min(w_1, w_2)$ denotes the lexicographical minimum of w_1 and w_2.

Officer σ Tells Her Story

-  is on duty today.
- On her desk, there is a list of the usual suspects

$$L = \left\{ \text{  }, \ldots \right\}$$

and there are many reports about crimes that happened recently.

Officer σ Tells Her Story

- Some of the reports contain the description of one of the usual suspects, say x, from the list L. Officer σ attaches a file copy to it, so the report now has the form $\langle x, x \rangle$, e.g.,

$$\langle x, x \rangle = \left\langle \; \text{  } \; , \; \text{  } \; \right\rangle$$

- Some reports contain the testimony w of an eye witness who has seen this suspect x on the scene of a crime. Officer σ attaches w to x, so the report now has the form $\langle x, w \rangle$, e.g.,

$$\langle x, w \rangle = \left\langle \; \text{  } \; , \; \boxed{\begin{array}{l}\text{witness's} \\ \text{testimony}\end{array}} \; \right\rangle$$

- There are also many other reports that contain neither the description of a suspect nor the testimony of a witness.

Officer σ Tells Her Story

- Officer σ is more than qualified for her job. That is why she can easily tell the description of a suspect apart from a witness's testimony:

$\langle x, x \rangle =$   $\neq$  witness's testimony $= \langle x, w \rangle$

- Better yet, she can easily check how reliable a witness is. Before filing her report, she verifies each witness testimony using a lie detector.

- Every once in a while, Officer Sigma takes two reports from the desk, say a and b. Holding a with her left hand and b with her right hand, she reads them both carefully.

- Officer σ then chooses one of a and b to pass on to her boss, Sgt. Ω, tossing the other one. Occasionally, she tosses them both.

Officer σ Tells Her Story

- How does she decide which reports to pass on and which to dump?
  - If $a = \langle x, w_1 \rangle =$ $\Big\langle$  , witness's testimony $\Big\rangle$ and

    $b = \langle x, w_2 \rangle =$ $\Big\langle$  , witness's testimony $\Big\rangle$,

    she chooses one of a and b to pass on, tossing the other one.
    She always chooses the shorter one (here a).
  - If one of a and b has the form $\langle x, x \rangle =$ $\Big\langle$  ,  $\Big\rangle$ and

    the other one has the form $\langle x, w \rangle =$ $\Big\langle$  , witness's testimony $\Big\rangle$,

    she passes report $\langle x, x \rangle$ on to Sergeant Ω, distractedly tossing $\langle x, w \rangle$ into the trashbin.
  - Otherwise, she rigorously tosses them both.

Do Strong, Total, Associative, Commutative OWFs Exist?

For the first stage, let $a, b \in \Sigma^*$ be any two given input strings. Define σ by

$$a\sigma b = \begin{cases} \langle x, \min(w_1, w_2)\rangle & \text{if } a = \langle x, w_1\rangle \text{ and } b = \langle x, w_2\rangle \\ & \text{for some } x \in \Sigma^* \text{ and } w_1, w_2 \in \mathrm{Wit}_M(x) \\ \langle x, x\rangle & \text{if } (a = \langle x, x\rangle \text{ and } b = \langle x, w\rangle) \\ & \text{or } (a = \langle x, w\rangle \text{ and } b = \langle x, x\rangle) \\ & \text{for some } x \in \Sigma^* \text{ and } w \in \mathrm{Wit}_M(x) \\ \text{undefined} & \text{otherwise,} \end{cases} \tag{3}$$

where $\min(w_1, w_2)$ denotes the lexicographical minimum of w_1 and w_2.

Do Strong, Total, Associative, Commutative OWFs Exist?

- It remains to show that σ has the desired properties.
- It is a matter of routine to check that σ is
  - commutative,
  - honest,
  - s-honest,
  - polynomial-time computable,
  - and not FP-invertible.
- In particular, σ is thus a one-way function.
- We now prove that σ is
  - strong and
  - associative.

