PQSHIELD

Think openly, build securely

White paper:

Scalable Ciphertext Compression Techniques for Post-Quantum KEMs and their Applications

Software | PQShield | November 10, 2020

A standard method to establish secure communications between two or more parties is to encrypt a common session key via a key encapsulation mechanism (or KEM). In this document, we propose compression techniques that allow, when the number of parties is large (10 or more), to divide by an order of magnitude the cost of this approach when used with post-quantum KEMs.

This has several potential applications to secure group messaging (e.g. Signal, WhatsApp, etc.). In particular, we show that it can be used inside the draft IETF standard MLS to reduce its bandwidth footprint by about a factor 2.

The companion article to this white paper is available at https://ia.cr/2020/1107.

1 Introduction

Secure communication within a system of several users is becoming indispensable in our everyday lives. One leading example is the recent trend in secure group messaging (Zoom, Signal, WhatsApp, etc.) to handle large groups – up to 50 000 users according to the IETF draft of the Message Layer Security (MLS) architecture [OBR+20]. The scenario is that users in a system, each holding their public and private key, frequently exchange messages with a group of users. The standard solution consists of individually encrypting the same message M using the public keys associated with the respective recipients in the group. However, the required *bandwidth* and *computational costs* grow by a factor N (where N is the number of recipients), compared to sending a message to a single recipient. Hence this scales poorly in the number of recipients.

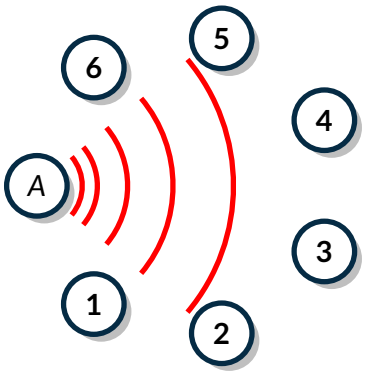


Figure 1: Broadcast

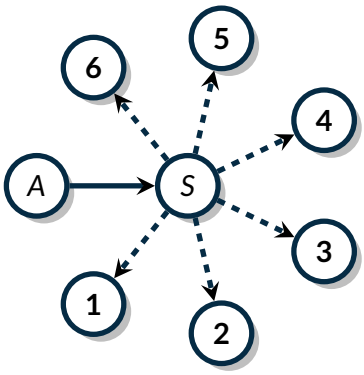


Figure 2: Server-aided group encryption

This problem is even more pronounced with post-quantum cryptography — that is, cryptography expected to be resilient against quantum adversaries. Virtually all post-quantum schemes incur large bandwidth and/or computational overheads compared to classical schemes. For example, all key encapsulation mechanisms (KEMs) still considered for standardization by NIST require an order of magnitude more bandwidth than ECDH [BCR+]. Thus, lowering the communication cost, even for a moderately large number of recipients N, say $N \geq 10$, is already valuable.

Multi-Recipient Key Encapsulation Mechanism (mKEM)

mKEM [Sma05] is a primitive designed with the previous motivations in mind. An mKEM is like a standard KEM that securely sends *the same* session key K to a group of recipients. Subsequently, the sender transmits *a single* ciphertext to all the recipients by encrypting the message M using K as a secret key for a secret-key encryption scheme. The appeal of mKEMs resides in their potential to slash the bandwidth and computational requirements.

However, almost all the literature on mKEMs is based on classical assumptions (e.g., Diffie-Hellman type assumptions) which do not endure quantum adversaries. Earlier works also require the underlying PKE to satisfy specific properties that seem somewhat tailored to classical Diffie-Hellman type assumptions. Therefore our first question is:

> *(Theoretical Question) Are there simple and efficient generic constructions of mKEM that can be based on versatile assumptions, including post-quantum ones?*

All previous works on mKEM were fairly theoretical, and did not provide an implementation. Since gains depend on the concrete mKEM implementation and on the choice of KEM used in the trivial solution, the benefit of an mKEM is unclear without proper comparison. Our second question is:

> *(Practical Question) What is the concrete gain of using an mKEM compared to the trivial solution? What are the concrete applications of mKEMs?*

1.1 Our Contributions

Theoretical Contribution

We provide a new simple and efficient generic construction of an IND-CCA secure multi-recipient KEM (mKEM) from any IND-CPA secure multi-recipient PKE (mPKE).[1] The construction is proven secure in the classical *and* quantum random oracle model ((Q)ROM). We show that IND-CPA secure mPKEs can be constructed easily from most assumptions known to imply standard PKEs. Concretely, we show how to construct mPKEs based on lattices and isogenies. In addition, we only require very natural properties from the underlying mPKE, such as IND-CPA.

