

Cryptanalysis of ring-LWE based key exchange with key share reuse

Scott Fluhrer

Cisco Systems

January 30, 2016

1 Abstract

This paper shows how several ring-LWE based key exchange protocols can be broken, under the assumption that the same key share is used for multiple exchanges. This indicates that, if these key exchange protocols are used, then it will be necessary for a fresh key share be generated for each exchange, and that these key exchange protocols cannot be used as a drop in replacement for designs which use Diffie-Hellman static key shares.

2 Introduction

Key agreement protocols are one of the oldest public key primitives known, dating back to the Diffie-Hellman protocol[1]. In a key agreement protocol, each side selects a private value, and exchange a public value (which is often called a key share); in most such protocols, each side sends a single message. Then, both side do computations based on their private value and the other side's public key shares, and derive the same secret value. The security goal is that someone listening into the exchanged public key shares would find it infeasible to derive that secret value.

Diffie-Hellman (and the similar Elliptic Curve Diffie-Hellman) protocols would be vulnerable to a Quantum Computer; someone with a large Quantum Computer could rederive the private values and thus obtain the common secret value. One research topic is to find alternatives that would be secure in that environment. Several such proposed alternatives are based on the ring-LWE problem. From a protocol standpoint, these proposals work largely like Diffie-Hellman, each side selects private values, one side sends its public value, the other side replies, and then they both compute a shared secret.

With Diffie-Hellman, it is perfectly safe to reuse the same public key share for multiple exchanges. One such use is the "ephemeral-static" mode; in this

case, Alice might select a private value, and publish the corresponding key share. Then, when Bob, Carol, Dave and Eve want to communicate with Alice, they can take Alice's key share, select their own private values, and then send to Alice their key shares, thus creating a secure connection. As long as Alice takes some well-known precautions, the connections are independent; Eve gets no advantage on deriving the secret used in the Alice to Bob connection.

This paper will show that ring-LWE based key agreement protocols are not safe in this mode; if Alice uses the same private value repeatedly, then Eve (by sending a series of messages, and seeing how Alice responds) is able to recover Alice's private value (and if Alice used that same static value to generate the keys used to communicate to Bob, Eve can then listen into that conversation).

3 Ring-LWE Key Exchange

The Ring-LWE problem[4] works in the Ring $\mathbb{Z}[x]/(x^N+1, p)$, for integers N and p . A member of this ring can be viewed as consisting of N integers (coefficients) each in $[0, p)$. It has been proposed as the basis for a number of cryptographical systems. These include a number of key agreement protocols, such as [3], [5], [2], and [7].

While these protocols differ in the details, they all follow the same basic paradigm.

- Alice and Bob agree on an element A ; it may be a global parameter, or it may be based on a seed provided by Alice
- Alice selects "small" elements S and E ; the value S (Alice doesn't actually need to remember the value of E) is Alice's private secret.
- Alice computes the value $B = AS + E$ (where the multiplication and addition are done using the ring primitives); this value is Alice's public key share, which she sends to Bob
- Bob also selects small elements S' and E' ; he computes the value $U = AS' + E'$ and the value $V' = BS'$.
- Bob then uses V' to compute an error-reconciliation vector C ; he sends U, C to Alice
- Alice computes the value $V = US$
- Both sides then use the error-reconciliation vector C to convert their V, V' into a shared secret, converting each coefficient of V, V' into one bit.

Some versions of the key agreement add additional error vectors at some places; as the attack can be modified to account for this, we will ignore it for now.

The idea behind this protocol is that Alice computes $V = SS'A + SE'$, while Bob computes $V' = SSA' + S'E$, they differ by $SE' - S'E$, as S, S', E, E' are

small elements, this is (with high probability) also small (in that each coefficient is close to 0), and so each element of V is "close" to the corresponding element of V' .

Of course, while they are close, they aren't identical. If we used any fixed mapping of the coefficients of V, V' to the shared secrets would mean that we would have some probability that an element of V would be on one side of the boundary, while the same element of V' would be on the other.

That's where the error-reconciliation vector comes in. We logically split up the possible coefficient values into four quadrants; quadrant I ($[0, p/4)$), quadrant II ($[p/4, p/2)$), quadrant III ($[p/2, 3p/4)$), and quadrant IV ($[3p/4, p-1]$)

There is a bit within the error-reconciliation vector that corresponds to each coefficient; it determines whether quadrants I and II are considered the same (e.g. values there may map to a 0 bit, while values in quadrants III and IV would map to a 1 bit); or whether quadrants I and IV are considered the same (e.g. values there may map to a 0 bit, while values in quadrants II and III would map to a 1 bit).

The intention is that Bob would, for each coefficient value of V' , determine the value of the error-reconciliation bit that would give the largest possible leeway for errors. As long as the absolute value of any coefficient of $SE' - S'E$ is no more than $p/8$, it will always be possible for Alice and Bob to agree on the shared secret. By choosing the size of the small values for S, S', E, E' , we can make this happen with extremely high probability.

4 The Attack

Here is the scenerio that we will assume for the attack:

- Alice uses a ring-LWE key exchange protocol to establish secure connections
- Alice uses the same key share to communicate with both Bob and the attacker Eve
- Eve's goal is to recover the value S the corresponds to Alice's public key share (and thus be able to decrypt Alice's traffic)
- Eve can perform the ring-LWE exchange protocol with Alice multiple times (with Eve providing a fresh key share each time)
- Each time after Alice and Eve has performed the key exchange protocol, Alice will derive her shared secret; Eve when then be able to generate one guess to Alice's shared secret, and Alice will indicate whether that guess matches what she has or not.