σ is Strong

- For a contradiction, suppose that there is a polynomial-time inverter i with respect to the first argument of σ.
- That is, for each string z in the range of σ and for each fixed first argument $a \in \Sigma^*$ for which there exists a corresponding second argument $b \in \Sigma^*$ with $a\sigma b = z$, we have that $a\sigma i(\langle a, z\rangle) = z$.
- The inverter i can be used to decide the set L in polynomial time:
  1. Given any input string x, to decide whether or not x is in L, compute the string $u = i(\langle\langle x, x\rangle, \langle x, x\rangle\rangle)$.
  2. Compute the unique strings v and w for which $\langle v, w\rangle = u$, i.e., v and w are the projections of our pairing function at u.
  3. Accept x if and only if $v = x$ and $w \in \mathrm{Wit}_M(x)$.

σ is Strong

- The above algorithm runs in polynomial time and thus shows that L is in P, which contradicts our assumption that $L \notin \mathrm{P}$.
- Thus, σ is not invertible with respect to the first argument.
- An analogous argument shows that σ is not invertible with respect to the second argument either.
- It follows that σ is strongly noninvertible.

σ is Associative

- Let $a, b, c \in \Sigma^*$ be any fixed arguments for σ.
- Consider the projections of our pairing function at a, b, and c, respectively:
  - $a = \langle a_1, a_2 \rangle$,
  - $b = \langle b_1, b_2 \rangle$, and
  - $c = \langle c_1, c_2 \rangle$.
- Let $k \in \{0, 1, 2, 3\}$ be the number that counts how many of a_2, b_2, and c_2 are elements of $\mathrm{Wit}_M(a_1)$.
  For example, if $a_2 = b_2 \in \mathrm{Wit}_M(a_1)$, but $c_2 \notin \mathrm{Wit}_M(a_1)$, then $k = 2$.

σ is Associative

- We have to show that

$$(a\widehat{\sigma}b)\widehat{\sigma}c = a\widehat{\sigma}(b\widehat{\sigma}c), \tag{4}$$

where $\widehat{\sigma}$ is the extension of σ. There are two cases to distinguish.

Case 1: $a_1 = b_1 = c_1$ **and** $\{a_2, b_2, c_2\} \subseteq \{a_1\} \cup \mathbf{Wit}_M(a_1)$**.**

The intuition in this case is that the number of witnesses occurring in the arguments of σ are decreased by one as follows:

- If none of σ's arguments contains a witness for "$a_1 \in L$," then σ is undefined, so $\widehat{\sigma}$ outputs $\perp$.
- If exactly one of σ's arguments contains a witness for "$a_1 \in L$," then σ—and thus $\widehat{\sigma}$ as well—has the value $\langle a_1, a_1 \rangle$.
- If both of σ's arguments contain a witness for "$a_1 \in L$," then $\widehat{\sigma}$ outputs $\langle a_1, w \rangle$, where $w \in \{a_2, b_2, c_2\}$ is the lexicographically smaller of the two witnesses.

σ is Associative

From the above three subcases, we conclude the following:

- If $k \in \{0,1\}$ then
$$(a\widehat{\sigma}b)\widehat{\sigma}c = \perp = a\widehat{\sigma}(b\widehat{\sigma}c).$$
- If $k = 2$ then
$$(a\widehat{\sigma}b)\widehat{\sigma}c = \langle a_1, a_1\rangle = a\widehat{\sigma}(b\widehat{\sigma}c).$$
- If $k = 3$ then
$$(a\widehat{\sigma}b)\widehat{\sigma}c = \langle a_1, \min(a_2, b_2, c_2)\rangle = a\widehat{\sigma}(b\widehat{\sigma}c),$$
where again $\min(a_2, b_2, c_2)$ denotes the lexicographically smallest of a_2, b_2, and c_2.

σ is Associative

Case 2: Either $(a_1 \neq b_1$ **or** $a_1 \neq c_1$ **or** $b_1 \neq c_1)$ **or** $(a_1 = b_1 = c_1$ **and** $\{a_2, b_2, c_2\} \not\subseteq \{a_1\} \cup \mathbf{Wit}_M(a_1))$**.**

In either of these two subcases of Case 2, one can verify that

$$(a \widehat{\sigma} b) \widehat{\sigma} c = \perp = a \widehat{\sigma} (b \widehat{\sigma} c).$$

- In each of the above cases, (4) is satisfied. Hence, σ is associative.

