Diffie–Hellman key exchange from the intractability of tensor equivalence

Robert Hines
alphanumericnonsense@gmail.com

October 20, 2024

Abstract

Given an R-module M, public $m_{pub} \in M$, and secret commuting elements $\alpha, \beta \in R$, one could hope that exchanging αm_{pub} and βm_{pub} leads to a secure shared secret $m_{sh} = \alpha\beta m_{pub} = \beta\alpha m_{pub}$. We instantiate this idea with $R = (M_n(S))^d$, $M = (S^n)^{\otimes d}$, for a finite ring S, with a Python prototype available for experimentation.

1 Introduction

1.1 R-module DH

We have an R-module M, with public m_{pub}, exchange $m_A := \alpha m_{pub}$, $m_B := \beta m_{pub}$ for random, secret, commuting $\alpha, \beta \in R$, and derive a shared key from $m_{sh} := \alpha\beta m_{pub} = \beta\alpha m_{pub}$. The shared secret m_{sh} should be unpredictable from the triple $(m_{pub}, m_A, m_B) \in M^3$. At a minimum, the action of the ring should be one-way in some sense.

We could restrict to the group of units in R, i.e. $\alpha, \beta \in R^\times$, and work with a "lossless" group action.

1.2 Linear action on tensors

We choose to work over a specific ring R and R-module M. described below.

Let S be a finite ring, d, n, positive integers, and $R = M_n(S)^d$ the d-fold product of $n \times n$ matrices over S. Let M be the R-module $(S^n)^{\otimes d}$, the d-fold tensor product of the free S-module of rank n. R acts diagonally on M, say from the left:

$$L = (L_1, \ldots, L_d) \in R, \quad L \cdot (e_{i_1} \otimes \cdots e_{i_d}) = (L_1 e_{i_1}) \otimes \cdots \otimes (L_d e_{i_d}),$$

extended linearly, where $\{e_1, \ldots, e_n\}$ is a basis for S^n. One could also let $n = n_i$ vary with the index $1 \le i \le d$, but we fix n for simplicity.

For concreteness and ease of sampling, we can take $S = \mathbb{Z}/2^\kappa\mathbb{Z}$ or $S = \mathbb{F}_{2^\kappa}$. As noted earlier, we could restrict to an action of $G = GL_n(S)^d = R^\times$ at the cost of rejecting sampling for invertible endomorphisms.

2 Cube Diffie-Hellman

Let $T \in M$ be a fixed public tensor. Alice and Bob randomly choose $A, B \in R$ that commute. Alice sends Bob $T_A = A \cdot T$ and Bob sends Alice $T_B = B \cdot T$. Their shared key is derived from

$$T_{AB} = T_{BA} = A \cdot T_B = B \cdot T_A.$$

For instance, Alice could act on the first half of the coordinates and Bob act on the second half:

$$T_A = (A_1, \ldots, A_{d/2}, I_n, \ldots I_n) \cdot T, \quad T_B = (I_n, \ldots I_n, B_1, \ldots, B_{d/2},) \cdot T.$$

The public tensor T and secret A, B, can be pseudo-randomly derived from seed, e.g. SHAKE them out.

3 Security and practicality

3.1 Is the action easy to compute?

If $T \in M$ and $L \in R$, then to compute T_L, one has to go through all n^d basis elements $e_{i_1} \otimes \cdots \otimes e_{i_d}$, apply the coordinate maps $L_j e_{i_j}$, and simplify. This grows exponentially in d, polynomially in n, and linearly in κ. However, the linear algebra is highly parallelizable.

3.2 Is the action hard to invert?

Problem: Given T and $L \cdot T = T_L$, find L. When $d = 2$, this is easy; just (pseudo)invert T. For larger d, this problem hopefully becomes difficult, a version of the "tensor isomorphism problem." See the series of papers starting with [1] for exploration of the computational complexity of tensor isomorphism and [2] for attacks on tensor isomorphism (in particular against the signature scheme ALTEQ).

3.3 What leaks?

How much information does the triple (T, T_A, T_B) leak, or, for static T, what does a sequence $(T, T_{A^{(i)}}, T_{B^{(i)}})_i$ leak? Does knowing that half of the dimensions haven't been acted on,

$$e_{i_1} \otimes \cdots \otimes e_{i_d} \mapsto \left(L_1 e_{i_1} \otimes \cdots \otimes L_{d/2} e_{i_{d/2}}\right) \otimes \left(e_{i_{d/2+1}} \otimes \cdots \otimes e_{i_d}\right),$$

give enough information to start an algorithm to find L from (T, T_L)?

