

## The Uniqueness of Unique Identifiers

### Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

### Abstract

This RFC provides information that may be useful when selecting a method to use for assigning unique identifiers to people.

### 1. The Issue

Computer systems require a way to identify the people associated with them. These identifiers have been called "user names" or "account names." The identifiers are typically short, alphanumeric strings. In general, these identifiers must be unique.

The uniqueness is usually achieved in one of three ways:

1) The identifiers are assigned in a unique manner without using information associated with the individual. Example identifiers are:

ax54tv  
cs00034

This method was often used by large timesharing systems. While it achieved the uniqueness property, there was no way of guessing the identifier without knowing it through other means.

2) The identifiers are assigned in a unique manner where the bulk of the identifier is algorithmically derived from the individual's name. Example identifiers are:

Craig.A.Finseth-1  
Finseth1  
caf-1  
fins0001

3) The identifiers are in general not assigned in a unique manner: the identifier is algorithmically derived from the individual's name

and duplicates are handled in an ad-hoc manner. Example identifiers are:

```
Craig.Finseth
caf
```

Now that we have widespread electronic mail, an important feature of an identifier system is the ability to predict the identifier based on other information associated with the individual. This other information is typically the person's name.

Methods two and three make such predictions possible, especially if you have one example mapping from a person's name to the identifier. Method two relies on using some or all of the name and algorithmically varying it to ensure uniqueness (for example, by appending an integer). Method three relies on using some or all of the name and selects an alternate identifier in the case of a duplication.

For both methods, it is important to minimize the need for making the adjustments required to ensure uniqueness (i.e., an integer that is not 1 or an alternate identifier). The probability that an adjustment will be required depends on the format of the identifier and the size of the organization.

## 2. Identifier Formats

There are a number of popular identifier formats. This section will list some of them and supply both typical and maximum values for the number of possible identifiers. A "typical" value is the number that you are likely to run into in real life. A "maximum" value is the largest number of possible (without getting extreme about it) values. All ranges are expressed as a number of bits.

### 2.1 Initials

There are three popular formats based on initials: those with one, two, or three letters. (The number of people with more than three initials is assumed to be small.) Values:

format	typical	maximum
I	4	5
II	8	10
III	12	15

You can also think of these as first, middle, and last initials:

I	4	5
F L	8	10
F M L	12	15

## 2.2 Names

Again, there are three popular formats based on using names: those with the first name, last name, and both first and last names. Values:

format	typical	maximum
First	8	14
Last	9	13
First Last	17	27

## 2.3 Combinations

I have seen these combinations in use ("F" is first initial, "M" is middle initial, and "L" is last initial):

format	typical	maximum
F Last	13	18
F M Last	17	23
First L	12	19
First M Last	21	32

## 2.4 Complete List

Here are all possible combinations of nothing, initial, and full name for first, middle, and last. The number of Middle names is assumed to be the same as the number of First names. Values:

format	typical	maximum
_ _ _	0	0
_ _ L	4	5
_ _ Last	9	13
_ M _	4	5
_ M L	5	10
_ M Last	13	18
_ Middle _	8	14
_ Middle L	12	19

_ Middle Last	17	27
F _ _	4	5
F _ L	5	10
F _ Last	13	18
F M _	5	10
F M L	12	15
F M Last	17	23
F Middle _	12	19
F Middle L	16	24
F Middle Last	21	32
First _ _	8	14
First _ L	12	19
First _ Last	17	27
First M _	12	19
First M L	16	24
First M Last	21	32
First Middle _	16	28
First Middle L	20	33
First Middle Last	26	40

### 3. Probabilities of Duplicates

As can be seen, the information content in these identifiers in no case exceeds 40 bits and the typical information content never exceeds 26 bits. The content of most of them is in the 8 to 20 bit range. Duplicates are thus not only possible but likely.

The method used to compute the probability of duplicates is the same as that of the well-known "birthday" problem. For a universe of N items, the probability of duplicates in X members is expressed by:

$$1 - \frac{N-1}{N} \times \frac{N-2}{N} \times \dots \times \frac{N-(X-1)}{N}$$

A program to compute this function for selected values of N is given in the appendix, as is its complete output.

