The "Quantum Annoying" Property of Password-Authenticated Key Exchange Protocols

Edward Eaton[1,2] Douglas Stebila[1]

[1]University of Waterloo [2]ISARA corporation

PAKE: Password-Authenticated Key Exchange

Goal: Perform a key exchange with authentication coming from a shared *low entropy* password.

Party A		Party B
Input: pw		Input: pw
$G \leftarrow H_1(pw)$		$G \leftarrow H_1(pw)$
$u \leftarrow_{\$} \mathbb{Z}_p$		$v \leftarrow_{\$} \mathbb{Z}_p$
$U \leftarrow G^u$	$U \rightarrow$	$V \leftarrow G^v$
	$\leftarrow V$	
$K \leftarrow V^u$		$K \leftarrow U^v$
Return		Return
$sk \leftarrow H_2(K)$		$sk \leftarrow H_2(K)$

Figure 1:A simplification of the CPace protocol [1]. Derived from SPEKE, CPace at its core derives a generator G from the password and then performs a Diffie-Hellman key exchange with the generator serving as the base point.

Breaking CPace with a quantum computer

This paper explores the ability of a quantum-enabled adversary to compromise the security of CPace. Despite the protocol being based on solving discrete logarithms, the scheme does not appear to be trivially broken by a quantum computer.

PAKE security limitations

As passwords are low-entropy, an adversary can always *guess* the password and then impersonate a party to see if the session completes and their guess is correct. This means the security of PAKEs is always limited by the number of online interactions an adversary makes.

If we assume that the password is drawn uniformly from a password space of size N and the adversary makes q_C online interaction, this means that the adversary will always have at least an advantage of q_C/N.

This advantage is unavoidable, so PAKEs focus on preventing *offline dictionary attacks*. In such an attack an adversary can observe or interact with a small number of sessions and then perform an offline computation over the password space to recover the password.

With a quantum computer

Despite the scheme being based on the discrete logarithm problem, the scheme is not obviously immediately broken by a quantum computer. An adversary can see U and V by observing a CPace session but does not have knowledge of the base point G. Thus one of the inputs to Shor's algorithm is missing.

A quantum adversary can perform an offline dictionary attack by guessing the password and then performing Shor's algorithm to see if their guess was correct. But this corresponds to *solving a discrete logarithm for each guess of the password*.

Quantum Annoying

Coined on the CFRG mailing list [2], a scheme is *Quantum Annoying* if it can be broken by quantum computers, but only through the computation of many discrete logarithms.

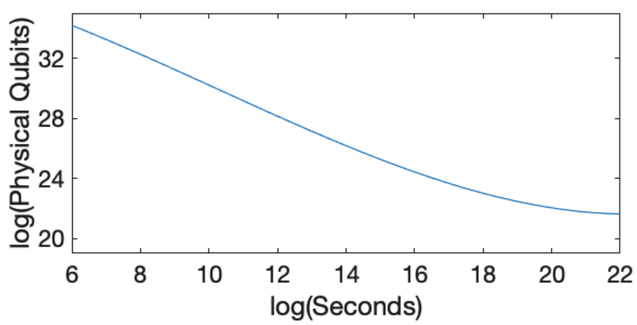

Figure 2:Estimates for the time / qubit trade-off for solving a discrete logarithm on the NIST P-160 curve. Estimates come from [3]. This estimate suggests roughly 18 million physical qubits to solve a discrete logarithm in a day, 308 million to solve in an hour, and 17.6 billion to solve in a minute.

If a scheme only allows for one guess at the password for each discrete logarithm solution, then the attack may not be feasible if the password space is large enough. Furthermore, since an adversary can already attempt an online attack, solving discrete logarithms may never be the weakest point in the security of the scheme.

Early quantum-enabled adversaries will want to compartmentalize use of quantum computer to the smallest and shortest possible useful computation. Otherwise, overhead caused by quantum error correction becomes prohibitive. Smallest useful computation could plausibly be thought of as a single DLOG.

This gives us the intuition for a security model: Provide the adversary with a *discrete logarithm oracle* but otherwise insist that they behave classically. This is a strong assumption that there are not other clever useful quantum computations an adversary can make. Nonetheless, it allows us to proceed and make concrete statements about the number of DLOG oracle queries needed.

The Generic Group Model

Similar to the random oracle model:

- In ROM: No specific hash function, adversary must query to get result of $H(x)$
- In GGM: No specific group instantiation, adversary must query to get representation of $g_1 \star g_2$. Representation of group elements that adversary gets have no structure.

Example: adversary queries (`CAC8286B30D35FFA` $\star$ `DA5A99F0116C5182`), receives response `4CEE32B98FA6C7B2`

Proof Outline

Idea: maintain a `secret representation' of group elements as elements in $\mathbb{Z}_p$ (additive).

Label	Public representation	Secret representation
$\mathcal{I}$	`31A2541CB2983A54`	0
$\mathcal{B}$	`E46E81D638111772`	1
U	`D387B63C1F0E6F30`	2579532190003582
V	`7BCA49C340E2253B`	5689843322479853
$U \star \mathcal{B}$	`39D4C0000B0B819F`	2579532190003583

In this view, answering group operation queries corresponds to addition $(\mathrm{mod}\ p)$. To answer $DLOG(a, b)$: Retrieve secret representations s_a, s_b and return $s_a^{-1} s_b$. We want to know when Adversary has enough information to calculate G^{uv}. Rather than keeping a specific representation G, maintain a variable g.

Label	Public representation	Secret representation
G	`9E5F2A3D71683A54`	g
$G \star G$	`64D2BAFCC78E715D`	$2g$
$G \star G \star \mathcal{B}$	`80EA053530DCF527`	$2g + 1$

As long as g is undefined, so is $DLOG(G, U)$ and $DLOG(G, V)$. So it is only possible to calculated G^{uv} after g has become defined. Adversary can cause this to happen with a $DLOG$ query that involves G. Adversary could also do this with enough group operation oracle queries as well.

When we are considering multiple possible generators $G_1, G_2, ...G_N$ (one for each password) then each DLOG query reduces the rank of this system by at most 1 (enforces a linear relationship between the g_i's). Thus after q_D queries to $DLOG$, at most q_D of the g_i variables are defined, giving the adversary q_D guesses at the password.

Say a query is made where the secret representation of the group elements is

$$(a_0 + a_1 g_1 + \cdots + a_N g_N, b_0 + b_1 g_1 + \cdots + b_N g_N).$$

A response δ is an attestation that $(\delta\vec{a} - \vec{b}) \cdot \vec{g} = b_0 - \delta a_0$. This is a linear constraint on the N variables, so viewed as a linear system each such query reduces the rank of the g_i variables by 1.

Results

Adversary's advantage is bounded by:

$$\frac{1}{2} + \frac{q_D + q_C}{N} + O\left(\frac{q_D q_G^2}{p}\right)$$

where q_D is the number of DLOG oracle queries, q_C is the number of online interactions made by the adversary, q_G is the number of group operation oracle queries, N is the size of password space, and p is the order of the group.

References

[1] Michel Abdalla, Björn Haase, and Julia Hesse. "Security Analysis of CPace", 2021. `ia.cr/2021/114`.

[2] Steven Thomas. ``Re: [Cfrg] CPACE: what the ``session id" is for?", 2020. *CFRG Mailing List*. `https://mailarchive.ietf.org/arch/msg/cfrg/dtf91cmavpzT47U3AVxrVGNB5UM/`.

[3] Vlad Gheorghiu and Michele Mosca. ``Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes", 2019. `https://arxiv.org/abs/1902.02332`.