Password Authenticated Key Exchanges (PAKEs) and the 'Quantum Annoying' Property

Edward Eaton and Douglas Stebila

July 20, 2021

This talk

The "Quantum Annoying" property: a quantum computer can break a system, but only by solving many discrete logarithms.

Discussing challenges in defining security models to prove this property and our proof that CPace is quantum annoying in the generic group model.

Why "post-quantum cryptography"? Talking about cryptography in the presence of a quantum computer.

Outline

- What is a PAKE and CPace
- CPace security and background
- The quantum annoying property
- Security model
- Proof methodology

PAKE

Password Authenticated Key Exchange.

Perform a key exchange where authentication comes from a low-entropy password Comes in two flavours, balanced and augmented.

Balanced:

- Setup: Two parties A and B share a low-entropy password pw.
- Goal: A and B share a high-entropy session key sk and are assured that they are communicating with someone who really knows the password.

CPace — simplified

Security Analysis of CPace by Michel Abdalla, Björn Haase, and Julia Hesse. `ia.cr/2021/114`

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

CPace security

Active adversary:

- Adversary can always guess pw, engage in protocol with target, and see if protocol completes.
- Each online interaction will always provide the adversary with one guess at the password.

Passive adversary:

- Adversary observes $U = G^u$ and $V = G^v$, but does not know G.
- No obvious action you can take to recover sk just from these observations.

Passive Adversary — Intuition

Enumerate password space as $pw_1, pw_2, \ldots, pw_N$.

From this, generators $G_1, G_2, \ldots, G_N$.

For any U, V, and index i, there exits u', v' such that $G_i^{u'} = U$ and $G_i^{v'} = V$.

Given a U, V any pw_i is equally likely, and causes a different sk_i that can only be calculated by solving a DH problem.

History

2019: Crypto Forum Research Group (CFRG) asks for candidate PAKEs for use in IETF protocols. One balanced, one augmented.

Four balanced and four augmented considered

2020: CPace announced as the balanced winner, OPAQUE as the augmented winner

Considerations in deciding winners: aspects of security proofs, flexibility, efficiency of protocols, ease of transition to post-quantum version ...

Can you compromise CPace with a quantum computer?

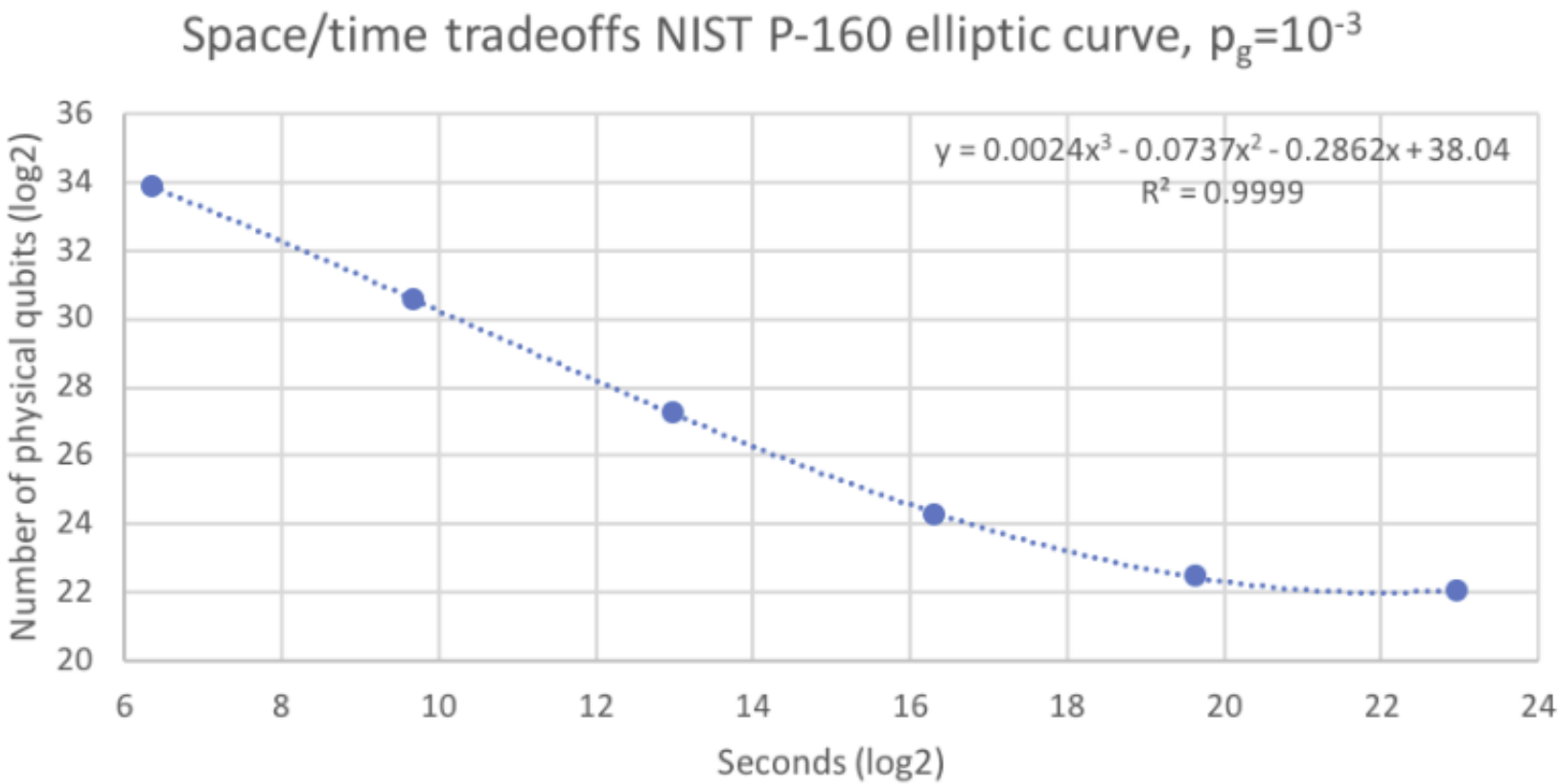
A quantum computer lets you solve $DLOG(G, U)$ for inputs G and U... but in CPace, the generator is unknown without the password!

