# A Note on Non-Interactive Key Exchange from Code Equivalence

Lindsey Knowles, Edoardo Persichetti, Tovohery Randrianarisoa and Paolo Santini

December 16, 2021

## Attack Description

A recent paper by Zhang and Zhang [1] claims to construct the first code-based non-interactive key exchange protocol, using a modified version of the Code Equivalence problem. We explain why this approach is flawed, and consequently debunk this claim. A simple Magma script confirms our results.

The authors propose to split the action of the monomial matrix into two parts, where only the even- or odd-numbered columns are affected. To do this, they define the sets $\mathrm{EM}_n(q)$ and $\mathrm{OM}_n(q)$, which are the subgroups of $\mathrm{Mono}_n(q)$ containing matrices that have a 1 in the odd-numbered and even-numbered elements of the diagonal, respectively. The set $\mathrm{sub}-M_n(q)$ is then defined as their union, and leads to the following problem.

**Problem 1 (sub-CLE Problem)** *Given two codes $\mathcal{C}$ and $\mathcal{C}'$ with generator matrices $\boldsymbol{G}, \boldsymbol{G}' \in \mathbb{F}_q^{k\times n}$ respectively, the sub-CLE$(n,k,q)$ problem asks for a pair of matrices $\boldsymbol{S} \in \mathrm{GL}_k(q), \boldsymbol{Q} \in$ sub-$M_n(q)$ such that $\boldsymbol{G}' = \boldsymbol{SGQ}$.*

Although the problem is well-defined, it is easy to see that it cannot provide security. Consider for instance the key exchange protocol described in Section 3.1, and, without loss of generality, let us restrict our attention to Alice's actions (the case of Bob is entirely equivalent). In the scheme, Alice transmits a code of the form $\boldsymbol{G}_a = \boldsymbol{S}_a\boldsymbol{G}\boldsymbol{Q}_a$, where $\boldsymbol{G}$ is public, and Alice's private information is $\boldsymbol{S}_a \in \mathrm{GL}_k(q)$ and $\boldsymbol{Q}_a \in \mathrm{OM}_n(q)$, i.e. $\boldsymbol{Q}_a$ only affects the odd-numbered columns of $\boldsymbol{G}$. Assume for simplicity that the code parameters are such that $k = n/2$. Then, one can generate a new instance $\hat{\boldsymbol{G}}_a = \boldsymbol{S}_a\hat{\boldsymbol{G}}$, where $\hat{\boldsymbol{G}}_a$ and $\hat{\boldsymbol{G}}$ are obtained by removing all the odd-numbered columns from $\boldsymbol{G}_a$ and $\boldsymbol{G}$, respectively. This is because the monomial transformation does not affect the even-numbered columns. Now, since these are square matrices, solving the associated system of equations is immediate, yielding $\boldsymbol{S}_a$ with overwhelming probability. Once $\boldsymbol{S}_a$ is known, finding $\boldsymbol{Q}_a$ from the original equation becomes trivial.

Note that this attack also works if $k < n/2$, since the system of equations is overdetermined and, with overwhelming probability, will yield a single solution.

Now, if $k > n/2$, it may be that the system allows for multiple solutions, and finding the correct one can be made very expensive by choosing parameters to this extent. To be sure, the authors do mention the above attack (or something similar to it) in Section 3.2.3, and suggest to choose[1] $k > n/2$ to avoid it. In fact, with this parameter choice, the system has a number of free variables equal to $(2k-n)k/2 > 0$. However, the authors crucially overlook the fact that the same attack can be performed on the dual of the specified codes. To be precise, consider a code generated by $\boldsymbol{G}$ and let $\boldsymbol{H}$ be a generator for its dual. Then, it is a well-known fact that any matrix of the form $\boldsymbol{THQ}$ is a generator for the dual of the code generated by $\boldsymbol{SGQ}$, where $\boldsymbol{S}, \boldsymbol{T} \in \mathrm{GL}_k(q)$ and $\boldsymbol{Q} \in \mathrm{Mono}_n(q)$. Using this, an attacker trying to recover Alice's information can write a new instance using the dual codes, as follows. First, the attacker computes the matrix $\boldsymbol{H}$ from $\boldsymbol{G}$, and after receiving $\boldsymbol{G}_a$, he computes the corresponding parity-check matrix $\boldsymbol{H}_a$. Then, it must be that $\boldsymbol{H}_a = \boldsymbol{THQ}_a$ for some $\boldsymbol{T} \in \mathrm{GL}_k(q)$. The attacker can now generate the instance $\hat{\boldsymbol{H}}_a = \boldsymbol{T}\hat{\boldsymbol{H}}$ as above, and solve for $\boldsymbol{T}$. In fact, this system is now overdetermined and, again, it will produce the correct solution with overwhelming probability. The matrix $\boldsymbol{Q}_a$ is then straightforwardly revealed, as before.