This last step can be implemented in practice if the ring-LWE key exchange is used to generate symmetric keys that Alice and Eve would use to communicate.

What Eve can do is generate her symmetric keys based on her guess; if Alice is able to decrypt (and respond) based on those keys, then (with high probability) Eve’s guess of the shared secret was correct; if Alice rejects the exchange, then Eve’s guess was not correct.

4.1 Basic Oracle Query

To make a query, Eve mostly follows the protocol; she selects small S', E' values (albeit not randomly), she computes V' , and generates the error-reconciliation vector C honestly, except for one location.

She deliberately selects S', E' so that coefficient 0 of Alice’s computation of US is near 0 (that is, near the boundary of quadrants I and IV; actually any of the quadrant boundaries could be used). For coefficient 0, Eve sets that error-reconciliation bit to indicate that the values in the range $[0, p/2)$ are mapped to one bit value, while values in $[p/2, p - 1]$ are mapped to the other. As Eve is able to compute correctly all the other bits of the shared secret (as she is performing the rest of the protocol honestly), this gives her a way to test the sign of the value of that intermediate coefficient.

In particular, if we select an S' , and call $\delta = (SS'A)[0]$ (where the notation $F[i]$ specifies coefficient i -th of the ring element F), and we use $E' = (X)^{-i}$ (for some integer i , where (X) stands for the ring element with a 1 in coefficient 1, and 0 elsewhere), then a key share of $(jS'A + kE')$ (for small integer j, k) would allow us to determine the sign of $(jSS'A + kS \cdot (X)^{-i})[0] = \delta j + k \cdot S[i]$.

4.2 The actual attack

The first step in the attack for Eve is to find a lightweight value S' where $(SS'A)[0] = \pm 1$. She can do this by searching for values S' which consists of at most three coefficients are $[1, -1]$ and the rest 0, and for which $S'B[0]$ is a small value; as B is Alice’s key share (which Eve can learn by running one exchange), this computation can be done off-line. As $S'B = S'(AS + E) = SS'A + S'E$ where E is known to be small, such a S' has a nontrivial probability of meeting the criteria.

We can further refine potential S' values by probing how a S' value works with several i coefficients; by fixing an i value, and trying several j values, and for each such value, to a binary search for the k value such that $j\delta + kS[i]$ is one sign, and $j\delta + (k + 1)S[i]$ is the other; this gives us the approximate value of $S[i]/\delta$ value, and we can use this to deduce whether $\delta = \pm 1$. We can also deduce the sign of δ .

Once we have such an S' value, then the attack is easy. We can set $k = \delta$, and then for any i , we can query the sign of $j + S[i]$, or in other words, whether $j \geq S[i]$. Binary search will quickly (with a handful of probes) give us the exact value of $S[i]$.

We query $S[i]$ for each i , that gives us the entire value of S , and the attack has succeeded.

The above can be done with perhaps 4,000 queries (assuming a ring size of $N = 1024$ and assuming that the value of S was generated using a discrete Gaussian Distribution with a standard deviation of circa 3).

5 Attack variants

One issue is that the [3] variant of the protocol has Alice add a second error vector to the computed V vector before doing the reconciliation; this would add in an error that Eve cannot control. Eve can compensate for this by either running multiple probes (and averaging out the error), or by increasing the j, k values (to attempt to magnify the signal over the fixed noise level).

In addition, the test (as written) assumes that Eve gets only a single bit per probe (that is, she can test whether her guess of the shared secret was accurate or not). If Alice sends the first encrypted message, then it is possible that Eve might probe several bits per attempt. That is, she might arrange to have several coefficients be near a Quadrant border, and she would be able to compute the shared secrets for each setting of the bits under test, and see which version matches the encrypted data she sees from Alice).

6 Conclusions and Recommendations

The above shows how ring-LWE based key exchange can be broken practically if the same key share is reused. Ring-LWE is still believed to be safe when a fresh key share is used every time; however one needs to be careful that is the situation.

One place where this can potentially come up is in the current TLS 1.3 draft[6]. In this draft, they allow a server to declare a 'static keyshare'. A client who wants to reestablish a connection with the server is able to send a message which includes both the client's key share and a message encrypted by a key derived from the server's static keyshare and the clients key share; this is called 0-RTT. The TLS 1.3 draft uses either DH or ECDH, which are both safe when used in this manner. However, if one were to replace the DH or ECDH with a ring-LWE based key exchange, this would become insecure.

These results have been specific to ring-LWE, however it would appear likely that these results would also extend to similar LWE-based protocols.

References

- [1] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, nov 1976.
- [2] Thomas Poppelmann Peter Schwabe Erdem Alkim, Leo Ducas. Post-quantum key exchange - a new hope. Eprint, 2015.

- [3] Xiaodong Lin Jintai Ding, Xiang Xie. A Simple Provable Secure Key Exchange Scheme Based on the Learning with Errors Problem. Eprint, 2012.
- [4] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. *Journal of the ACM*, 60(6):1–35, nov 2013.
- [5] Chris Peikert. Lattice Cryptography for the Internet. In *Post-Quantum Cryptography*, pages 197–219. Springer Science Business Media, 2014.
- [6] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 (rev 11). IETF, 2015.
- [7] Arjun Chopra Vikram Singh. Even More Practical Key Exchanges for the Internet using Lattice Cryptography. Eprint, 2015.