Failure is not an Option: Standardization Issues for Post-Quantum Key Agreement

Daniel Kirkwood, Bradley C. Lackey,
John McVey, Mark Motley,
Jerome A. Solinas,
David Tuller

National Security Agency

Introduction

- Key agreement is one of the fundamental cryptographic primitives in public-key cryptographic standards.
- Post-quantum key agreement will be required as part of the process of upgrading standards to provide quantum resistance.

Diffie-Hellman Key Agreement

Key agreement is most commonly performed using a protocol built on the Diffie-Hellman primitive.

For the purpose of discussion we will take a Diffie-Hellman primitive to consist of:

1. three finite sets – $\mathcal{S}$ (private keys), $\mathcal{T}$ (public keys), and $\mathcal{K}$ (session keys)
2. two functions $F : \mathcal{S} \longrightarrow \mathcal{T}$ and $G : \mathcal{S} \times \mathcal{T} \longrightarrow \mathcal{K}$
3. a distribution χ on $\mathcal{S}$.

The functions F and G must satisfy the conditions that, for $a, b \leftarrow \chi$

1. $G(a, F(b)) = G(b, F(a))$ with high probability.
2. It is computationally infeasible to recover $G(a, F(b))$ given $F(a)$ and $F(b)$.

Post-Quantum Diffie-Hellman Key Agreement

Several post-quantum key agreement schemes analogous to Diffie-Hellman have been proposed.

They fall into two families:

- Isogeny-based key agreement, e.g. [Rostovtsev and Stolbunov, 2006].
- Lattice-based key agreement, e.g. [Ding, 2012, Peikert, 2014].

Key Leakage

A recurring problem with public-key based encryption and key agreement is that of *key leakage:* algorithm failure can reveal some information about the recipient's private key.

Example: elliptic curve cryptography. The "point-off-the-curve" attack of [Biehl, et al., 2000]

This particular attack is prevented by deploying **public key validation** as a part of the protocol. A test is performed on the public key to verify that it has the proper form – in this example one checks that the public key is a point on the specified curve having the correct order.

Key Leakage

Post-quantum example: [Howgrave-Graham, et al., 2003] present an attack on NTRUEncrypt.

Decryption failures occur when the coefficients of a certain integer vector span an unusually large range. This causes an information leak.

To attack Bob, Alice prepares ciphertexts having this form. She learns information about Bob's key based on whether or not the decryption of these ciphertexts succeeds or fails.

If Bob is reusing his key pair, Alice eventually collects enough information to recover his private key.

Reuse of Key Pairs

In a number of standard protocols, re-use of public-private key pairs can occur.

For example:

- TLS, when either the client or the server uses a static key.
- IKE v2, where re-use of ephemeral public keys is permitted.

Remedies

The NTRUEncrypt problem is solved by requiring messages to be formatted in a particular way before encryption.

Key agreements share the same key leakage problem in the event of agreement failure.

The way to solve the problem in this case is to employ public key validation, as is done in the case of elliptic curves.

(Of course, the algorithm must be designed so that failures are sufficiently rare when valid public keys are used.)

Public Key Validation

With classical public-key cryptography, a public key can be validated directly – i.e. by performing a check on the public key itself.

For example, in ECDH the received elliptic curve point is checked to verify that it is actually a point on the specified curve having the correct order.

Public Key Validation

Unfortunately, such direct public key validation is not always possible for lattice-based and isogeny-based schemes.

Indeed, for either of these schemes, the security of the algorithm depends on public keys being indistinguishable from random.

Therefore some other way must be found to prevent forced-failure attacks other than direct public key validation.

Options

There are a number of steps one could take.

- Restrict the number of times an ephemeral key can be reused (rather than restricting to a certain period of time).

  The bound may depend on the specific use case (broadcast for example) as well as the rate of leakage.

- In the static case, use an encryption algorithm (e.g. modified NTRUEncrypt) for which key leakage is not an issue.

- Users who reuse their keys could perform indirect validation (explained below) of the other user's ephemeral public key.