## Formal Security Clarification

In Theorem 2, the authors show that a sub-CLE instance can be reduced to an instance of the Linear Code Equivalence (LE) problem in polynomial time. We explain this reduction. Suppose that $\boldsymbol{G}_2 = \boldsymbol{SG}_1\boldsymbol{Q}$ for some invertible matrix $\boldsymbol{S}$ and monomial matrix $\boldsymbol{Q}$. By generating a random invertible $k \times k$ matrix $\boldsymbol{S}_2$ and a random $k \times n$ matrix $\boldsymbol{R}_1$, one can always construct two matrices $\tilde{\boldsymbol{G}}_1 = \begin{pmatrix} \boldsymbol{G}_1 & 0 \\ 0 & \boldsymbol{R}_1 \end{pmatrix}$ and $\tilde{\boldsymbol{G}}_2 = \begin{pmatrix} \boldsymbol{G}_2 & 0 \\ 0 & \boldsymbol{S}_2\boldsymbol{R}_1 \end{pmatrix}$. It is not difficult to show that

$$\begin{pmatrix} \boldsymbol{S} & 0 \\ 0 & \boldsymbol{S}_2 \end{pmatrix} \tilde{\boldsymbol{G}}_1 \begin{pmatrix} \boldsymbol{Q} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix} = \tilde{\boldsymbol{G}}_2. \tag{1}$$

Thus an instance of the LE problem always produces an instance of sub-CLE problem. One can check that a solution to Equation (1) will always produce a solution to $\boldsymbol{G}_2 = \boldsymbol{SG}_1\boldsymbol{Q}$.

By this reduction of an LE problem to a sub-CLE problem, it is suggested that in general, it should be difficult to solve an instance of the sub-CLE problem. However, this does not tell us the whole story. The reduction method actually tells us that a sub-CLE instance in the form of Equation (1) is not easy to solve, hence we can say this is a worst-case scenario. However, from

[1] Note that, despite their own recommendation, the authors propose wrong parameters in Table 1, e.g. $n = 400, k = 194$.

this we cannot tell anything about the average case and, as we have shown in the previous part, the sub-LE problem can be efficiently solved with very high probability. To see why the previous attack does not work on the instance of Equation (1), one can try to apply the attack. For simplicity, we assume that $n = 2k$ and let $\tilde{\boldsymbol{G}}_1 = \begin{pmatrix} \boldsymbol{G}_1 & 0 \\ 0 & \boldsymbol{R}_1 \end{pmatrix}$ and $\tilde{\boldsymbol{G}}_2 = \begin{pmatrix} \boldsymbol{G}_2 & 0 \\ 0 & \boldsymbol{R}_2 \end{pmatrix}$. Suppose that $\boldsymbol{S}\tilde{\boldsymbol{G}}_1 \begin{pmatrix} \boldsymbol{Q} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix} = \tilde{\boldsymbol{G}}_2$. Notice that this implies $\boldsymbol{S}$ is of the form $\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_1 & 0 \\ 0 & \boldsymbol{S}_2 \end{pmatrix}$. Therefore the equation becomes

$$\begin{pmatrix} \boldsymbol{S}_1\boldsymbol{G}_1\boldsymbol{Q} & 0 \\ 0 & \boldsymbol{S}_2\boldsymbol{R}_1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_2 & 0 \\ 0 & \boldsymbol{R}_2 \end{pmatrix}.$$

Since, our attack consists first of considering the part where the columns are not permuted, then we take the right half of this equation. That gives us $\boldsymbol{S}_2\boldsymbol{R}_1 = \boldsymbol{R}_2$. This only helps us recover $\boldsymbol{S}_2$ but we do not have any information about $\boldsymbol{S}_1$ and therefore it will not be possible to recover $\boldsymbol{S}$.

To conclude, our attack cannot be used to solve a particular instance produced by the reduction of an LE instance to a sub-CLE instance. In other words, our attack does not help solving the Linear Code Equivalence problem. However, for random instances of sub-LE, our attack works with overwhelming probability.

## References

[1] Zhuoran Zhang and Fangguo Zhang. Code-based non-interactive key exchange can be made. Cryptology ePrint Archive, Report 2021/1619, 2021. *https://ia.cr/2021/1619*.