Practical Contribution 1

An immediate consequence of our theoretical contribution is to open the door to a large number of post-quantum instantiations of mKEM. A natural next step is to study these mKEM instantiations at a practical level and compare them to the trivial solution of running standard KEMs in parallel. Doing this work is one of our practical contributions. At least 9 post-quantum schemes are compatible with our construction of mKEM:

- CSIDH
- FrodoKEM
- Kyber
- LAC
- NewHope
- Round5
- Saber
- SIKE
- ThreeBears

[1] IND-CPA and IND-CCA are standard notions of security against chosen-plaintext and chosen-ciphertext attacks, respectively. IND-CCA is stronger, but it is typical to prove that a scheme is IND-CPA, and convert it generically into an IND-CCA scheme, which we also do here. PKE stands for public-key encryption scheme.

Out of these 9 schemes, 4 are NIST Round 3 schemes (2 finalists, 2 alternates), and 4 are NIST Round 2 schemes. For a subset of these schemes (CSIDH, FrodoKEM, Kyber, SIKE),[2] we performed a systematic study of their bandwidth efficiency. We found that for all of these schemes, our mKEM variants are more compact than the trivial solution with the original schemes by *at least one order of magnitude*. In addition, we implemented their mKEM counterparts and compared their performance (cycle count). We found our mKEM variants to be (asymptotically) faster than the trivial solution with original schemes by factors ranging from 1.92 to more than 35.

Practical Contribution 2

We show that we can combine the mKEM primitive with the TreeKEM protocol and obtain significant bandwidth savings. The importance of TreeKEM could be best understood by looking at its parent protocol, MLS [OBR+20, BBM+20], a IETF draft for secure (group) messaging that has gained considerable industrial traction. TreeKEM constitutes the cryptographic backbone of MLS, as well as its main efficiency bottleneck. Indeed, given N users, it requires each of them to compute and send $O(\log N)$ ciphertexts at regular intervals. We highlight a simple but powerful interplay between TreeKEM and mKEM, and show that our technique can reduce communication cost by a factor between 1.8 and 4.2 compared to standard post-quantum KEMs.

2 Post-Quantum mKEMs

We provide two types of IND-CPA secure mPKEs instantiations: one based on lattices, and two based on isogenies (in the SIDH and CSIDH setting). We then convert them generically into IND-CCA secure mKEMs in the ROM and QROM. As we see in section 2.3, both types of instantiations fit with many of the NIST round 2 candidate (single-recipient) PKE/KEMs.

2.1 Instantiation from lattice assumptions

The Lindner-Peikert (LP) framework [LP11, LPR10] provides a mPKE with good decomposability properties. We briefly present the LP framework:

- Given a publicly known $\mathbf{A}$, the private key sk_i of the i-th recipient is a pair of matrices $(\mathbf{S}_i, \mathbf{E}_i)$ with short coefficient, and his public key is $\mathsf{pk}_i = \mathbf{B}_i := \mathbf{AS}_i + \mathbf{E}_i$.
- An encryption of M is $(\mathsf{ct}_0, \widehat{ct}_i) = (\mathbf{RA} + \mathbf{E}', \mathbf{RB}_i + \mathbf{E}_i'' + \mathsf{Encode}(\mathsf{M}))$, where $\mathbf{R}, \mathbf{E}', \mathbf{E}_i''$ are random matrices with short coefficients, and $\mathsf{Encode}(\mathsf{M})$ is an encoding of M.

Our mPKE based on the LP framework encrypts M to N recipients as $\vec{ct} = (\mathsf{ct}_0, \widehat{ct}_1, \ldots, \widehat{ct}_N)$, where $\mathsf{ct}_0 = \mathbf{RA} + \mathbf{E}'$ and $\forall i \in [N], \widehat{ct}_i = \mathbf{RB}_i + \mathbf{E}_i'' + \mathsf{Encode}(\mathsf{M}))$. From a theoretical point of view, one can reduce the security of our proposal to LWE with arbitrary many samples, which is itself equivalent to LWE with a finite number of samples [GMPW20]. The security with respect to known attacks also remains unchanged for the schemes we considered in practice. The LP framework can be instantiated with the LWR assumption instead of LWE, in which case our construction still applies.