You can guess the password, and then solve a DLOG, but then you are solving an entire discrete logarithm problem for each guess of the password!

Estimates from Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes by Gheorghiu and Mosca:

Solving a DLOG problem over NIST P-160 curve:

- In one day: $\sim$ 18 million physical qubits
- In one hour: $\sim$ 308 million physical qubits
- In one minute: $\sim$ 17.6 billion physical qubits



Property was dubbed "Quantum Annoying" by Steve Thomas: In the presence of an adversary with a quantum computer, the only feasible offline attack is one that requires a discrete logarithm solution for each guess of the password.

May not need to replace a quantum annoying scheme as quickly. Online attacks may still be easiest way to break PAKE.

Just one question: how do we prove that a scheme is quantum annoying?

Selecting a Model

Asymptotically, an adversary can do better than one DLOG / guess:

Put Shor's algorithm inside Grover's algorithm! Asymptotically: $\sqrt{N}$ Shor's algorithm iterations.

This may be asymptotically optimal (although unclear how to prove that) but likely does not reflect the best approach for early quantum adversaries.

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.

Selecting a Model

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.

Provides a good sanity check for the quantum annoying property and rules out the simplest attacks.

Generic Group Model

Often used to prove generic hardness of DLOG problem / variants

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.

In GGM, best attack on DLOG is baby-step giant-step

GGM does not hold up for $\mathbb{Z}_p$ but doesn't do bad for elliptic curve groups

Our Model

We will extend the generic group model by providing the adversary classical access to a discrete logarithm oracle $DLOG(\cdot, \cdot)$ as well as classical access to the group operation oracle $\star$.

Task: bound the adversary's probability of success in terms of the number of discrete logarithm oracle queries made.

To our knowledge, adding a DLOG oracle to GGM is a novel contribution.

Proving security in this model

Take a simple case. Adversary sees group elements G, $U = G^u$, and $V = G^v$.

Adversary has access to group operation oracle $\star$ and discrete logarithm oracle $DLOG$.

How do we "know" when the adversary is able to calculate G^{vu}?

Given G, $U = G^u$, $V = G^v$, and oracles $DLOG$, $\star$, calculate G^{uv}

Obviously if adversary queries $DLOG(G, U) \rightarrow u$ they can return V^u.

What if they query $DLOG(G, U \star U)$? $DLOG(G, U \star V)$? $DLOG(G, G \star U)$?

How do we know when the queries they've made have told them enough to calculate G^{uv}?

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
$\mathcal{B} \star \mathcal{B} \star \mathcal{B}$	`48AA20A7D65DC34F`	3
U	`D387B63C1F0E6F30`	2579532190003582
V	`7BCA49C340E2253B`	5689843322479853
$U \star \mathcal{B}$	`39D4C0000B0B819F`	2579532190003583

In this view, answering group operation queries corresponds to addition $\pmod p$

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
$\mathcal{B}$	`E46E81D638111772`	1
G	`9E5F2A3D71683A54`	g
$G \star G$	`64D2BAFCC78E715D`	$2g$
$G \star G \star \mathcal{B}$	`80EA053530DCF527`	$2g + 1$

Secret representations are now in $\mathbb{Z}_p[g]/\langle g^2 \rangle$.

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 — more on this later.

When we are considering multiple possible generators G_1, G_2, ... G_N (one for each password) then:

- Maintain a secret variable g_i for each G_i.
- Secret representations are now linear functions in the g_is. $\mathbb{Z}_p[g_1, \ldots, g_N]$.
- We have a linear system with N variables.
- 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.
- Each defined g_i could result in one guess at the password. N total, so q_D/N chance of getting password.

Why does a query to the DLOG oracle reduce the rank of the system by at most 1?

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

$$\begin{aligned} & \delta(a_0 + a_1 g_1 + \cdots + a_N g_N) = b_0 + b_1 g_1 + \cdots + b_N g_N \\ \iff & (\delta \vec{a} - \vec{b}) \cdot \vec{g} = b_0 - \delta a_0 \end{aligned}$$

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.

Additional considerations:

- How do we choose the responses to DLOG oracle δ so that we are consistent with all previous DLOG and group operation oracle queries?
- How do we respond to group operation oracle queries in a way consistent with all previous group operation oracle queries and with all previous DLOG queries?
- How does an active adversary, who contributes half of the Diffie-Hellman pair, change the situation?
- How do we interface with the multi-instance nature of the adversarial model?

End result: with this model, quantum annoying is meaningful. An adversary needs to make a DLOG query for each password guess, no more, no less.

Adversary's advantage is bounded by:

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

- q_D: number of DLOG oracle queries
- q_C: number of online interactions made by the adversary
- q_G: number of group operation oracle queries
- N: size of password space
- p: order of group

Conclusion

Quantum annoyingness is meaningful, at least in our limited model.

Future work:

- Can quantum annoyingness be proved in a more realistic model?
- How do our techniques apply to other PAKEs?

Thank you for your attention.