COMP128: A Birthday Surprise

Stuart Wray
swray@bournemouth.ac.uk

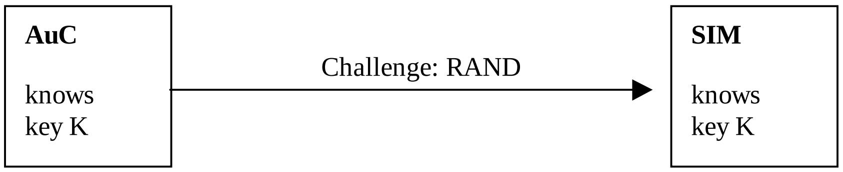
11th May 2003

Abstract

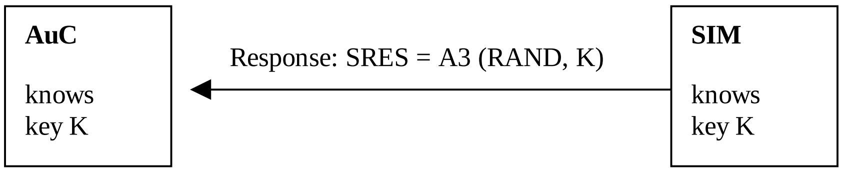
In 1998 Wagner, Goldberg and Briceno published an attack [1] on the COMP128 cryptographic hash algorithm, used for both session key generation and authentication in GSM phones. This attack exploits the structure of COMP128 in which "the birthday paradox guarantees that collisions will occur pretty rapidly" [1]. However, the birthday paradox does not in fact guarantee this, and in principle a small number of keys will be immune to their attack. Surprisingly, in practice a rather large number of keys are immune to their attack: around 2^{76}. If COMP128 were composed of truly random sub-functions then a deeper collision-inducing attack on this stronger subset of keys would not be feasible. However, experiment shows that such a deeper attack does work, although the new attack requires considerably more challenges than the old one.

1. GSM security and COMP128

Each SIM card in its GSM handset contains a different 128 bit secret key K, known only to it and to an Authentication Centre (AuC) in the network. When a GSM phone call is being set up the AuC sends RAND, a freshly generated 128 bit random number, via the network and the GSM handset to the SIM:



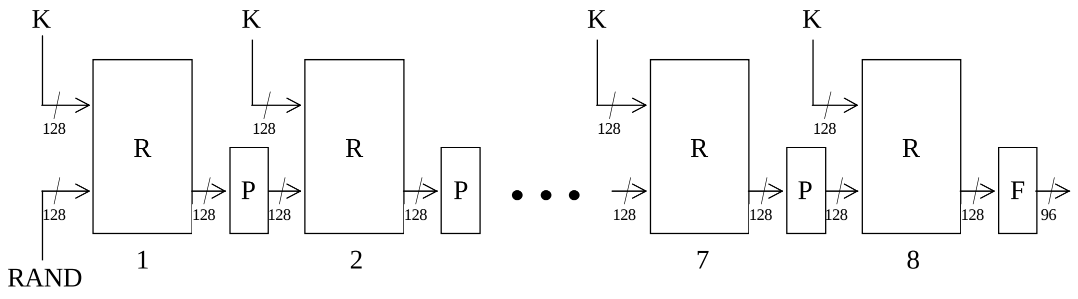
The SIM combines RAND and K using a hash function called A3, which gives a 32 bit "signed response" SRES. The SIM sends SRES back to the AuC via the handset and the network:



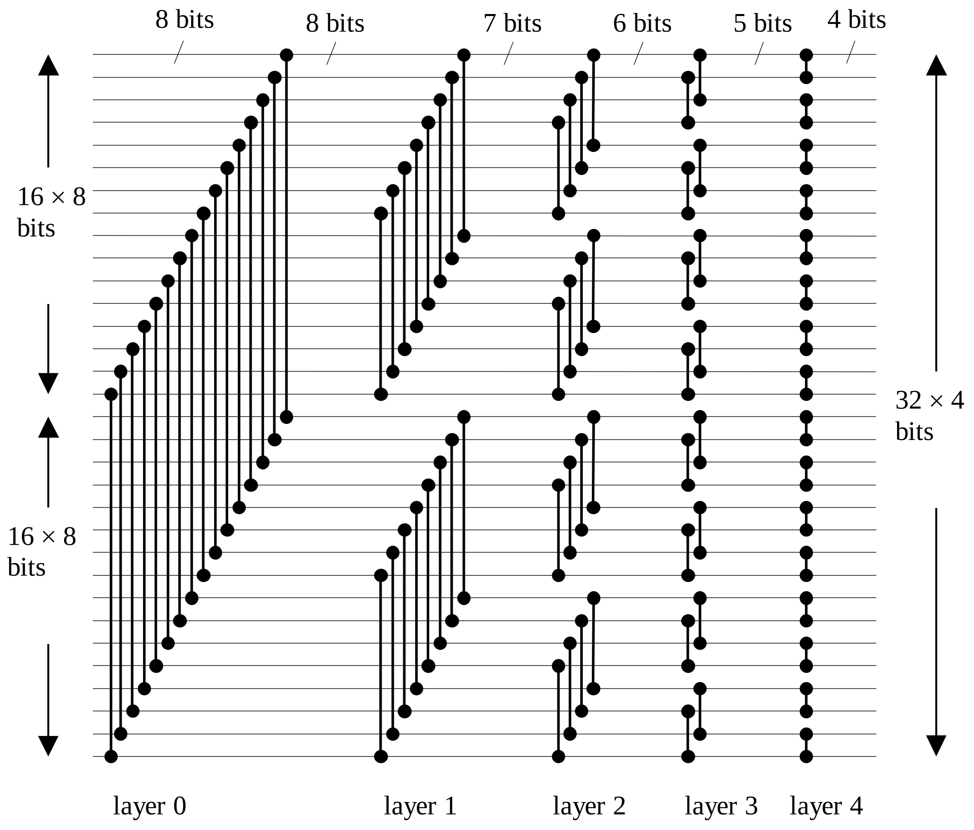
The AuC compares SRES to the value which it computed using its own copy of RAND and K. If they are equal, the AuC believes that the SIM is authentic and the call is allowed to proceed.

The speech exchanged between the GSM handset and the network is encrypted using an algorithm called A5 which has a 64 bit session key. For each new call the required A5 session key is generated using a hash function called A8. This takes the same 128 bit challenge and 128 bit key K and produces the 64 bit session key, so no further exchange of data is required for this step.

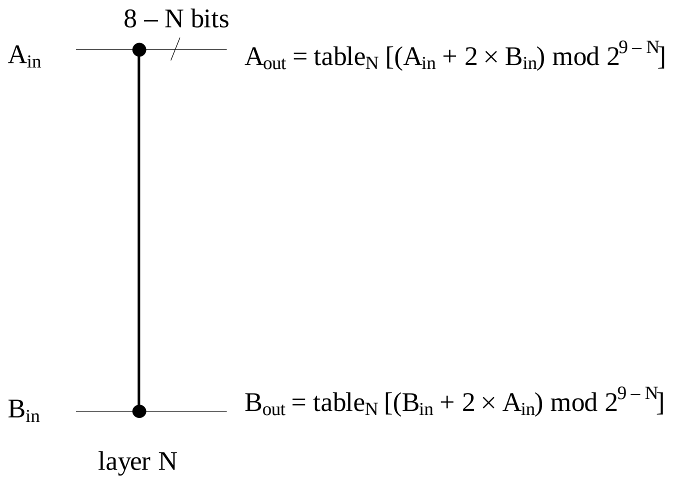
Both of the hash functions A3 and A8 are in fact implemented using the COMP128 algorithm: A3 and A8 simply return different portions of the 96 bit output from the last of COMP128's 8 rounds:



R is the round function, described below. P is a permutation and F is a permutation/selection.
Each round R is a so-called "butterfly" network:



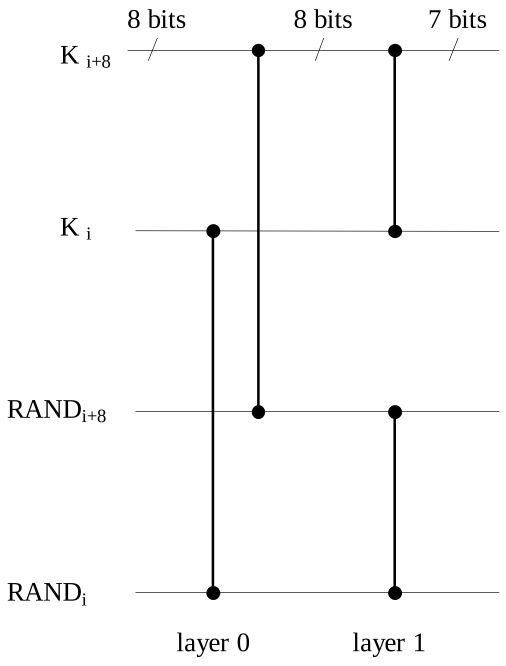
In each layer there are 16 combining operations, each taking a pair of inputs to a pair of outputs:


8 – N bits
A_{in}
$A_{out} = table_N [(A_{in} + 2 \times B_{in}) \bmod 2^{9-N}]$
B_{in}
$B_{out} = table_N [(B_{in} + 2 \times A_{in}) \bmod 2^{9-N}]$
layer N

(For example, $table_0$ has 512 entries, each an 8 bit value, and $table_4$ has 32 entries, each a 4 bit value.)

2. The WGB Attack

In 1998 Wagner, Goldberg and Briceno published an attack [1] in which the key K can be inferred by sending the SIM a series of chosen RAND challenges and analysing its responses. The WGB attack relies on the fact that collisions can be made to occur at layer 1 of the first round:



As the diagram indicates, K can be attacked piecemeal, since at layer 1 the 4 × 7 = 28 bits of output only depend on the pair of bytes (K_i , K_{i+8}) from the key and the pair ($RAND_i$, $RAND_{i+8}$) from the challenge. The attacker iterates through the values of ($RAND_i$, $RAND_{i+8}$) while keeping the other bytes of the RAND challenge the same until a collision happens in the 96 bit output of the COMP128 function. This is overwhelmingly likely to happen before the range of ($RAND_i$, $RAND_{i+8}$) challenges is exhausted, because 28 bits of output is small in comparison to the 2^{16} inputs which can be used. (This is the “birthday paradox”.)

Having induced such a collision, the attacker now has two values of ($RAND_i$, $RAND_{i+8}$) which produce the same output, almost certainly because of a collision at layer1 of round 1. The attacker can now iterate through the values of (K_i , K_{i+8}) in her own computer, searching for the key bytes which would also produce a collision for those same values of ($RAND_i$, $RAND_{i+8}$). By repeating this process 8 times all 16 bytes of K can be extracted.

The expected number of challenges before this attack succeeds is around 150,000. The WGB paper demonstrated that this was a realistic attack when the SIM was in the attacker's physical possession by using a smartcard reader to issue these challenges and extract its key. At 6.25 challenges a second, this took about 8 hours.