The "1%" column is the number of items (people) before an organization of that (universe) size has a 1% chance of a duplicate. Similarly for 2%, 5%, 10%, and 20%.

bits	universe	1%	2%	5%	10%	20%
6	64	2	3	4	5	6
7	128	3	3	5	6	8
8	256	3	4	6	8	12
9	512	4	6	8	11	16
10	1,024	6	7	11	16	22
11	2,048	7	10	15	22	31
12	4,096	10	14	21	30	44
13	8,192	14	19	30	43	61
14	16,384	19	27	42	60	86
15	32,768	27	37	59	84	122
16	65,536	37	52	83	118	172
17	131,072	52	74	117	167	243
18	262,144	74	104	165	236	343
19	524,288	104	147	233	333	485
20	1,048,576	146	207	329	471	685
21	2,097,152	206	292	465	666	968
22	4,194,304	291	413	657	941	1369
23	8,388,608	412	583	929	1330	1936
24	16,777,216	582	824	1313	1881	2737
25	33,554,432	822	1165	1856	2660	3871
26	67,108,864	1162	1648	2625	3761	5474
27	134,217,728	1644	2330	3712	5319	7740
28	268,435,456	2324	3294	5249	7522	10946
29	536,870,912	3286	4659	7422	10637	15480
30	1,073,741,824	4647	6588	10496	15043	21891
31	2,147,483,648	6571	9316	14844	21273	30959

For example, assume an organization were to select the "First Last" form. This form has 17 bits (typical) and 27 bits (maximum) of information. The relevant line is:

17	131,072	52	74	117	167	243
----	---------	----	----	-----	-----	-----

For an organization with 100 people, the probability of a duplicate would be between 2% and 5% (probably around 4%). If the organization had 1,000 people, the probability of a duplicate would be much greater than 20%.

#### Appendix: Reuse of Identifiers and Privacy Issues

Let's say that an organization were to select the format:

First.M.Last-#

as my own organization has. Is the -# required, or can one simply do:

Craig.A.Finseth

for the first one and

Craig.A.Finseth-2

(or -1) for the second? The answer is "no," although for non-obvious reasons.

Assume that the organization has made this selection and a third party wants to send e-mail to Craig.A.Finseth. Because of the Electronic Communications Privacy Act of 1987, an organization must treat electronic mail with care. In this case, there is no way for the third party user to reliably know that sending to Craig.A.Finseth is (may be) the wrong party. On the other hand, if the -# suffix is always present and attempts to send mail to the non-suffix form are rejected, the third party user will realize that they must have the suffix in order to have a unique identifier.

For similar reasons, identifiers in this form should not be re-used in the life of the mail system.

#### Appendix: Perl Program to Compute Probabilities

```
#!/usr/local/bin/perl

for $bits (6..31) {
    &Compute($bits);
}
exit(0);

# -----

sub Compute {
    $bits = $_[0];
    $num = 1 << $bits;
    $cnt = $num;

    print "bits $bitsnumber $num:0;

    for ($prob = 1; $prob > 0.99; ) {
        $prob *= $cnt / $num;
        $cnt--;
    }

    print "", $num - $cnt, "$prob0;

    for (; $prob > 0.98; ) {
```

```

        $prob *= $cnt / $num;
        $cnt--;
    }
    print "", $num - $cnt, "$prob0;

    for (; $prob > 0.95; ) {
        $prob *= $cnt / $num;
        $cnt--;
    }
    print "", $num - $cnt, "$prob0;

    for (; $prob > 0.90; ) {
        $prob *= $cnt / $num;
        $cnt--;
    }
    print "", $num - $cnt, "$prob0;

    for (; $prob > 0.80; ) {
        $prob *= $cnt / $num;
        $cnt--;
    }
    print "", $num - $cnt, "$prob0;

    print "0;
}

```

## Appendix: Perl Program Output

```

bits 6  number 64:
      2      0.984375
      3      0.95361328125
      4      0.90891265869140625
      5      0.85210561752319335938
      6      0.78553486615419387817

bits 7  number 128:
      3      0.9766845703125
      3      0.9766845703125
      5      0.92398747801780700684
      6      0.88789421715773642063
      8      0.79999355674331695809

bits 8  number 256:
      3      0.988311767578125
      4      0.97672998905181884766
      6      0.94268989971169503406
      8      0.89542306910786462204
     12      0.76969425214152431547