Indirect Public Key Validation

One way to validate public keys indirectly is to use the mechanism described in [Fujisaki and Okamoto, 1999] (as was done in [Peikert, 2014]).

This mechanism was devised as a way to combine an asymmetric encryption algorithm and a symmetric encryption algorithm that are each secure in a weak sense to form a hybrid encryption algorithm that is secure in a strong sense.

In the context of a key agreement, the following variant of the mechanism provides an indirect method of public key validation.

Indirect Public Key Validation

Let:

- Enc_k and Dec_k denote the encryption and decryption functions of a secure symmetric encryption algorithm keyed with k.
- PKDF() denote a suitable one-way function from bitstrings to private keys.

  All ephemeral keys are required to be generated as outputs of PKDF().
- KDF be a key derivation function based on a cryptographic hash function.

Indirect Public Key Validation

1. Alice obtains Bob's reusable public key K_B.
2. Alice chooses a random seed r_A and computes $k_A = \text{PKDF}(r_A)$ and the corresponding public key K_A.
3. Alice derives shared secret value SSV from K_B, k_A and computes session key and validation key via $SK \| VK = \text{KDF}(SSV)$.
4. Alice sends K_A and $c_A = \text{Enc}_{VK}(r_A \oplus SK)$ to Bob.
5. From k_B, K_A, Bob derives SSV', then SK' and VK'.
6. Bob computes $r'_A = \text{Dec}_{VK'}(c_A) \oplus SK'$. If the public key corresponding to $\text{PKDF}(r'_A)$ is K_A, then Bob uses SK' to communicate with Alice, else he returns a failure message.

Indirect Public Key Validation

- When Alice's public key is malformed, Bob returns the same failure message *regardless of why the key agreement fails*. Therefore she receives no information from Bob about whether or not the original key agreement would have succeeded when using this malformed key.
- Alice cannot even perform exchanges using valid public keys *of her choice*. She only gets to choose the seed for the one way function PKDF().

Potential Concerns

- Although this validation procedure effectively eliminates the problem of malformed keys, the resulting key agreement differs somewhat from classical Diffie-Hellman.
- One can conceive of protocols which become insecure when used with a key agreement requiring indirect public key validation.
- For example, consider an authenticated key agreement in which the ephemeral participant authenticates by signing his public key. If he then uses indirect public key validation, he could be vulnerable to a replay attack by the other party.

Potential Concerns

When key leakage is present and direct public key validation is unavailable:

- If one party reuses his key, he must validate (indirectly) the other party's public key.
- Since this exposes the other party's private key, the other party cannot reuse it at all.
- Therefore **the users cannot both reuse their keys.**
- In particular, there can be no static-static key agreement using this approach.

Further Questions

- Can the security proof for the Fujisaki-Okamoto transformation be easily extended to the variant for key agreement?
- Is there a better solution to the key validation problem?
- What potential problems are introduced by the use of indirect public key validation?
- Is there some way to perform static-static key agreement securely?

Conclusion

- Post-quantum key agreement will be needed.
- Using a Diffie-Hellman analogue with reusable keys introduces potential vulnerabilities when public keys cannot be validated directly.
- There are simple means by which public key validation can be performed indirectly.
- Using indirect public key validation means that the key agreement is not an exact analogue of Diffie-Hellman.

I. Biehl, B. Meyer, and V. Muller
Differential Fault Attacks on Elliptic Curve Cryptosystems.
CRYPTO 2000, LNCS 1880, 2000

J. Ding.
A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem
Cryptology ePrint Archive no. 688, 2012

E. Fujisaki and T. Okamoto
Secure Integration of Asymmetric and Symmetric Encryption Schemes.
CRYPTO '99, LNCS 1666, 1999

C. Peikert
Lattice Cryptography for the Internet
Post Quantum Cryptography, LNCS 8772, 2014

N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J. Silverman, and A. Singer
The Impact of Decryption Failures on the Security of NTRU Encryption.
CRYPTO 2003, LNCS 2729, 2003

A. Rostovtsev and A. Stolbunov
Public Key Cryptosystem Based on Isogenies.
Cryptology ePrint Archive no. 145, 2006