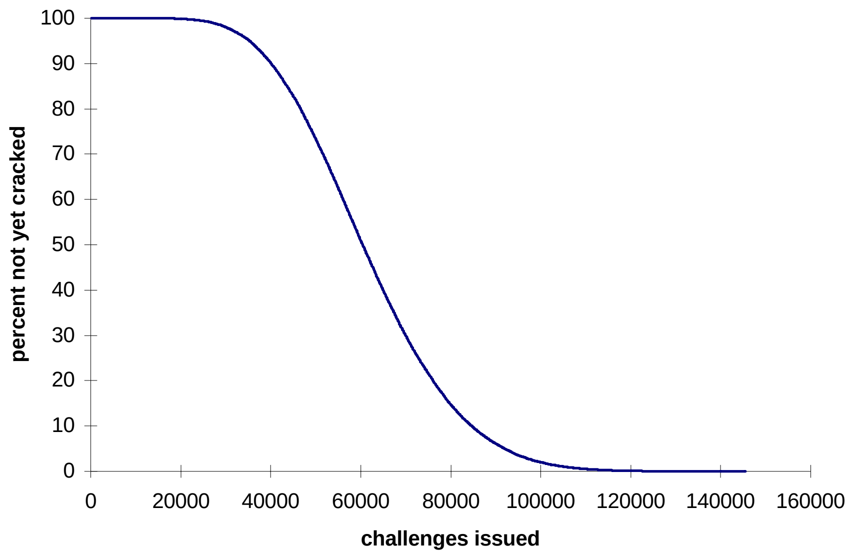
There has been speculation that this attack could also be launched over the air, and thereby clone or eavesdrop on a phone which was not in the attackers physical possession. There appears to be little difficulty with this in theory, the nuisance being that only the 32 bits of the COMP128 result delivered by A3 are available, rather than the full 96 bits. This means that the rate of "random" collisions from layers or rounds other than layer 1 of round 1 increases dramatically, to a rate comparable to that of collisions induced by the attack. However, it is unlikely (around 1 in 14,000) that such a random collision will have any valid (K_i , K_{i+8}) corresponding to an apparently successful pair of ($RAND_i$, $RAND_{i+8}$) challenges. After such a red-herring, the attacker must therefore simply go back to the SIM and try further challenges. Eventual success is overwhelmingly likely.

3. Improving the WGB attack

The original description of the attack quoted around 150,000 challenges as being needed to extract a key. Tantalisingly their report says "we have some optimisations to get this number down a bit" [1] without saying what those optimisations are. Before demonstrating how to thwart the WGB attack we will first show how to make it quicker.

The structure of the first two layers of COMP128 is not an evenly distributed random function. Some ($RAND_i$, $RAND_{i+8}$) challenges are more productive than others, because they have the potential to cause collisions for several different values of (K_i , K_{i+8}). It makes sense to use the most productive challenges first. This can be done by exhaustively iterating through all (K_i , K_{i+8}) and for each one exhaustively iterating through all ($RAND_i$, $RAND_{i+8}$). In this way, for every (K_i , K_{i+8}) we can find every collision-inducing pair of ($RAND_i$, $RAND_{i+8}$). The ($RAND_i$, $RAND_{i+8}$) can then be arranged in an improved challenge schedule, so that the ($RAND_i$, $RAND_{i+8}$) which are most productive taken over all the (K_i , K_{i+8}), are used first.

A further improvement can then be obtained by following this challenge schedule in parallel for all eight (K_i , K_{i+8}). When the first seven values of (K_i , K_{i+8}) have been determined, the attacker can stop querying the SIM, since the last (K_i , K_{i+8}) can then be found by an off-line brute-force computation. With a small extra complication noted in the next section, this method gives the following performance (results from 100,000 keys chosen at random):



Using this technique, the median number of challenges needed to determine a key is around 60,000.

4. Thwarting the WGB attack

The weakness of the WGB attack became evident when performing the exhaustive search needed to improve on it. As noted, the structure of the first two layers of COMP128 is not an evenly distributed random function. Just as some ($RAND_i$, $RAND_{i+8}$) challenges are more productive than others, some values of (K_i , K_{i+8}) are more susceptible to attack than others. These susceptible (K_i , K_{i+8}) have several pairs of ($RAND_i$, $RAND_{i+8}$) challenges which can induce a collision.

Other values of (K_i , K_{i+8}) are less susceptible to attack, and a few values of (K_i , K_{i+8}) have no pairs of ($RAND_i$, $RAND_{i+8}$) which can induce a collision.

Despite the statement that “the birthday paradox guarantees that collisions will occur pretty rapidly” in [1], quoted as gospel in [2], the “birthday paradox” in fact offers no such guarantee. All it offers is a high probability, assuming that the hash function is truly random. Under that assumption, for each (K_i , K_{i+8}) the probability of not obtaining a collision after all the 2^{16} ($RAND_i$, $RAND_{i+8}$) challenges have been used is approximately 3.35×10^{-4}. Since there are 2^{16} possible values of (K_i , K_{i+8}), one would expect that approximately 22 of these would be collision-free after all possible challenges had been issues.

