Decentralized management of One-Time Pad key material for a group

Jeroen van de Graaf

[1]Departamento de Ciência da Computação, UFMG

jvdg@dcc.ufmg.br

Abstract. *Suppose a group of users share copies of a large file of truly random bits, possibly distributed through portable USB sticks or external hard drives. In this note we present a randomized, distributed key management scheme allowing these users to use this file as a One-Time Pad key, without fear of two users using the same key material twice.*

1. Motivation and problem

Consider the following setting: a small group of people, who meet on some regular basis, collaborate on some project. They wish to protect the texts they themselves produce: emails, chats, some typed documents maybe. Suppose that these people are closely watched by three- and four-letter agencies with lots of expertise in cryptography, like journalists working on classified information. In this situation it makes perfect sense to use the One-Time Pad (OTP). The amount of information to be encrypted is very small relative to current storage media such as SD cards, USB memory sticks or external hard drives, the latter being able to store 1 terabyte. So key transportation is possible through personal meetings or through couriers, personal or commercial (FedEx).

As a concrete example, suppose that these persons share a 1 GB random file which has been reliably distributed to each member by copying it to USB sticks on an air-gapped, and otherwise protected, secure hardware platform. They would like to interchange documents over the internet, encrypting them using the file as an OTP key. We assume this OTP key to be secret towards outsiders, in particular the spy agency.

However, it is well-known that an OTP key should never be used twice, since in that case the OTP loses its security properties, and a statistical analysis allows partial recovery of the plaintexts[2]. So the problem they need to solve is this: How to avoid collision of the key material? How to avoid that different members use the same part of the OTP key to encrypt different documents? How to decide who uses which part of the OTP key file considering these constraints?

One solution is to have a central server controling key distribution, allocating parts of the OTP key to the members. However, this would require the group members to be online, and constitutes a single point of failure. It also constitutes a single point for a spy agency to monitor: even not having access to the OTP key file, they might be able to find out who communicates with whom, and the sizes of the messages sent. Traffic analysis, in other words. So we prefer a solution without a central server.

Another solution is letting a sender choose a start position p at random, and use the bits from that position onward as a OTP key. This p is included as meta data, sent in the clear (for simplicity of exposition–we can do better), to inform the recipient where to find the position of the decrypt key. However, in this case there is a risk that two members choose the same start position, and confidentiality is compromised. For concreteness, suppose that the 1GB key file is divided into 1024 blocks of 1MB each, and that the group members send 1MB messages. From the birthday paradox (see for instance [3, 4] we know that after only $1.17\sqrt{1024} \approx 37$ message we have 50% chance that a least two messages collide and have been compromised. The question addressed in this note is: can we do better? The answer is YES.

2. Towards a better solution

In order to reduce the collision probability, we let a sender choose several start positions, and derive a OTP subkey from each. In other words, the sender chooses N start positions at random, where $N = 16$ (this choice will be discussed later). Again, the random start positions $p_1, \ldots, p_N$ are added as metadata to the encrypted message.

Now let M be a plaintext message of size 1MB, let K be the OTP key of 1GB, and let C be the ciphertext. We divide K in 1024 blocks of size 1MB each, and define K_p as the pth block, to be used as a OTP sub-key. Now randomly choose N different start positions $p_1, \ldots, p_N$ with $p_i \in \{1 \ldots 1024\}$. Then compute the net OTP key as $K^* := K_{p_1} \oplus K_{p_2} \oplus \ldots \oplus K_{p_n}$ and the ciphertext $C := M \oplus K^*$, where $\oplus$ denotes the bitwise xor operation.

Observe that now, for two parties to collide, they would have to choose the *same* subset of N positions in a total of 1024 possibilities. So the space of possibilities is of size $\binom{1024}{N}$, and applying the approximation for the birthday paradox to this example, we obtain $1.17 \cdot \binom{1024}{N}^{1/2}$. For $N = 16$ this gives $1.17 \cdot \binom{1024}{16}^{1/2} \approx 1.17 \cdot (10^{34.79})^{1/2} \approx 2.91 \cdot 10^{17}$. This means that around $2.91 \cdot 10^{17}$ messages would be needed to have a 50% change of a collision occuring. Even though 50% is much higher than desirable, it shows that our approach is promising.