[2] The full version of this paper studies the bandwidth efficiency of all 9 schemes.

2.2 Instantiation from isogeny assumptions

We briely recall the SIDH PKE [DFJP14], which essentially applies the hashed El Gamal [ElG85] construction to the SIDH key-exchange. Let p be an odd prime of the form $2^{e_2}3^{e_3} - 1$, and E be a supersingular elliptic curve over $\mathbb{F}_{p^2}$ such that $|E(\mathbb{F}_{p^2})| = (2^{e_2}3^{e_3})^2$. We denote by $B_2 = \{P_2, Q_2\}$ and $B_3 = \{P_3, Q_3\}$ bases for the torsion subgroups $E[2^{e_2}]$ and $E[3^{e_3}]$, respectively. Public parameters are $\mathsf{pp} = (E, \{(e_j, B_j)\}_{j=2,3}, H)$, where H is a hash function.

- The private key sk_i of the i-th recipient is an isogeny $\varphi_{\langle R_3^{(i)} \rangle} : E \to E/\langle R_3^{(i)} \rangle$ and its kernel $\langle R_3^{(i)} \rangle$. The public key is $\mathsf{pk}_i = (E_3^{(i)}, U_2^{(i)}, V_2^{(i)})$, where:

$$E_3^{(i)} = E/\langle R_3^{(i)} \rangle, \qquad U_2^{(i)} = \varphi_{\langle R_3^{(i)} \rangle}(P_2), \qquad V_2^{(i)} = \varphi_{\langle R_3^{(i)} \rangle}(Q_2).$$

- An encryption of M is $(\mathsf{ct}_0, \widehat{ct}_i)$, where:
  - $\mathsf{ct}_0 = (E_2, U_3, V_3) := (E/\langle R_2 \rangle, \varphi_{\langle R_2 \rangle}(P_3), \varphi_{\langle R_2 \rangle}(Q_3))$, for a random isogeny R_2.
  - $\widehat{ct}_i = H(J_i) \oplus \mathsf{M}$, where J_i is the j-th invariant of $E/\langle R_2, R_3^{(i)} \rangle$, which can be efficiently computed from pk_i and R_2.

As with the LP framework, our mPKE encrypts M to N recipients as $\vec{ct} = (\mathsf{ct}_0, \widehat{ct}_1, \ldots, \widehat{ct}_N)$, mutualizing the cost of ct_0. We show that its security relies on SSSDH [DFJP14], the decisional variant of the problem CSSDH used in SIKE. Our construction also applies to CSIDH.

2.3 Practical mKEM Instantiations

In this section, we concretely instantiate the generic mKEM framework laid out in previous sections. We take the PKEs underlying these 4 schemes: Kyber, FrodoKEM, SIKE and CSIDH. We first modify them into efficient mPKEs (following section 2) and then into mKEMs via our generic transformation. We then compare the bandwidth and computation efficiency of the standard KEM and their mKEM variants. Results are given in Table 1, and show that mKEMs typically provide gains of *an order of magnitude* in communication and computation.

Our methodology in a nutshell

Until the end of this document, we denote by $|\mathsf{x}|$ the bytesize of an object x. We use two metrics (k_{comm} and k_{cycles}) to compare mKEMs to the trivial solution that uses (single-recipient) KEMs in parallel. The first metric k_{comm} measures the (asymptotic) ratio between the data sent when N KEMs are used in parallel, versus one mKEM with N recipients. Hence we have:

$$k_{\mathsf{comm}} = \lim_{N \to \infty} \frac{N|\mathsf{ct}_0| + N|\widehat{ct}_i|}{|\mathsf{ct}_0| + N|\widehat{ct}_i|} = 1 + \frac{|\mathsf{ct}_0|}{|\widehat{ct}_i|}$$

Similarly, k_{cycles} is the ratio (number of cycles to perform N encapsulations using KEM) / (number of cycles to encapsulate 1 message to N recipients using mKEM). Measurements are performed on a processor i7-8665U (Whiskey Lake) @ 1.90GHz, with Turbo Boost disabled.

Table 1: Impact of our solution on various schemes.
For *Bandwidth*, sizes are in bytes. For *Speed*, times are in kilocycles.