Surprisingly, an exhaustive search shows that there are 769 values of (K_i , K_{i+8}) which are collision-free at layer 1, and for which the WGB attack therefore does not work. These values of (K_i , K_{i+8}) are listed in Appendix A

The original WGB attack works only for keys with none of these collision-free free (K_i , K_{i+8}) components. The improved WGB attack described above also works for keys with only one of these collision-free (K_i , K_{i+8}) components since it will still crack the last (K_i , K_{i+8}) off-line by brute-force, using at most 2^{16} trial hashes.

For those keys with two collision-free free (K_i , K_{i+8}) components we can extend the improved WGB attack. (This is the “small extra complication” noted above.) In this case, when we have used the last ($RAND_i$, $RAND_{i+8}$) challenges in the schedule if only two (K_i , K_{i+8}) components remain uncracked, we can still perform a brute-force off-line attack, using at most $769 \times 769 = 591{,}361$ trial hashes. (If we did not know the 769 collision-free (K_i , K_{i+8}) components, this brute force off-line attack would need to search through up to 2^{32} trail hashes which would certainly inconvenient although not entirely infeasible.) For larger numbers of strong (K_i , K_{i+8}) components, this final brute force attack ceases to be so practical, but less than one in 10,000 keys chosen at random will have more than two such components.

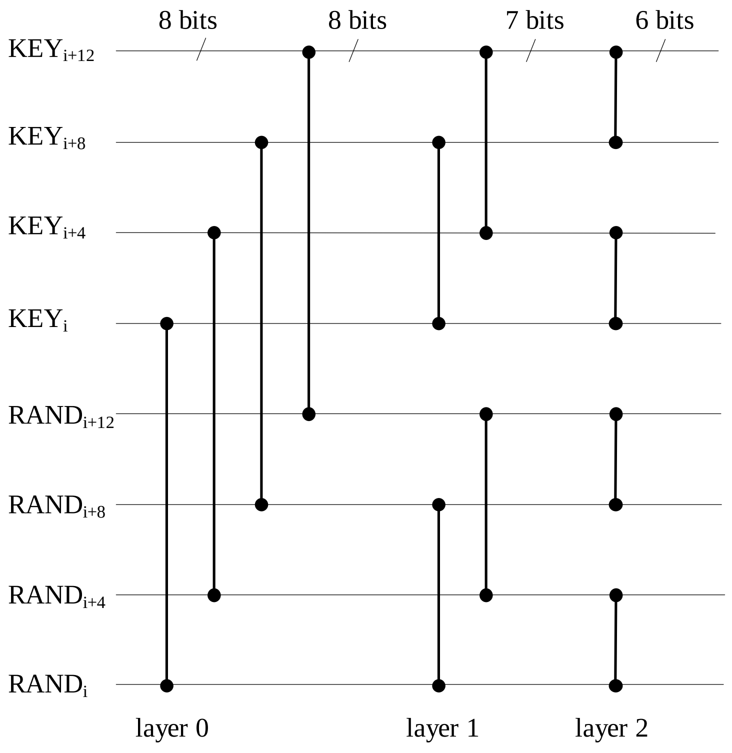
Thus, to generate stronger keys which are fully resistant to the WGB attack, one should select all the (K_i , K_{i+8}) components from the list of strong ones:

```
for i in [0, 1, 2, 3, 4, 5, 6, 7]
        let x        = randomly chosen number from the list in Appendix A
        let $K_i$      = first two hex digits of x
        let $K_{i+8}$  = last two hex digits of x
```

Since there are 769 stronger (K_i , K_{i+8}) components, this means that the stronger keys generated by this method have an effective length of $8 \times \log_2 769 = 76.7$ bits. This is still a reasonable key length, and longer than the 64 bit session key length used for voice encryption with the A5 algorithm.