We also address the efficiency of the scheme, analysing what percentage of the available random key material can be used without making compromising security. We claim that, even for reasonably small values of N, the users can essentially use 99% of the random key material, while the probability of having collisions remains neglible.

3. More on the collision probability

The preceding approximation of the collision probability is incomplete, in the sense that it calculates how many message can be sent in order to have a collision with 50% chance. However, a collision is a catastrophic event, and its probability should be kept very low: $\varepsilon = 10^{-10}$ or less.

Let S be the total set size, and k be the number of elements chosen. (For the birthday paradox $S = 365, k = 23$). Let $\varepsilon = p(S, k)$ be the probability of having a collision. A well-known approximation for $\overline{p}(S, k) = 1 - \varepsilon$ based on Taylor series [3] is $\overline{p}(S, k) = e^{-(k(k-1))/2S}$. Approximating $k - 1 \approx k$ and taking logs on both sides, we get $k^2 = 2\ln(\frac{1}{\varepsilon}) \times S$. For the birthday paradox, with $\varepsilon = 1/2$, this gives us $k = \sqrt{2\ln(2)}\sqrt{S} \approx 1.17\sqrt{S}$, the approximation used in the first example using only 1 position, and which is known to give the correct answer, $k = 23$, for $S = 365$.

Now using the numbers of the second example we have $S = \binom{1024}{16} \approx 10^{34}$, and setting $\varepsilon = 10^{-10}$ we obtain $k = \sqrt{2\ln(\frac{1}{1-\varepsilon})}\sqrt{S} = \sqrt{2(10^{-10})}\sqrt{10^{34}} = 1.41 \cdot 10^{12}$. Here we used that $\ln(\frac{1}{1-\varepsilon}) \approx \ln(1+\varepsilon) \approx \varepsilon$ for ε very small. This formula shows that the failure probability $\overline{q}$ and the the size of the possibility space S balance fairly: if we want to fix the number of messages k, but want to reduce ε by a factor 10 (say), we need to multiply S by 10. The value $k = 1.41 \cdot 10^{12}$ is higher than we need, as the discussion in the next section shows.

4. Higher order collisions

The preceding analysis cannot be the complete picture since the limit on the number of message seems to be very high, whereas we know for sure that after 1024 messages of 1MB we *must* have exhausted the 1GB key material. The point is that the analysis above only explores collisions between *two* different messages, whereas much more complicated collisions are conceivable.

Let $p_{r1}, \dots, p_{rA}$ denote the OTP key positions chosen when encrypting message r, where A is the total number of positions (above $A = 1024$). Let us describe these N positions as a 0/1 row vector P, where $\overrightarrow{P}_{ri} = 1$ iff $i \in \{p_{r1}, \dots, p_{rN}\}$. By vertically listing the row vectors, we obtain a $T \times A$ matrix over $\mathbb{F}_2$ called $\mathcal{P}$, after T messages have been sent. As soon as $\mathcal{P}_{T\times A}$ has a linear dependency we have a problem, since it means that there exists a non-empty subset $I \subseteq \{1 \dots T\}$ such that $\bigoplus_{i\in I} C_i = \overrightarrow{0}$, implying that $\bigoplus_{i\in I} M_i = \bigoplus_{i\in I} K_i^*$. So if $\bigoplus_{i\in I} K_i^* = \mathbf{0}$ the adversary knows that $\bigoplus_{i\in I} M_i = \mathbf{0}$. The collision described in the previous section is a special case of this, with $\#I = 2$.