Scheme	Bandwidth				Speed		
	$\lvert ct_0 \rvert$	$\lvert \widehat{ct}_i \rvert$	$\lvert ct \rvert$	k_{comm}	KEM	mKEM	k_{cycles}
FrodoKEM-640	9 600	120	9 720	**81**	4 949	251	**19.68**
FrodoKEM-976	15 616	128	15 744	**123**	10 413	388	**26.86**
FrodoKEM-1344	21 504	128	21 632	**169**	18 583	520	**35.74**
Kyber-512	640	96	736	**7.67**	181	43	**4.25**
Kyber-768	960	128	1 088	**8.5**	279	52	**5.32**
Kyber-1024	1 408	160	1 568	**9.8**	415	62	**6.71**
SIKE/p434	330	16	346	**21.6**	1 657 655	759 202	**2.18**
SIKE/p503	378	24	402	**16.8**	2 301 014	1 037 470	**2.22**
SIKE/p751	564	32	596	**18.6**	6 900 792	3 150 070	**2.19**
CSIDH-512 (PKE)	64	16	80	**5**	37 455 411	19 438 022	**1.92**

3 Application to Secure Messaging

In this section, we show how our mKEM can be used to optimize the *TreeKEM* protocol [BBR18, ACDT20] used within secure group messaging. The resulting protocol has a lower communication cost than the standard version of TreeKEM [BBR18, ACDT20].

Continuous group key agreement (CGKA)

CGKA forms the backbone of secure *group* messaging (SGM) protocols. Informally, one can think of CGKA as a group key exchange where the group members dynamically change and the (group) session keys need to be re-established in each epoch to maintain strong security. Once a session key is established for a given epoch, a user can use the key to securely communicate with group members. Thus a SGM protocol can be described as a continuum of running CGKA and exchanging secured messages. We focus on TreeKEM, the CGKA at the heart of the SGM protocol MLS [BBM+20]. It was first described in [BBR18] and various improvements have been proposed, see e.g. [ACDT20]. TreeKEM is one of MLS' main bottlenecks due to the large amount of public key material sent. Our efforts are directed at optimizing the Update algorithm of TreeKEM, de-

scribed in section 3.1. Since this operation is one of the most frequently performed by TreeKEM, we expect that improving the efficiency of Update will improve the efficiency of TreeKEM (and hence the MLS protocol) on a similar scale. Details on TreeKEM follow.

Dendrologic notations

In a (binary or m-ary) tree T, a *leaf* is a node with no child, an *internal node* is a node that is not a leaf, and the root root is the unique node that has no parent. By synecdoche, we may abusively refer to a node by its label; for example in Figure 3, "1" denotes the bottom left node.

Let u be a node in a tree T. Its siblings, $\text{siblings}(u)$, is the set of nodes $v \neq u$ in T with the same parent as u. Its *path*, $\text{path}(u)$, is the set of nodes between u and root, including u but excluding root. Its *co-path*, $\text{copath}(u)$, is the set of siblings of nodes in its path: $\text{copath}(u) = \bigcup_{v \in \text{path}(u)} \text{siblings}(v)$. For example, in Figure 3, the only sibling of "1" is "2", its path is the set of red nodes (●), and its co-path is the set of green nodes (●).

3.1 TreeKEM and our Proposal

In TreeKEM, a (binary or m-ary) tree T is constructed with the N group members as its leaves. As an example, Figure 3 illustrates the tree T associated to a group of 16 users numbered from 1 to 16. Let PRG be a pseudorandom generator. Then, to each node i is associated a secret seed seed_i and a keypair $(\text{pk}_i, \text{sk}_i) = \text{mGen}(\text{pp}; \text{PRG}(\text{seed}_i)_\text{L})$, where $\text{PRG}(\cdot)_\text{L}$ (resp. $\text{PRG}(\cdot)_\text{R}$) denotes the left (resp. right) half output of the PRG. In particular, mGen is run on randomness $\text{PRG}(\text{seed}_i)_\text{L}$. The root does not need a keypair, as its seed will be the group secret I (i.e., session key). The *TreeKEM invariant* states that a group member u knows seed_i if and only if $i \in \text{path}(u)$. When a user u performs an update (via Update), he does the following:

