DARK Lite: A fast DARK implementation

Michael Brand

November 23, 2023

Executive Summary

We describe a new algorithm from the FinTracer DARK family of algorithms, demonstrating how FinTracer DARK can be sped up to speeds comparable with those of standard FinTracer.

1 Introduction

FinTracer DARK is a variation on the FinTracer algorithm which presently composes the backbone of the FinTracer algorithm suite. Its advantage over standard FinTracer is that it can be used even when no two-sided-viewable propagation graph can be defined, but this comes at the expense of additional computation and communication.

Through successive documents [4], the privacy guarantees of DARK were improved, but this was done at the expense of speed and communication sizes. Ultimately, bounds were proved regarding how privacy-preserving a DARK algorithm can be, and these bounds showed that the "Extreme DARK" algorithm of [4] is optimal in this respect.

In terms of its core functionality, however, what FinTracer DARK can do is strictly a superset of what can be done with standard FinTracer. In fact, one can think of it as a possible replacement for FinTracer. However, this comes at a significant cost in performance. In [4], the state of the art is summarised as follows:

> "[T]here is a sense in which Extreme DARK can be thought of as (in addition to everything else) also a replacement for FinTracer. In such a perspective, it makes sense to compare the two algorithms in terms of their computational weights.
>
> Extreme DARK is an $(R-1)$-times slower version of the algorithm of DARKer, requiring both $(R-1)$-times the computation and $(R-1)$-times the communication (with all other factors being negligible) [where R is the number of peer nodes used].
>
> Even omitting the $R-1$ factor, DARK requires more communication than FinTracer by requiring signatures to be transmitted instead of just tag values. (However, signatures only need to be exchanged once, when setting up the graph, so this is only a small factor heavier than the synchronisation step of FinTracer.) Also, tags in Extreme DARK are doubly-encrypted, meaning that one may incur a size increase due to the encryption. (If one uses a second layer of ElGamal, tag size increases by 50%. Note, however, that for the purpose of Extreme DARK the second layer only needs to be a public key encryption system, not necessarily a semi-homomorphic one.)
>
> [...]
>
> The greatest cost difference between the algorithms is, however, not in the communication sizes they require but in their computational costs. Extreme DARK requires each communicated tag to be encrypted and decrypted at every propagation step. That is, unfortunately, part of what provides the algorithm's security. While encryption can be performed quickly, decryption is roughly 100 times slower than refreshing, causing in total a slow-down in the algorithm by a factor of $100(R-1)$ compared with the basic FinTracer."

The costs of encryption, it is further noted in [4], can be avoided by using our stockpile of pre-encrypted zeroes, much as is done in "refresh" operations in standard FinTracer, but such a trick does not work for decryption, which slows down the entire algorithm.

These points, unfortunately, continue to hold, with the following additional remarks. As noted, the speeds quoted refer to the use of ElGamal homomorphic encryption under the Ed25519 group. For the particular needs of Extreme DARK, however, *any* public key algorithm would have worked. One may conclude from this that by not choosing ElGamal, but rather the fastest public-key cryptosystem available, one would have gained more speed. The reality is, however, that Ed25519 is very fast. It is hundreds of times faster than RSA, and comparable in its encryption/decryption speeds to a symmetric encryption algorithm, like AES[1].

In this document we show how we can speed up FinTracer DARK to speeds comparable with those of standard FinTracer. To do this, we trade off some of the algorithm's privacy guarantees. Specifically, the new algorithm makes the following concession in terms of privacy: all peer nodes discover regarding all other peer nodes the total number of DARK addresses they write to and the total number of DARK addresses they read from. As we have done in other cases, it is possible to add to this value nonnegative amounts of noise, so as to obfuscate it using differential privacy. This document does not delve into the exact amounts of such noise that would be required for effective differential privacy. Interested readers can refer to [5, 6].

Whether or not such a concession is legitimate is a policy decision. It should be considered in each case separately whether divulging to all REs the approximate total number of DARK addresses used for read and for write by each RE is appropriate.

The importance of a fast DARK algorithm is clear for the overall practicality of the use of DARK technology. However, recently, various algorithmic improvements to the FinTracer algorithm suite have been introduced (most particularly the LineTracer algorithm but more generally algorithms that work under the SoNaR model [7]) that make the use of DARK propagation not only a method to match DARK-style typologies, but also an essential ingredient in matching standard FinTracer typologies, meaning that it is important across the board. Thus, this algorithm addresses an important and central issue in the current suite of FinTracer algorithms.

2 DARK Lite using a One-Time Pad

The idea for the DARK Lite algorithm is two-fold. First, we modify the algorithm so that it no longer requires public key encryption, and can work equally well with symmetric key encryption. Second, we convert the algorithm to work with the fastest (secure) cryptosystem we can.