5. A new attack

The new attack relies on knowing that the key is formed only from the stronger (K_i , K_{i+8}) components, as described above. If this is already know, the attack can proceed, but otherwise this must be established by first eliminating all the weak keys using improved WGB attack. (Which takes at most 150,000 challenges.) The new attack then proceeds to induce collisions at layer 3:

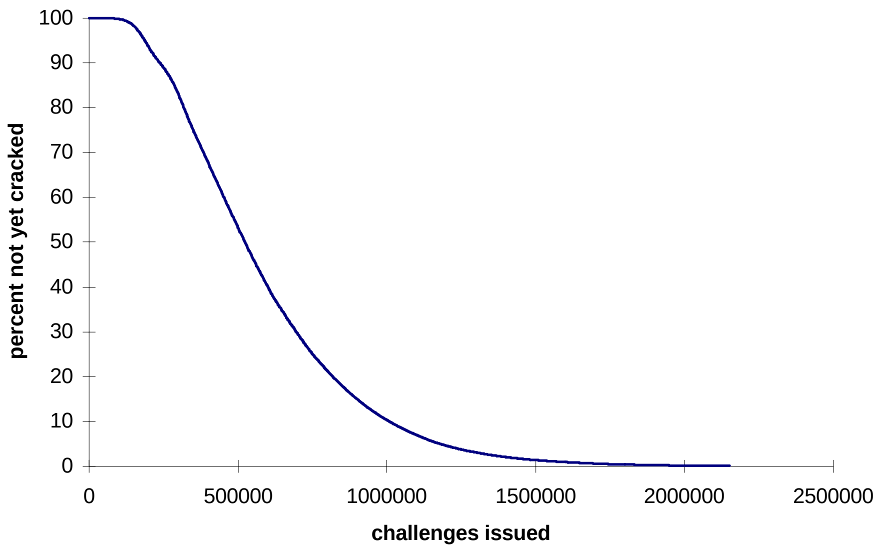


Four (K_i , K_{i+4}, K_{i+8}, K_{i+12}) components need to be found. Each of these is built from two of the 769 stronger (K_i , K_{i+8}) components. In order to find each (K_i , K_{i+4}, K_{i+8}, K_{i+12}) component, 32-bit ($RAND_i$, $RAND_{i+4}$, $RAND_{i+8}$, $RAND_{i+12}$) challenges are issued, in a way similar to the original WGB attack, while keeping the other bytes of RAND constant. When a pair of bit ($RAND_i$, $RAND_{i+4}$, $RAND_{i+8}$, $RAND_{i+12}$) challenges is found which induces a collision, the 769×769 possible (K_i , K_{i+4}, K_{i+8}, K_{i+12}) components are searched until one is found which also gives a collision for the same pair of challenges.

In this case no optimised challenge schedule is available, since the size of the key and challenge space preclude exhaustive search. Nor is it known whether any of the (K_i , K_{i+4}, K_{i+8}, K_{i+12}) components are collision free. Given the size of the layer 2 output (48 bits) versus the RAND input (32 bits) this would seem extremely improbable.

In theory, if COMP128 were a truly random function, to induce collisions at the 48-bit wide output of layer 2 would require around 2×10^7 challenges. Experimentally, we find that the median number of challenges to determine one (K_i , K_{i+4}, K_{i+8}, K_{i+12}) component is actually about 127,000. However, it takes less than four times this number of challenges to determine all four such portions of the whole key. This is because the attacks on each of the four components of the key can proceed in parallel, as in the optimised WGB attack described above, and after three components have been discovered, the fourth can be cracked off-line.

A simulation which randomly re-samples 5290 experimental results for the time to crack a single (K_i , K_{i+4}, K_{i+8}, K_{i+12}) component indicates that the median number of challenges needed to crack a whole strong key would be about 525,000:



6. Conclusions

The attack on COMP128 described by Wagner, Goldberg and Briceno can be improved to give an expected number of challenges before success of 60,000, rather than the 150,000 in the original. However, COMP128 has stronger keys, with an effective key size of 76.7 bits, which are resistant to those attacks. These stronger keys are in turn vulnerable to another attack, but has an expected number of challenges before success of 525,000 plus another 150,000 to establish that the key is stronger if this is not already known.

If any GSM operators are still using COMP128, they could therefore immediately improve the security for new subscribers by generating stronger keys for them. Of course, an even better choice would be to use another hash algorithm without the weaknesses of COMP128.