How to make σ Total

- Fix any string $x_0 \notin L$ (one must exist, since $L \notin \mathrm{P}$).
- Let a_0 be the pair $\langle x_0, 1x_0 \rangle$. Note that a_0 is
  - neither of the form $\langle x, x \rangle$ for any $x \in \Sigma^*$,
  - nor of the form $\langle x, w \rangle$ for any $x \in \Sigma^*$ and any witness $w \in \mathrm{Wit}_M(x)$ (because $x_0 \notin L$ and thus does not have any witnesses).

  By definition of σ, for each y, $(a_0, y) \notin D_\sigma$ and $(y, a_0) \notin D_\sigma$.
- Define the total function $\hat{\sigma} : \Sigma^* \times \Sigma^* \to \Sigma^*$ as follows:
  - Whenever $(a, b) \in D_\sigma$, define $\hat{\sigma}(a, b) = \sigma(a, b)$;
  - otherwise, define $\hat{\sigma}(a, b) = a_0$.
- $\hat{\sigma}$ inherits each of the other properties of σ. ❑

Concluding Remarks

- Rabi and Sherman (1997) prove: No total, associative function can be injective.
- Homan (2004) generalizes this to "Tight lower bounds on the ambiguity of strong, total, associative, one-way functions."
- Hemaspaandra, Pasanen, and Rothe (2006) show: If $\mathrm{P} \neq \mathrm{NP}$ then some strongly noninvertible functions are invertible.

Concluding Remarks

- Hemaspaandra, Pasanen, and Rothe (2006) show: If $\mathrm{P} \neq \mathrm{NP}$ then some strongly noninvertible functions are invertible.

  Proof Idea: Assuming $\mathrm{P} \neq \mathrm{NP}$, let ρ be a total 2-ary one-way function. Define $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ as follows:

$$\sigma(a,b) = \begin{cases} 0\rho(x,y) & \text{if } (\exists x,y,z \in \Sigma^*)[a = 1\langle x,y\rangle \wedge b = 0z] \\ 0\rho(y,z) & \text{if } (\exists x,y,z \in \Sigma^*)[a = 0x \wedge b = 1\langle y,z\rangle] \\ 1xy & \text{if } (\exists x,y \in \Sigma^*) \\ & \quad [(a = 0x \wedge b = 0y) \vee (a = 1x \wedge b = 1y)] \\ ab & \text{if } a = \varepsilon \vee b = \varepsilon. \end{cases}$$

Concluding Remarks

- Hemaspaandra, Rothe, and Saxena (2008) show:

(s,t,c,a)	characterization
$(\mathrm{N},\mathrm{N},\mathrm{N},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{N},\mathrm{N},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{N},\mathrm{Y},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{N},\mathrm{Y},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{Y},\mathrm{N},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{Y},\mathrm{N},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{Y},\mathrm{Y},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{N},\mathrm{Y},\mathrm{Y},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$

(s,t,c,a)	characterization
$(\mathrm{Y},\mathrm{N},\mathrm{N},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{N},\mathrm{N},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{N},\mathrm{Y},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{N},\mathrm{Y},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{Y},\mathrm{N},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{Y},\mathrm{N},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{Y},\mathrm{Y},\mathrm{N})$	$\mathrm{P} \neq \mathrm{NP}$
$(\mathrm{Y},\mathrm{Y},\mathrm{Y},\mathrm{Y})$	$\mathrm{P} \neq \mathrm{NP}$

Shamir's No-Key Protocol: A Story

Step	Ella	Parents	Paula
1	Ella buys a box B and two padlocks, x and y, asks her parents to hand y over to Paula in the hospital, and keeps x for herself		
2	locks the message m in the box B using her padlock x		
3		$B_x \Rightarrow$	
4			puts her padlock y on B
5		$\Leftarrow B_x^y$	
6	removes her padlock x		
7		$B^y \Rightarrow$	
8			removes her padlock y and reads the message m

Shamir's No-Key Protocol: How to Do It Mathematically

Step	Ella	Parents	Paula
1	agree upon a large prime p, which is public		
2	computes $x = m^e \bmod p$ for message m and private key e satisfying $\gcd(e, p-1) = 1$		
3		$x \Rightarrow$	
4			computes $y = x^d \bmod p$ for her private key d satisfying $\gcd(d, p-1) = 1$
5		$\Leftarrow y$	
6	computes $z = y^{e^{-1}} \bmod p$		
7		$z \Rightarrow$	
8			computes and reads the message $m = z^{d^{-1}} \bmod p$