In Section 3, we demonstrate a more practical approach to attaining these goals, but we begin in this section by demonstrating how DARK Lite can be implemented using the absolute fastest (secure) system around, assuming one has the ability to perform pre-work. The fastest such system is, namely, the use of a One-Time Pad (OTP).

As was done in Extreme DARK, DARK Lite handles tag propagation from each peer node R_1 to each peer node R_2 separately. We assume, for the purpose of this document, that R_1 and R_2 are distinct, because propagation between a node and itself can be done trivially, using the same mechanism as for standard FinTracer.

Consider, therefore, a specific choice of distinct R_1 and R_2, and assume that for an end-to-end implementation of DARK Lite we apply the following, in parallel, on all such pairs.

Because in this section we consider an OTP-based solution, we must also assume that such an OTP was coordinated between R_1 and R_2 ahead of time. One way to do this is for R_1 to send to R_2, in advance, a large cache of random bit blocks, each block 128 bytes in size.

Note 2.1. To speed up the algorithm even further, R_1 may also prepare in advance a second cache of random bits to be used for generating the random permutation σ that it uses in this protocol. There are very quick ways to generate a random permutation, if one has a source of enough randomness.

[1] At least, assuming one is not using a chipset where AES is implemented as a basic assembly instruction.

This second cache, however, must remain secret, private to R_1.

Once the necessary random bits have been prepared and coordinated ahead of time, the algorithm for a specific (R_1, R_2) pair proceeds as follows.

First, the DARK addresses to be written to / read from are coordinated. This only needs to be done once when setting up a new propagation graph. When propagating multiple times along the same graph, the same set of addresses can be reused. Furthermore, all parties need to coordinate two values, s and r, where s is an upper bound to the number of addresses being sent to, and r is an upper bound to the number of addresses being read from. If there is no need to hide the number of addresses being written to / read from, then s and r can simply equal these values, respectively. Otherwise, they are computed by R_1 and R_2, respectively, by adding differential privacy noise.

See [5, 6] for a discussion of how to compute noise amounts.

Note 2.2. If R_1 sends a set of addresses (under different hashes) to multiple other peer nodes, it should use the same s value each time. Similarly, if R_2 reads from the same set of addresses (under different hashes) in order to receive data from multiple peer nodes, it should use the same r value each time.

This one-time setup of addresses and parameters is described in Algorithm 1.

Algorithm 1 DARK Lite setup

1: R_1 computes the (hashed) DARK addresses it wishes to write to. Let s' be the number of these addresses.
2: R_2 computes the (hashed) DARK addresses it wishes to read from. Let r' be the number of these addresses.
3: **if** differential privacy is used **then**
4: R_1 sets $s \leftarrow s'$+nonnegative differential noise.
5: R_2 sets $r \leftarrow r'$+nonnegative differential noise.
6: ▷ See [5, 6] for how to compute noise amounts.
7: **else**
8: R_1 sets $s \leftarrow s'$.
9: R_2 sets $r \leftarrow r'$.
10: **end if**
11: R_1 sends to AUSTRAC A_w, an array of s hashes, of which s' values correspond to hashes of addresses R_1 wishes to write to, and the rest are fake/random ones.
12: R_2 sends to AUSTRAC A_r, an array of r hashes, of which r' values correspond to hashes of addresses R_2 wishes to read from, and the rest are fake/random ones.
13: ▷ The order of values in A_w and A_r should be non-revealing. Particularly, it should not reveal which are true addresses and which not. For example, hashes may be sorted, or may be ordered randomly. See [4] for how to choose fake/random hashes.
14: AUSTRAC transmits the value of s to R_2 and the value of r to R_1.

After the addresses are set up, we propagate tag values via Algorithm 2. This algorithm uses an array of blocks, each block holding 128 random bytes, $C[1], \ldots, C[s+r]$. Once s and r have been coordinated, such an array can be produced simply by removing $s+r$ elements from the previously-coordinated cache. The rest of the algorithm is then as follows.

Note 2.3. Algorithm 2 only replaces the tag propagation bits of the Extreme DARK algorithm. It is still the case that R_1 will need to sum up all tag values it wishes to send to a single address before it sends any value to AUSTRAC, and to refresh the result before sending it to AUSTRAC, and it is still the case that R_2 receives through AUSTRAC results from all other REs separately and will need to sum those together in order to reach the final result. See [4] for the remaining portions. For efficiency, the process by which R_1 refreshes its tag values is assumed to utilise a precomputed stockpile of zeroes, as per the usual FinTracer process. In Algorithm 2, on Line 4, R_1 accesses this same stockpile.

