Hash Functions

P1. M is a message of any size; $64 \leq |H(M) = m| \leq constant.$

P2. $\forall M$ message, function $H(M)$ is easy to compute.

P3. For any given $m = H(M)$, it is hard (computationally infeasible) to find M.

P4. For any given $\langle M, H(M) \rangle$ it's hard (computationally infeasible) to find $M', M' \neq M$, such that $H(M') = H(M)$.

P5. (Although $\exists M, M' | H(M) = H(M')$ since $|H(M)| \leq$ constant) it is hard (computationally infeasible) to find any two messages $M, M', M \neq M'$, such that $H(M) = H(M')$.

NOTE: Attack *resistance*: P3= preimage, P4= second preimage; P5: collision
Properties P1-P3 are of a *one-way* function.
Properties P1-P4 are of a *weak one-way* function.
Properties P1-P5 are of a *strong one-way* function.

Relationships among Hash Functions Properties

P5 ==> P4

If a hash function is *collision resistant,* then *it is second-preimage resistant.*

Proof. Prove $\prec$P4 ==> $\prec$P5. Fix x_j and find distinct x_i such that $H(x_i) = H(x_j)$ (by $\prec$P4). Hence $\prec$P5 is true since (x_i,x_j) is a pair of distinct inputs having the same hash value.

P5 =/=> P3

A function that is *collision resistant* is *not* necessarily *preimage resistant*.

Proof. Assume P5 ==> P3 and provide a counter-example as follows. For example, let **g(x)** be a collision-resistant hash function such that $|g(x)| = n$ bits, and define function **h(x)** as follows:
h(x) = 1 || x, if |x| = n bits; h(x) = 0 || g(x), otherwise.
Hence, **h(x)** is a (n+1)-bit hash function that is not preimage resistant.

P4 =/=> P3

A function that is *second-preimage resistant* is *not* necessarily *preimage resistant*.

Proof. Assume P4 ==> P3 and provide a counter-example as follows. For example, let **h(x) = x, |x| = fixed length m**. h(x) is collision and second preimage resistant but not preimage resistant.

Attacks against One-Way Functions - Search Space

$|H(M)| = m$ bits, hash function has 2^m outputs.

Problem

Given hash function H, and a specific value $H(M)$ for M, if H is applied to k random inputs M_1',...,M_k', what is the value of k such that:

$$P\{H(M'_i) = H(M)\} = 0.5 \text{ for some } \quad i \in [1, k]$$

Solution ($k = 2^{m-1}$ implies no gain over full search).

- For a single value M' in $\{M_1',\ldots, M_k'\}$,

$$P\{H(M') = H(M)\} = \frac{1}{2^m} \text{ and}$$

$$P\{H(M') \neq H(M)\} = 1 - \frac{1}{2^m}$$

- For k values $\{M_1',\ldots, M_k'\}$ picked at random

$$P(H(M'_i) \neq H(M)\} = \left[1 - \frac{1}{2^m}\right]^k \text{ for all } i \in [1, k] \text{ and}$$

$$P(H(M'_i) = H(M)\} = 1 - \left[1 - \frac{1}{2^m}\right]^k \text{ for some } i \in [1, k],$$

$$\cong 1 - 1 + \frac{k}{2^m} \text{ for } m \geq 64, \text{ since } (1-a)^k \cong 1 - ka$$

$$= \frac{1}{2} \text{ for } k = 2^{m-1}.$$

WE MUST DO BETTER THAN RANDOM SEARCH TO DEFEAT THE COLLISION FREEDOM PROPERTY

- **“BIRTHDAY PARADOX”**

- **GENERAL CASE OF “BIRTHDAY PARADOX”**

- **OVERLAP BETWEEN TWO SETS OF MESSAGES**

- **BIRTHDAY ATTACK**

- **EXAMPLE OF BIRTHDAY ATTACK**

BIRTHDAY PARADOX

Find the minimum value of k such that:

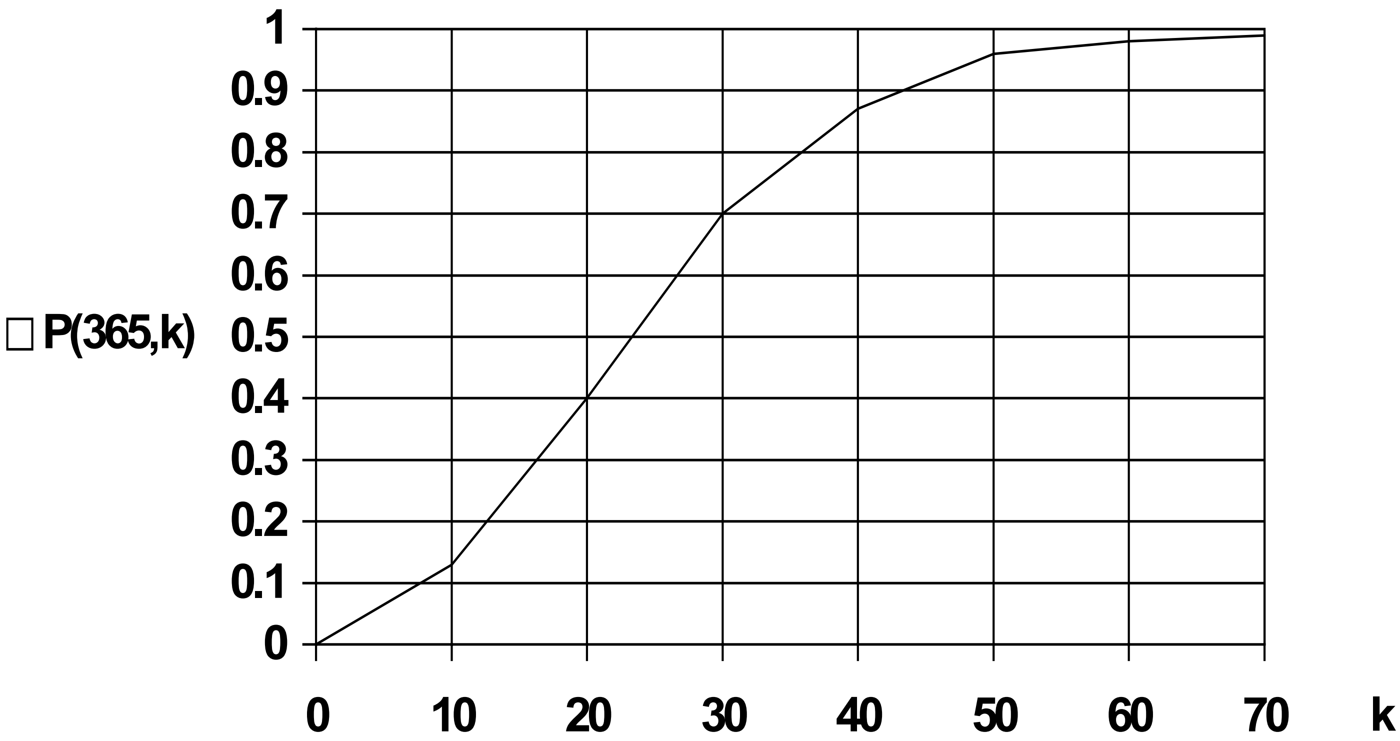
P\{at least one pair of k people have same birthday\} = 0.5

General problem

Let $P(n,k) = P$\{there is at least a pair of duplicates among k instances of a uniformely distributed random variable with values in [1,n]\}.
Find the minimum values of k such that $P(n,k) = 0.5$.

$P(365,k)=0.5$
$Q(365,k) = P$\{no pair of people have same birthday\}$=1-P(365,n)$.
Suppose $k<=365$ (otherwise there are duplicates).
Let N = number of ways to choose k values in [1,365] with no duplicates.
$N = 365*364*...*(365-k+1) = 365!/(365-k)!$
The total number of ways to choose k values in [1,365] is $T = 365^k$.
Thus, $Q(365,k) = N/T = 365!/(365-k)!/365^k$, and
$P(365,k) = 1 - 365!/(365-k)!/365^k$.