```

bits 9 number 512:  
4 0.98832316696643829346  
6 0.97102570187075798458  
8 0.94652632751096643648  
11 0.89748056780293572476  
16 0.78916761796439427457

bits 10 number 1024:  
6 0.98543241551841020964  
7 0.97965839745873206645  
11 0.94753115178840541244  
16 0.8888886335604777014  
22 0.79677613655632184564

bits 11 number 2048:  
7 0.98978773152834598203  
10 0.97823367137821537476  
15 0.94990722378677450166  
22 0.89298119682681720288  
31 0.79597589885472519455

bits 12 number 4096:  
10 0.98906539062491305447  
14 0.97800426773009718762  
21 0.94994111694430838355  
30 0.89901365764115603874  
44 0.79312138620093930452

bits 13 number 8192:  
14 0.98894703242829806733  
19 0.97932692503837115439  
30 0.94822407309193512681  
43 0.89545741661906652631  
61 0.7993625840767998314

bits 14 number 16384:  
19 0.98961337517641645434  
27 0.97879319536756481668  
42 0.94876352395820107155  
60 0.89748107890372830209  
86 0.79973683158771624591

bits 15 number 32768:  
27 0.98934263776790121181  
37 0.97987304880641035165  
59 0.94909471808051404373  
84 0.89899774209805793923  
122 0.79809378598190949816



bits 16 number 65536:

37	0.98988724065590050216
52	0.97996496661944154649
83	0.94937874420413270737
118	0.89996948010355670711
172	0.79884228150816105618

bits 17 number 131072:

52	0.98993311138884398925
74	0.97960010416289267088
117	0.94952974978505377823
167	0.89960828942716541956
243	0.79894309171178368167

bits 18 number 262144:

74	0.98974844864797828503
104	0.97977315557223210174
165	0.94968621078621640041
236	0.8995926348279144058
343	0.7994422793765953994

bits 19 number 524288:

104	0.98983557888923057178
147	0.97973841652874515962
233	0.94974719445364064185
333	0.89991342619657743729
485	0.79936749144148444568

bits 20 number 1048576:

146	0.98995567500195758015
207	0.97987072919607220989
329	0.94983990872655321702
471	0.89980857451706741656
685	0.79974215234216872172

bits 21 number 2097152:

206	0.98998177463778547214
292	0.97994400939715686771
465	0.94985589918092261374
666	0.89978055267663470396
968	0.79994886751736571373

bits 22 number 4194304:

291	0.98999013137747737812
413	0.97991951242142538714
657	0.94991674892578203959
941	0.89991652739633254399
1369	0.79989205747440361716

bits 23 number 8388608:  
412 0.98995762604049764022  
583 0.97997846530691334888  
929 0.94991024716640248826  
1330 0.89999961063320443877  
1936 0.79987028265451087794

bits 24 number 16777216:  
582 0.98997307486745211857  
824 0.97999203469417239809  
1313 0.94995516684099989835  
1881 0.89997049960675035152  
2737 0.79996700222056416063

bits 25 number 33554432:  
822 0.98999408609360783906  
1165 0.9799956928177964155  
1856 0.9499899669674316538  
2660 0.8999664414095410736  
3871 0.79992328289672998132

bits 26 number 67108864:  
1162 0.98999884535478044345  
1648 0.9799801637652703068  
2625 0.94997437525354821997  
3761 0.89999748465616635773  
5474 0.79993922903192515861

bits 27 number 134217728:  
1644 0.9899880636014986024  
2330 0.97998730103356856969  
3712 0.94997727934463771504  
5319 0.89998552434244594167  
7740 0.79999591580103557309

bits 28 number 268435456:  
2324 0.98999458855588851058  
3294 0.97999828329325222587  
5249 0.94998397932368705554  
7522 0.89998576049206902017  
10946 0.79999058777500076101

bits 29 number 536870912:  
3286 0.98999717306002099626  
4659 0.97999160965267329004  
7422 0.94999720388831232487  
10637 0.89999506567702891591  
15480 0.7999860979665908145

bits 30 number 1073741824:  
4647 0.98999674474047760775  
6588 0.97999531736215383937  
10496 0.94999806770951356061  
15043 0.89999250738244507275  
21891 0.79999995570982085358

bits 31 number 2147483648:  
6571 0.98999869761078929109  
9316 0.97999801528523688976  
14844 0.94999403283519279206  
21273 0.89999983631135749285  
30959 0.79999272222201334159

#### References

Bruce Lansky (1984). The Best Baby Name Book. Deephaven, MN:  
Meadowbrook. ISBN 0-671-54463-2.

Lareina Rule (1988). Name Your Baby. Bantam. ISBN 0-553-27145-8.

#### Security Considerations

Security issues are not discussed in this memo.

#### Author's Address

Craig A. Finseth  
Networking Services  
Computer and Information Services  
University of Minnesota  
130 Lind Hall  
207 Church St. SE  
Minneapolis, MN 55455-0134

EMail: Craig.A.Finseth-1@umn.edu or  
fin@unet.umn.edu

Phone: +1 612 624 3375  
Fax: +1 612 626 1002