Algorithm 2 DARK Lite with OTP

R_1 creates a secret random permutation σ over the numbers $1, \ldots, s+r$.
R_1 sends to AUSTRAC $S = [(\sigma[i], t[A_w[i]] \oplus C[\sigma[i]])$ for $i \in 1, \ldots, s]$, where $t[x]$ is the tag value for a true address x and can be arbitrary otherwise. ▷ "$\oplus$" indicates xor.
    ▷ In [4] it is recommended to use random values for fake addresses, but in fact arbitrary values are as good.
R_1 allots Z, an array of r fresh zeroes, from its encrypted zero stockpile. ▷ See Note 2.3.
R_1 sends to AUSTRAC $T = [(\sigma[s+i], Z[i] \oplus C[\sigma[s+i]])$ for $i \in 1, \ldots, r]$.
for $j \in 1, \ldots, r$ **do**
    if $\exists i, A_r[j] = A_w[i]$ **then**
        AUSTRAC sets $T[j] \leftarrow S[i]$.
    end if
end for
AUSTRAC sends T to R_2.
for $j \in 1, \ldots, r$ **do**
    R_2 computes the resulting tag value for address $A_r[j]$ as $T[j]$.second$\oplus C[T[j]$.first], ignoring all fake addresses.
end for
    ▷ Following this algorithm, the values $C[1], \ldots, C[s+r]$ are discarded from the cache, and should never be reused.

3 DARK Lite using a stream cipher

In total, the technique above

1. Replaces a public key cipher with a block cipher by R_1 pre-preparing an encrypted stock of zeroes which it then shares with AUSTRAC.

2. Replaces a block cipher with an OTP by pre-sending the OTP and communicating via a code-book.

In the final step in this metamorphosis, we simply replace the OTP with a stream cipher.

Stream ciphers have a major advantage compared to OTPs in that they do not need to be communicated in advance (only their keys do), and they have a major advantage compared to block cipher (possibly barring special AES-supporting hardware) because they are much faster. In [1], as an old example from 2008, using dedicated hardware, the stream cipher `Grain128x32` was measured to generate bits at 14.5Gbps, whereas the best result for AES was 311Mbps. The modern version of this algorithm, from 2019, `Grain-128AEADv2` [2], has been clocked [3] (synthetically) at over 33Gbps, in a suitably optimised hardware implementation. Without special optimisations, [2] measures `Grain-128AEADv2` (synthetically) as attaining 10.6Gbps by working on 32 processors, each working at 662Mhz. In short, stream ciphers are fast.

Utilising a stream cipher to generate the OTP, the full algorithm becomes as described in Algorithm 3.

Algorithm 3 DARK Lite with a stream cipher

if Using a set of DARK addresses for the first time **then**
    Run Algorithm 1.
end if
R_1 and R_2 coordinate a random stream cipher key, k.
    ▷ R_1 may simply choose this, and send to R_2.
R_1 and R_2 generate $C[1], \ldots, C[s+r]$ using the stream cipher, with k as their mutual key.
Run Algorithm 2, using the $C[1], \ldots, C[s+r]$ values computed.

In total, the key k is used to hide tag values from AUSTRAC, the hash keys hide from AUSTRAC the identities of the addresses themselves, the permutation σ hides from R_2 which DARK addresses were written to and which not, and the ElGamal key hides the tag values from R_1 and R_2.

Together, these provide the cryptographic guarantees of the algorithm.

4 Efficiency analysis

DARK Lite is an efficient implementation of Extreme DARK. Like Extreme DARK, it has a built-in $R-1=3$ slowdown compared with standard FinTracer or even with FinTracer DARK, because each tag needs to be propagated $R-1$ times: each R_1 does not know, in this algorithm, which other peers its messages will be delivered to, so it needs to deliver them to each, separately.

Considering, however, a single (R_1, R_2) pair, and ignoring one-time setup and any negligible portion (such as the generation of σ, the coordination of C, or the xor operations), the dominant differences between DARK Lite and standard FinTracer are not the computational costs at all, but rather the communication costs: whereas FinTracer propagates its tags in a single hop, DARK Lite must do so in two hops, one from R_1 to AUSTRAC and one from AUSTRAC to R_2.

We conjecture, therefore, that realistic cost comparisons between DARK Lite and standard FinTracer will show DARK Lite to be only approximately $2(R-1)=6$ times slower than standard FinTracer, making it a practical algorithm and even a realistic alternative for standard FinTracer where direct RE-to-RE communications are not allowed.

This, as stated, does not include one-time setup costs, which may still be high because they involve the computation of a hash per destination address.