Diagram of $P(365,k)$ vs. k



GENERAL CASE OF DUPLICATIONS

Find $P(n,k) = P\{Xi = Xj \in \{X1,\ldots,Xk\}$ for some i, j, $\mathrm{X} = \mathrm{u.d.r.v.}\}$

$$P(n,k) = 1 - \frac{n!}{(n-k)!n^k} = 1 - \ 1 - \frac{1}{n} \ \ldots \ 1 - \frac{k-1}{n}$$

But $(1-x) \le e^{-x}$ for all $x \ge 0$, thus

$$P(n,k) > 1 - e^{-\frac{1}{n}} \cdot e^{-\frac{2}{n}} \cdot \ldots \cdot e^{-\frac{k-1}{n}} = 1 - e^{-\frac{k(k-1)}{2n}}$$

$$P(n,k) = \frac{1}{2} \Rightarrow \frac{1}{2} = 1 - e^{-\frac{k(k-1)}{2n}} \Rightarrow 2 = e^{\frac{k(k-1)}{2n}} \Rightarrow \ln(2) \cong \frac{k^2}{2n} \Rightarrow$$

$$k \cong \sqrt{2(\ln 2)n} \cong 1.17\sqrt{n} \cong \sqrt{n}$$

If $n = 2^m$, $k \cong 2^{\frac{m}{2}}$.

Inequality (1-x) <= e^{-x} for all x >= 0

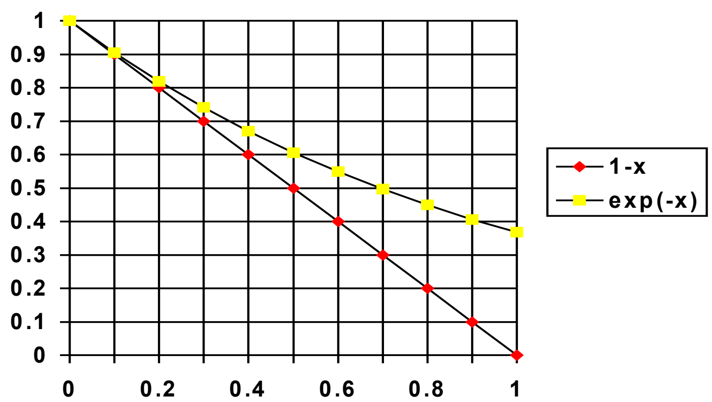
Let $f(x) = e^{-x}$.

$$\frac{df(x)}{dx} = -e^{-x} \Rightarrow \frac{df(0)}{dx} = -1.$$

The tangent to f at $x = 0$ is $ax + b$ where $a = -1$.

At $x = 0, f(0) = 1,$ so $a \cdot 0 + b = 1.$

So tangent at $x = 0$ is $1 - x$. Since tangent is under the curve of e^{-x}, the inequality holds .



OVERLAP BETWEEN TWO SETS OF MESSAGES

Let X be a random variable uniformly distributed over {1,...,n} and
x = {x_1, ..., x_k}, y = {y_1, ..., y_k} two sets of k instances (k $\leq$n) of X.

Problem: What is the probability that x and y overlap
i.e., (x_i,y_j) | x_i = y_j for some i, j in [1, k] ?