3.4 Parameters

Suppose we work with $|S| = 2^\kappa$, e.g. $S = \mathbb{Z}/2^\kappa\mathbb{Z}$ or $\mathbb{F}_{2^\kappa}$, and that we want 128 bits from each of Alice and Bob for example minimal parameters, i.e. $128 \leq \kappa n^2 d/2$. We'll also take $d \geq 6$ so that Alice and Bob each act in at least three dimensions. Example parameters include:

1. $n = 2$, $d = 6$, $\kappa = 11$ (large κ),
2. $n = 2$, $d = 64$, $\kappa = 1$ (large d),
3. $n = 7$, $d = 6$, $\kappa = 1$ (large n).

The size of all exchanged data (say T, T_A, T_B) is $3\kappa n^d$, so only the first of the parameters above seems reasonable.

4 Prototype implementation

We provide a prototype implementation[1] for $S = \mathbb{Z}/(2^\kappa)$ and $\kappa | 8$ for convenience. Here's an example run with parameters slightly larger than the first above ($n = 2$, $d = 6$, $\kappa = 16$):

```
$ python3 cube_dh.py
parameters : n = 2, d = 6, kappa = 16

T (public): [[[[[[ 6999 48507] [ 106 39397]] [[29622 45172] [56903 37941]]] [[[ 3752 35102] [25117 37227]] [[20587 39370] [46213 50416]]]] [[[[18796 54988] [34877 34765]] [[37716 35593] [35536 38122]]] [[[21094 30389] [44962 2818]] [[ 6430 30411] [10122 7344]]]]] [[[[[33165 3259] [11937 33802]] [[33372 21321] [13153 12473]]] [[[55809 45830] [11908 10156]] [[13979 30612] [ 5325 53250]]]] [[[[ 6731 22545] [36523 64389]] [[45170 5212] [64283 14850]]] [[[58554 14635] [28665 39356]] [[57957 30254] [36067 26898]]]]]]
```

[1] https://github.com/alphanumericnonsense/cube-dh

A (secret): [[[62316 29062] [51306 52531]] [[56638 32529] [62674 59095]] [[53965 5762] [52113 10028]]]

B (secret): [[[26958 43055] [19425 64211]] [[42302 57624] [20639 65054]] [[17535 24987] [34565 26000]]]

T_A (public): [[[[[31931 18135] [51283 41378]] [[172 55887] [48929 26851]]] [[[29187 29928] [60962 63313]] [[57256 59011] [58333 9313]]]] [[[[26156 25462] [24714 42828]] [[24436 7670] [35444 30854]]] [[[21394 38181] [63467 64553]] [[6714 2534] [2450 17702]]]]] [[[[[16809 42597] [10565 6507]] [[59884 61434] [36809 36070]]] [[[31588 34528] [40368 49136]] [[55854 25039] [49871 29333]]]] [[[[59859 37883] [23395 20289]] [[49480 47164] [42661 28536]]] [[[20933 22161] [19133 51433]] [[56434 45894] [12155 60274]]]]]]

T_B (public): [[[[[39056 5867] [46087 65182]] [[47803 34197] [9307 7785]]] [[[10175 61870] [3371 53399]] [[6814 23807] [62834 54026]]]] [[[[5196 43083] [2914 60839]] [[17951 63369] [31863 61630]]] [[[13995 14143] [48028 22749]] [[64863 47318] [2548 6969]]]]] [[[[[1860 24716] [45627 29721]] [[38904 360] [3608 10303]]] [[[31362 3577] [8338 13052]] [[17381 17041] [36399 55067]]]] [[[[5498 64687] [29962 33328]] [[21063 48695] [57375 34706]]] [[[15082 53864] [29338 45040]] [[34412 27490] [46026 64591]]]]]]

T_AB = T_BA (secret): [[[[[[4754 44508] [15227 19277]] [[776 9802] [36008 9083]]] [[[5281 25364] [23473 1613]] [[21036 57259] [52389 9664]]]] [[[[37778 63246] [42066 40550]] [[64190 48704] [50744 39264]]] [[[59739 8696] [56584 31976]] [[64996 40669] [26227 41699]]]]] [[[[[63286 59975] [52734 12706]] [[10111 1323] [51307 7716]]] [[[63966 8437] [62973 43819]] [[19073 61677] [27913 56225]]]] [[[[31114 33711] [44180 24024]] [[6335 19455] [3475 24598]]] [[[9448 8295] [37250 45066]] [[48431 24161] [28353 23324]]]]]]

SUCCESS (0.0031592845916748047 seconds)

key (secret) = d3126ad01c529b433b23553a401a3b5f047d0c4e284743f8fc6ee24635cbc948

References

[1] J. Grochow and Y. Qiao, "On the complexity of isomorphism problems for tensors, groups, and polynomials i: Tensor isomorphism-completeness," *SIAM Journal on Computing*, vol. 52, no. 2, pp. 568–617, 2023, doi: 10.1137/21M1441110.

[2] L. Ran and S. Samardjiska, "Rare structures in tensor graphs - bermuda triangles for cryptosystems based on the tensor isomorphism problem." Cryptology ePrint Archive, Paper 2024/1396, 2024.Available: https://eprint.iacr.org/2024/1396