1. Generate a new secret seed seed_u for the leaf u.
2. For each node i in $\text{path}(u)$, update its keypair: $(\text{pk}_i, \text{sk}_i) = \text{mGen}(\text{pp}; \text{PRG}(\text{seed}_i)_\text{L})$, then compute a new secret seed for its parent node: $\text{seed}_{\text{parent}(i)} = \text{PRG}(\text{seed}_i)_\text{R}$.
3. For each node i in $\text{path}(u)$, compute the multi-recipient ciphertext

$$\vec{ct}_i \leftarrow \text{mEncaps}(\text{pp}, (\text{pk}_j)_{j \in \text{siblings}(i)}; \text{seed}_{\text{parent}(i)}).$$

   Note that here mEncaps is running with randomness $\text{seed}_{\text{parent}(i)}$.
4. Send the update package $(\text{pk}_i, \vec{ct}_i)_{i \in \text{path}(u)}$ to the server, which dispatches it to the other group members (this is known as *server-side fan-out*).

Upon receiving the update package, a user v processes it (via Process) as follows:

1. Update each public key pk_i he received.
2. Compute the closest common ancestor node w of the nodes u and v, then recover its corresponding seed_w by decapsulating the adequate ciphertext $\vec{ct}_i$.
3. Recover the secret seeds of all remaining common ancestors of u and v by computing $\text{seed}_{\text{parent}(i)} = \text{PRG}(\text{seed}_i)_\text{R}$. The update secret is $I = \text{seed}_{\text{root}}$

This description is more generic than previous ones [BBR18, ACDT20, BBM+20], since all existing instantiations of TreeKEM take T to be a binary tree, in which case a single-recipient KEM suffices. While our description uses mKEM as a building block, it is easily adapted to work with an mPKE.

Figure 3 illustrates the "classical" instantiation of TreeKEM, with "1" sending an update. Each update package contains $\lceil \log_2(N) \rceil$ public keys (one for each node ( ) in the path), and as many ciphertexts (one for each node ( ) in the co-path). Hence its bytesize is at most:

$$\lceil \log_2(N) \rceil \cdot \left(|\mathsf{pk}| + |\mathsf{ct}_0| + |\widehat{ct}_i| \right) \tag{1}$$

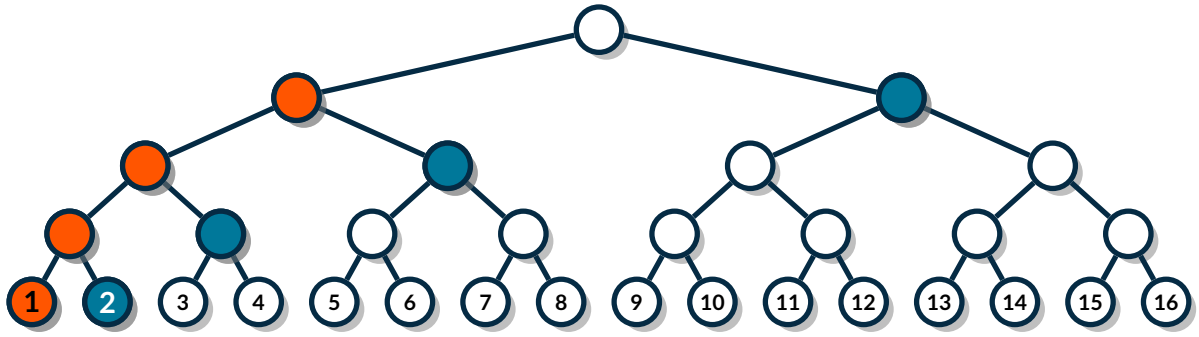


Figure 3: TreeKEM.

We now show how to obtain efficiency gains by instantiating TreeKEM with mKEM. As mentioned in [BBR18], TreeKEM can be instantiated with an m-ary tree; see Figure 4 for an example where "1" issues a package update. At first, it is not obvious that Figure 4 is more efficient than Figure 3, since an update package now contains 2 public keys and 6 ciphertexts.