So we must answer the following question: if we have row vectors of size A, and we keep on adding rows, what is the probability that the resulting matrix has a linear dependency after having added T rows? Obviously, T cannot exceed A but the question is how close we can get.

For *random matrices*, where each entry is 0 or 1 with 50% chance, the formula for the expected dimension (rank) of $\mathcal{P}_{T\times A}$ is known ([1], §3.5). However, a downside using this approach is that each row has an expected Hamming weight of 512, so we need to do $N = 512$ file seeks and xors on average. We would like to reduce N to a much smaller value, though very small values, like $N = 1, 2$ or 3, lead to collisions. But how about $N = 8$ or $N = 12$? How large can T be before we have a linear dependency?

Though similar problems have been studied in the context of low-density parity check codes, we have not yet been able to find a relevant reference or derive an exact formula. Instead we did some computer simulations: we randomly generate a new row vector of Hamming weight N, we check if the new vector is independent and use straightforward Gaussian elimination over $\mathbb{F}_2$ to eliminate all zeroes below the diagonal. The results of these simulations for $A = 1024$ over 100 experiments are as follows:

N	1	2	3	4	5	6	7	8	10	12
avg	39.6	459.2	937.4	996.8	1014.9	1018.2	1021.4	1021.1	1021.3	1021.7
min	3	110	906	984	1008	1010	1015	1013	1014	1015
max	110	639	961	1013	1023	1023	1024	1023	1023	1023

These results show that for $A = 1024$ and a value for N as low as 8 the dimensionality of the matrix essentially behaves as a random matrix, meaning that the probability of linear dependencies is extremely low until T approaches A. Given the data it seems safe to conjecture that for $N \geq 8$ and $T = 1000$, the probability that a dependency occurs is negligible. Finding an analytical formula for this probability is one of our main research priorities; otherwise more efficient and more simulations are needed.

5. Some observations and conclusion

(1) The random OTP file should be stored in encrypted form, so that if it falls into the wrong hands not everything is compromised. Security then degrades to cryptographic security. Using counter mode with a strong symmetric cipher is a good option because we want random access to the file. (2) The positions used could be included using symmetric encryption to make the adversary's task harder. (3) One should include message authentication such as an adaptation of Galois Counter Mode, where the OTP blocks are interpreted as if they were the blocks generated by the counter mode. (4) To protect against leakage of individual bits one could apply unconditional all-or-nothing transforms [5] before encryption. (5) Choosing the right block size (1MB in the example) depends on the application. Maybe more than one size is useful.

Bruce Schneier once discarded the OTP as follows: "What a one-time pad system does is take a difficult message security problem – that's why you need encryption in the first place – and turn it into a just-as-difficult key distribution problem. It's a 'solution' that doesn't scale well, doesn't lend itself to mass-market distribution, is singularly ill-suited to computer networks, and just plain doesn't work." I see his point, but there are situations in which using the OTP makes perfect sense. People use email and WhatsApp often with people they meet on a frequent basis. Why can't they use the OTP here? We should work harder to make this a viable option.

Note added in proof: One of the referees pointed out that the analysis presented here does not include so-called *meet-in-the-middle attacks*, as presented in [6]. These attacks imply that the actual security level of this scheme is much lower than what is claimed, so assessing their impact is currently a top research priority.

References

[1] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST Special Publication 800-22 Revision 1a, April 2010.

[2] C. Shannon, *Communication Theory of Secrecy Systems*. Bell System Technical Journal 28 (4): 656–715..

[3] P. Halmos. *I Want to Be a Mathematician.* Springer-Verlag. ISBN 978-0387960784.

[4] Anonymous. *Birthday Problem* `http://en.wikipedia.org/wiki/Birthday_problem`

[5] Stinson, D. *Something About All or Nothing (Transforms).* Designs, Codes and Cryptography, Volume 22 Issue 2, March 2001, Pages 133–138

[6] K. Nishimura and M. Sibuya, *Probability To Meet in the Middle*. J. Cryptology, vol 2 (1), 1990.