Solution: Given only x_1, $P(y_1 = x_1) = \frac{1}{n}$, $P(y_1 \neq x_1) = 1 - \frac{1}{n} \Rightarrow$

$$P(y_1 \neq x_1, \ldots, y_k \neq x_1) = {1 - \frac{1}{n}}^{k} \Rightarrow P(y_i = x_1 \text{ for some } i \in [1,k]) = 1 - {\frac{1}{n}}^{k}.$$

Assume $x_1, \ldots, x_k$ distinct and n, k are large.

$$P(y_1 \neq x_1, \ldots, y_k \neq x_1) = {1 - \frac{1}{n}}^{k} \Rightarrow P(x \neq y) = {{1 - \frac{1}{n}}^{k}}^{k} = {1 - \frac{1}{n}}^{k^2}$$

$$P(x_i = y_i \text{ for some } i, j \in [1,k]) = 1 - {1 - \frac{1}{n}}^{k^2} > 1 - {e^{-\frac{1}{n}}}^{k^2} = 1 - e^{-\frac{k^2}{n}}$$

$$1 - e^{-\frac{k^2}{n}} = \frac{1}{2} \Rightarrow k = \sqrt{(\ln 2)n} = 0.83\sqrt{n} \cong \sqrt{n}$$

If $n = 2^m, k = \sqrt{2^m} = 2^{\frac{m}{2}}$.

Birthday Attack

Let (A, B) be a *distributed service* where A signs clients' messages to be sent to B by appending an *encrypted m-bit digest*

A client's (chosen plaintext) *birthday attack* against distributed service (A, B):

1. The client generates $2^{m/2}$ variants of a message *acceptable* to A (i.e., A will sign any of these message variants) and $2^{m/2}$ variants of a forged message, which are *unaceptable* to A (i.e., A will not sign any of these message variants).

2. The client computes the digest for each message in the two sets and compares the two sets of digest to find a match ; *With probability 0.5, the client will find a match*; if no match is found, the client generates more messages and tries again until a match is found.

3. The client submits the *acceptable message* that has a match for A's signature. A signs it.

4. The client attaches A's signature to the *forged, matching message* and sends it to B.

5. The *forged message is accepted* by B as a valid message from A.

Lesson: *One should never sign anything without first adding a secret.*

Keyed Hash Functions = Message Authentication Codes (MACs)

(Weak) MAC

Q1. M is a message of any size; $|h_K(M) = m| \leq constant,$ K is secret.

Q2. $\forall$ message M, function $h_K(M)$ is easy to compute if K is known.

Q3. Given any $<M_i, h_K(M_i)>$ i = 1 ,…, n, it is hard (computationally infeasible) to find $< M, h_K(M)>$ such that $M \neq M_i$.

Strong MAC

Q1. M is a message of any size; $|h_K(M) = m| \leq constant,$ K is secret.

Q2. $\forall$ message M, function $h_K(M)$ is easy to compute if K is known.

Q4. Given any $<M_i, h_K(M_i)>$ i = 1 ,…, n, it is hard (computationally infeasible) to find $< M, h_K(M)> \neq <M_i, h_K(M_i)>$.

Obviously, Strong MAC => (Weak) MAC

Relationships between MAC Properties and Hash Function Properties

A (weak) MAC (keyed hash function) has the hash function properties.
That is, let H = h_K have properties Q1 - Q3. Then, H has properties

(1) P5 (collision resistance),
(2) P4 (second preimage resistance), and
(3) P3 (preimage resistance).

Proof.
(1) Prove that ∢P5 => ∢Q3. One can find a pair (M,M'), M ♑ M', such that H(M) = H(M') (possible by ∢P5). However, to compute H(M) = H(M') without the secret key K, call the MAC oracle and obtain <M_i,h_K (M_i)> i = 1 ,…, n, such that M_i ♑ M, for all i, and $M_j = M'$ for some j ♍ [1,n]. (This is allowed by the definition of the MAC oracle). Output <M,H(M')>. This implies ∢Q3.

(2) Property Q3 => P4 follows directly from (1) and P5 => P4.

(3) Prove ∢P3 => ∢Q3. Pick a random value H(M) and find M (possible by ∢P3). Then compute <M_i,h_K (M_i)> i = 1 ,…, n, such that M ♑ M_i, which is allowed by the definition of the MAC oracle. Output <M,H(M)>. This implies ∢Q3.