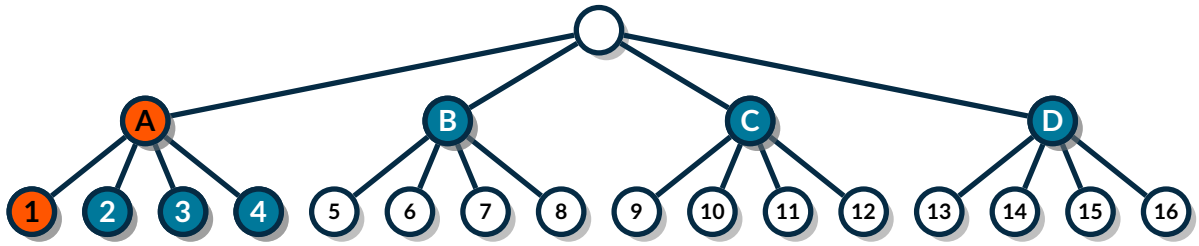
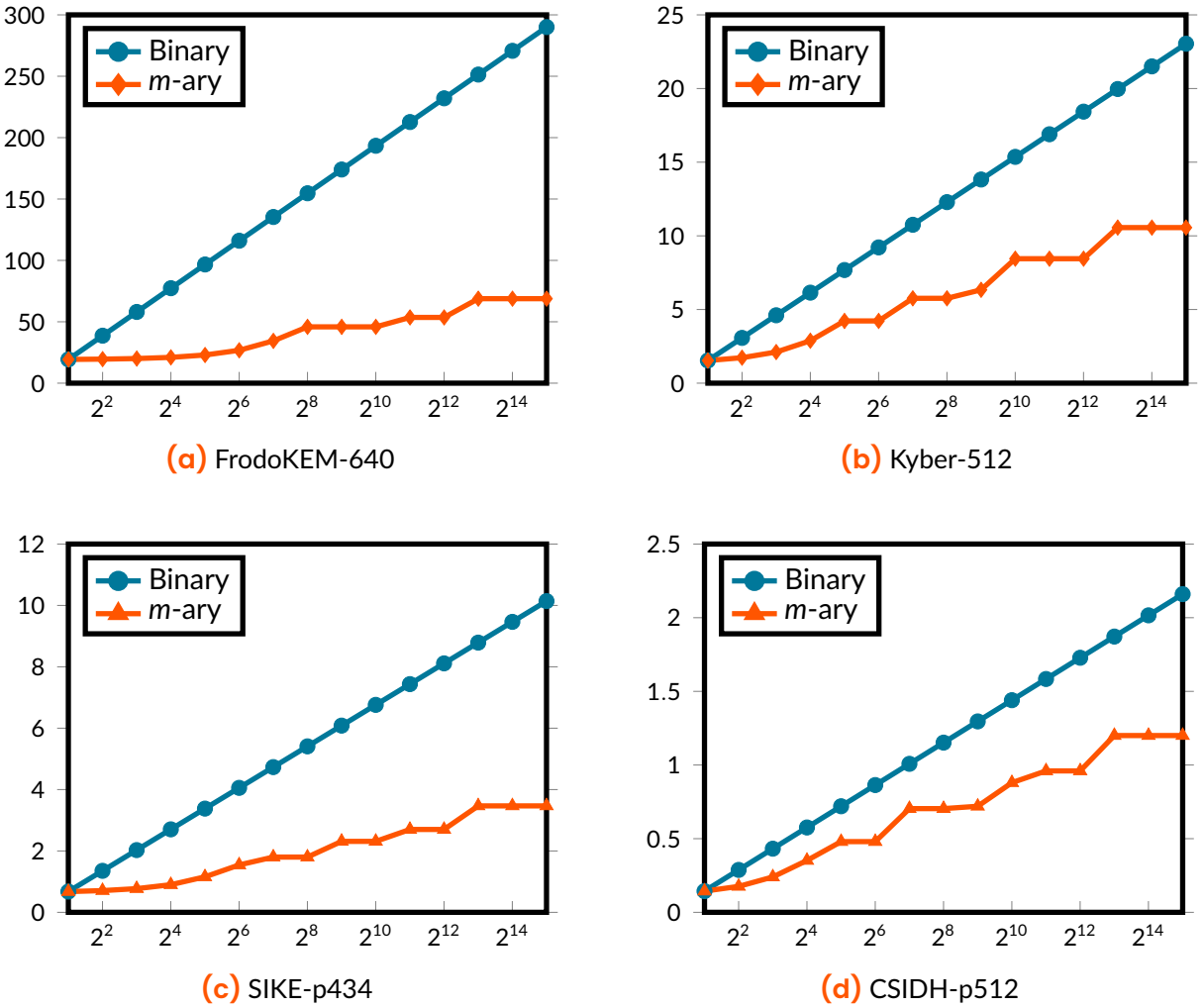