The obvious moral of the story about COMP128's weakness was that cryptographic algorithms should be subject to greater public review. However, while this is true it glosses over the fact that this was also a protocol failure: if SIMs had been programmed to limit the rate at which challenges could be issued then the original WGB attack would never have been practical. An attack which needs 50,000 challenges to work could have been made infeasible by a rate limit in the protocol. Many GSM phones will not make 50,000 calls in their entire life. A "belt-and-braces" approach is preferable, even if one is convinced that the braces alone are sufficient.

7. References

[1] "GSM Cloning"
Wagner, Goldberg & Briceno
http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.htm

[2] "Reducing the Collision Probability of Alleged Comp128",
Handshuh & Paillier,
in "Smart Card Research and Applications",
LNCS vol. 1820 (2000), pp 380-385.

Appendix A

The 769 strong (K_i, K_{i+8}) are in hexadecimal:

```
000b 003f 005b 006d 0119 01a8 01f8 0293 031e 035a 036b 03ba 03cf 03fc
0452 04ca 0532 05a9 05e0 063b 0657 085c 089f 09ce 0a88 0aa6 0b00 0b7f
0bd2 0c42 0ce8 0d18 0d9d 0e14 0ea4 0ea9 0ed9 0f4a 0f4d 0f6f 0fa5 0fd0
0fe6 1011 1056 10ad 10cf 1110 116b 11c4 11da 11db 1228 123d 12ba 12cb
12e8 140e 14bf 1588 1596 15d6 15e0 1638 1643 1728 173a 180d 1871 18c3
1901 192f 1935 1965 1975 1978 19cd 1a5c 1abb 1b20 1ba4 1bb4 1bc0 1c68
1cd7 1d37 1d5f 1d87 1e03 1e1e 1e20 1f44 1f54 1fca 201b 201e 2057 2069
2079 207b 2137 21c8 2268 22dc 23b5 23cf 2432 2581 25a5 2685 2691 273e
276e 2777 2793 2812 2817 2859 28ac 2a61 2c39 2cd2 2cd7 2d6d 2e7b 2e93
2ee1 2f19 3074 3088 30ba 30f8 31bb 31d4 3205 3224 32e4 3334 3374 3433
34fa 3519 354e 3650 3668 36ae 371d 3721 3751 378c 379e 3816 38f9 392c
3977 39d9 39fa 3a17 3a80 3abe 3b06 3be4 3be7 3beb 3c78 3cd0 3cd3 3d12
3d9d 3df6 3e27 3e5c 3e5d 3f00 3fc9 3fcb 406d 40ea 414e 4184 41a3 41a5
420c 425f 42ba 42df 4316 43e9 441f 4489 44fe 45d1 4651 46a0 48d9 494a
49a7 49ab 49b1 49cb 49f9 4a0f 4a49 4a6f 4a80 4ab3 4ac5 4b4e 4b5b 4b93
4bc7 4cba 4ccc 4d0f 4d4e 4d93 4df4 4e35 4e41 4e4b 4e4d 4e76 4e7d 4e91
4ef6 5036 50c4 5137 5146 5204 5264 528e 53d7 541f 5489 548d 54e5 556b
55be 5610 5665 56ba 56d8 5706 5720 5771 5889 58b7 5928 5983 5a03 5acb
5af1 5b00 5b4b 5bb7 5c08 5c1a 5c3e 5ca6 5cc8 5ccc 5ce1 5ce6 5d3e 5dc4
5e86 5eec 5f1d 5f42 608e 60f9 612a 61da 61f1 61f5 627d 6374 638a 63a0