5 Bells and whistles

5.1 Decentralised DARK

This document addresses the problem that DARK requires much more computation than standard FinTracer. A different problem, however, is that DARK requires much more *communication through the AUSTRAC node*. This second aspect of the problem is the topic of this section.

In general, DARK Lite does not alleviate the problem of communication through the AUSTRAC node. However, an important observation is that no information held by AUSTRAC is used for DARK propagation, and there is no intentional knowledge gained by AUSTRAC. Thus, even more so than in other algorithms in the FinTracer suite, in DARK the role of the AUSTRAC node is that of a trusted third party.

As such, it is certainly possible, technically, to offload this task on other "trusted" third parties. In fact, it is possible to split the load between multiple nodes, e.g. by having a different "trusted third party" handle the interaction between each (R_1, R_2) pair. Such trusted third parties can be multiple nodes managed by AUSTRAC, they can be managed by other organisations, and in theory they can even be handled by the REs themselves, as long as the party acting as postmaster between R_1 and R_2 is neither R_1 nor R_2.

The problem with delegating the task of postmaster is only that the postmaster receives incidental information, which we need to trust the postmaster not to use. The worst usage is to collude with either the sending or the receiving RE, in which case the full address set used by the other party is easily revealed. (The hashed addresses are directly revealed. Revealing the full addresses merely requires enumerating over potential values and checking their hash results.)

A non-colluding postmaster is privy to:

1. The approximate number of sending addresses,
2. The approximate number of receiving addresses, and
3. The approximate number of matching addresses.

Note 5.1. As described here, it may seem that the exact number of matching addresses is revealed. This topic was discussed in [4]. In short, it is possible for R_1 to create fake addresses in A_w in a way that is coordinated with R_2, so that R_2 can insert the same fake addresses into A_r. In [4], the number of fake addresses required of each type is calculated.

Beyond differential privacy, such matched fake addresses can also be used to verify that the postmaster is doing its job properly.

Thus, with collusion a postmaster may discover much private information, and without collusion the postmaster is privy to a small amount of information that may still be, potentially, commercially valuable. The trusted third party therefore needs to be chosen carefully.

Nevertheless, at least on a technological level there is no problem to decentralise the DARK algorithm. This is true for all DARK variants, not just for DARK Lite.

5.2 DARK Extra Lite

As a final remark, we remind the reader that it is possible to speed the algorithm up even further.

DARK Lite was designed as a sped-up version of Extreme DARK, optimising over Extreme DARK's computations and obviating the need to enlarge communication volumes. Thus, DARK Lite's performance reflects the core of Extreme DARK: $R-1$ passes of propagation steps, each divided into 2, making for a slowdown of $2(R-1)=6$ compared to the one-step propagation that standard FinTracer can do.

However, a user more conscious of performance can choose to use exactly the same kind of optimisation on the original FinTracer DARK, rather than on Extreme DARK. This algorithm, which we'll dub "DARK Extra Lite", works at only about twice the work (computation and communication) of standard FinTracer, making it a highly practical alternative.

The cost is in the fact that the postmaster, AUSTRAC, is privy to more general statistics than in Extreme DARK (255, as opposed to 20 such statistics, in our $R=4$ scenario), and that significantly more differential privacy noise is required in order to obfuscate the value of these. The full calculation is exactly as for the original DARK variants.

References

[1] Good, T., Benaissa, M. (2008). Hardware performance of eStream phase-III stream cipher candidates. In *Proc. of Workshop on the State of the Art of Stream Ciphers (SACS'08)*. (Available at https://www.ecrypt.eu.org/stream/docs/hardware.pdf)

[2] Hell, M., Johansson, T., Maximov, A., Meier, W., Sönnerup, J., Yoshida, H. (2019). `Grain-128AEADv2-A` lightweight AEAD stream cipher.

[3] Sönnerup, J., Hell, M., Sönnerup, M., Khattar, R. (2019). Efficient hardware implementations of `Grain-128AEAD`. In *International Conference on Cryptology in India*, 495–513. Springer, Cham.

[4] Brand, M. (2021), Extreme DARK: A more secure DARK variant. (Available as: `extreme.pdf`)

[5] Brand, M. (2022), *The Complete, Authorised and Definitive FinTracer Compendium (Volume 1, Part 1, 3rd Revision [a.k.a. "Purgles" edition]): A work in progress.* (Available as: `FT_combined.pdf`)

[6] Brand, M. (2022), Some incremental progress in FinTracer technology. (Available as: `small_updates.pdf`)

[7] Brand, M. (2022), Fast privacy-preserving edge-finding, SoNaR, and their uses in FinTracer pair problems. (Available as: `edge_finding.pdf`)