Figure 4: 4-ary TreeKEM

Note that when a user u issues an update, the update package may encapsulate several times the same information. Precisely, for each $i \in \mathsf{path}(u)$, the update package encapsulates $\mathsf{seed}_{\mathsf{parent}(i)}$ under the key pk_j for each $j \in \mathsf{siblings}(i)$. In the example of Figure 4, this means that an update package issued by 1 encapsulates seed_A under $\mathsf{pk}_2, \mathsf{pk}_3, \mathsf{pk}_4$, and $\mathsf{seed}_{\mathsf{root}}$ under $\mathsf{pk}_B, \mathsf{pk}_C, \mathsf{pk}_D$. The bandwidth gain happens exactly here: since the same value seed_A is encapsulated under $\mathsf{pk}_2, \mathsf{pk}_3, \mathsf{pk}_4$, one can use mKEM to perform this (multi-)encapsulation. And similarly at each level of the tree. Hence the total size of an update package is at most:

$$\lceil \log_m(N) \rceil \cdot \left(|\mathsf{pk}| + |\mathsf{ct}_0| + (m-1) \cdot |\widehat{ct}_i| \right). \tag{2}$$

It is clear from (2) that whenever $|\mathsf{pk}| + |\mathsf{ct}_0| \gg |\widehat{ct}_i|$, it is advantageous efficiency-wise to take $m > 2$. A good rule of thumb is to take $m - 1 \approx \frac{|\mathsf{pk}|+|\mathsf{ct}_0|}{|\widehat{ct}_i|}$, which provides a gain $O(\log m)$ compared to TreeKEM. We evaluate the bandwidth costs of both TreeKEM and our version, when instantiated with either FrodoKEM, Kyber, SIKE or CSIDH. The results are given in Figure 5, and show that our proposal improves the communication cost for large groups by a factor between 1.8 and 4.2.



(a) FrodoKEM-640 (b) Kyber-512

(c) SIKE-p434 (d) CSIDH-p512

Figure 5: Comparing the classic "binary" TreeKEM with m-ary TreeKEM, when instantiated with either FrodoKEM, Kyber, SIKE or CSIDH. In each case, the x-axis is the number N of group members (from 2 to 2^{15}) and the y-axis is the maximum size of an update package in kilobytes. The arity m depends on the scheme and the group size N, and is omitted for readability. The graphs for "*Binary*" and "*m-ary*" are computed with (1) and (2), respectively.

References

[ACDT20] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2020*, LNCS, pages 248–277. Springer, Heidelberg, August 2020.

[BBM+20] Richard Barnes, Benjamin Beurdouche, Jon Millican, Emad Omara, Katriel Cohn-Gordon, and Raphael Robert. The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-09, Internet Engineering Task Force, March 2020. Work in Progress.

[BBR18] Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS). Research report, Inria Paris, May 2018.

[BCR+] Elaine Barker, Lily Chen, Allen Roginsky, Miles Smid, Elaine Barker, Lily Chen, Allen Roginsky, and Miles Smid. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. In *Technical Report; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2006. 2012*, page 15158. https://doi.org/10.6028/NIST.SP.800-56Ar3.

[BF07] Manuel Barbosa and Pooya Farshim. Randomness reuse: Extensions and improvements. In *IMA International Conference on Cryptography and Coding*, pages 257–276. Springer, 2007.

[DFJP14] Luca De Feo, David Jao, and Jerome Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Journal of Mathematical Cryptology*, volume 8 (3), pages 209–247, 2014.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In *PKC 2020*, LNCS, pages 623–651. Springer, Heidelberg, 2020.

[HTAS09] Harunaga Hiwatari, Keisuke Tanaka, Tomoyuki Asano, and Koichi Sakumoto. Multi-recipient public-key encryption from simulators in security proofs. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 293–308. Springer, Heidelberg, July 2009.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[MH13] Takahiro Matsuda and Goichiro Hanaoka. Key encapsulation mechanisms from extractable hash proof systems, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 332–351. Springer, Heidelberg, February / March 2013.

[OBR+20] Emad Omara, Benjamin Beurdouche, Eric Rescorla, Srinivas Inguva, Albert Kwon, and Alan Duric. The Messaging Layer Security (MLS) Architecture. Internet-Draft draft-ietf-mls-architecture-04, Internet Engineering Task Force, January 2020. Work in Progress.

[Sma05] Nigel P. Smart. Efficient key encapsulation to multiple parties. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 208–219. Springer, Heidelberg, September 2005.