63f0 6452 64a6 6519 6556 6676 6692 66d0 67a8 67e9 681c 6822 6836 68b4
6920 697b 6988 6a73 6a7c 6a8c 6aa7 6b03 6b11 6b55 6b6e 6bad 6be2 6c7b
6c84 6d00 6d2d 6d40 6df9 6e27 6e6b 6e83 6eb1 6ed2 6f0f 6f4a 6fb0 6fcb
7070 7072 708c 70fc 7118 7157 7270 7291 729a 72ad 736a 73de 73e0 7430
7433 7463 74b3 7519 764e 7666 7679 768f 7727 7739 7780 778a 77b5 77f5
7819 783c 78d8 7920 7976 7983 79a1 7a81 7b20 7b2e 7b69 7b6c 7b9d 7bcc
7bd2 7bf6 7c6a 7cf5 7cfd 7d4e 7d62 7d8d 7ef6 7efe 7f0b 803a 804a 8077
80c6 8125 817a 8181 82a4 82b1 8359 836e 8379 8393 8441 846c 84dd 8526
85d4 865e 86a4 86ab 871d 878d 87bd 87f4 880a 8815 8830 8869 88d0 8944
8954 8958 899b 8a63 8a77 8ac1 8ba0 8baf 8c37 8c6a 8c70 8d54 8d7d 8d87
8e52 8e60 8e97 8ed6 8ede 8f76 90af 90dd 9126 914e 9172 91f2 9266 929e
92b7 9302 9327 932e 934b 934d 9383 93df 94ab 94d9 95b5 95f0 9615 96de
96df 978e 98a0 98df 9a72 9b89 9bff 9cf1 9d0d 9d3d 9d7b 9e37 9e92 9ebc
9f08 9fac 9fbd 9fd4 9fee 9ffc a046 a063 a08b a098 a0b9 a0e7 a179 a1a6
a1e8 a2a2 a341 a3b1 a40e a41b a482 a486 a4c7 a4ce a4d2 a50f a525 a541
a5ab a5e9 a60a a65c a664 a6a1 a749 a76a a7e7 a801 a867 a905 a90e a9ed
aaec aaf6 ab49 ab86 ab94 aba5 abff ac28 ac9f acb0 acc7 ad10 ad6b ad72
ae36 af8b af90 b06f b0ac b149 b16e b182 b1a3 b1d1 b1e4 b1e8 b34a b374
b3e2 b41b b468 b4e0 b523 b577 b595 b5f5 b5f7 b6d3 b758 b75b b792 b7f0
b8e5 b8ec b9a0 ba03 ba12 ba30 ba42 ba4c ba56 bb1a bb31 bbec bbf6 bc9e
bcde bcf3 bcfe bd87 bd9f bdca bdd9 be3a be55 bed9 bf14 bfd4 c01b c0fc
c18a c1eb c318 c3f6 c411 c450 c45d c4d7 c54a c5c5 c680 c6d7 c74b c7a4
c7ac c7f1 c821 c85c c8cd c93f c9ca c9f5 ca04 ca1f cabd cac9 cae0 cb12
cb3f cb49 cb5a cb6f cc4c cc5c cc7b ccde cd19 cdc8 ce09 cea4 ceed cf03
cf10 cf23 d00f d03c d066 d088 d0d6 d145 d1b1 d1ea d20b d22c d26e d27b
d2a4 d33c d3b6 d431 d485 d49f d4bf d5e5 d615 d68e d6d0 d6e6 d71c d72c
d753 d7c4 d7c6 d856 d878 d90e d939 d948 d994 d9bd d9be da11 da61 daf8
db11 dc22 dd84 dd90 de73 de8e de96 debc decc df42 df93 df96 df98 e005
e015 e073 e0b4 e0ca e12e e15c e26b e2b3 e3f9 e432 e43b e4b1 e4e7 e554
e5b8 e5d5 e60f e65c e6d6 e73b e7a0 e7a7 e7e4 e80c e812 e8a1 e8b1 e943
e967 e9a5 ea40 ead1 eafa eb3b ebc1 ec5e ecaa ecb8 ecbb eda9 edce ee9f
eef5 f063 f095 f0b7 f15a f161 f19c f1c7 f291 f3bc f44d f487 f561 f577
f57c f5b5 f5c9 f5ee f5fa f5fb f63d f64e f67b f67e f6aa f6bb f6c3 f7b5
f801 f830 f8da f938 f949 f960 f96d f9e3 fa34 fa39 faea faf5 fafc fbf5
fc03 fc70 fc9f fcc0 fcfa fcfd fd7c fdfc fe44 fe7e febc ff9b ffab
```