Efficient quantum-resistant trust Infrastructure based on HIMMO

Oscar Garcia-Morchon, Ronald Rietman, Ludo Tolhuizen,
Jose-Luis Torre-Arce, Sauvik Bhattacharya and Maarten Bodlaender

Philips Group Innovation, Research, Eindhoven, The Netherlands

Abstract. Secure Internet communications face conflicting demands: advances in (quantum) computers require stronger, quantum-resistant algorithms, while at the same time the Internet of Things demands better-performing protocols; and finally, communication links usually depend on a single root-of-trust, e.g., a certification authority, a single point-of-failure that is too big of a risk for future systems.
This paper proposes a hybrid infrastructure that combines a quantum-resistant HIMMO key pre-distribution scheme based on multiple Trusted Third Parties with public-key cryptography to address these problems. During operation, any pair of devices can use HIMMO key material and public-keys to establish a secure link, and public-keys are then certified by multiple TTPs. The solution is resilient to the capture of individual roots of trust, without affecting performance, while public keys can provide features such as forward secrecy. Combining HIMMO identities with public-keys enables secure certification of public keys and distribution of HIMMO key material from multiple TTPs, without requiring an out-of-band channel. The infrastructure can be tuned to fit Internet of Things scenarios benefiting from an efficient, non-interactive and authenticated key exchange, or to fit use cases where the use of multiple TTPs provides privacy safe-guards when lawful interception is required.
As proof-of-concept we show how TLS can benefit from these ideas with minimal extensions, while exhibiting good security features and performance compared with solutions based on public-key cryptography only.

Keywords Post-Quantum Cryptography, HIMMO, TLS, Security Architecture.

1 Introduction

Secure Internet communications are a must nowadays. Otherwise, services such as doing our tax declaration online, communicating with friends, or checking our online banking account would not be feasible. However, the existing security infrastructure is under stress due to several reasons. First, as soon as a large enough quantum computer is built, Shor's algorithm can be used to break all public-key algorithms currently used in TLS. This threat has led to the recent NIST and ETSI announcement of future standardization plans of post-quantum cryptography [6]. However, most existing candidates are not a direct drop-in replacement due to their large key sizes or signatures. For instance, the McEliece crypto system [21] would require 1 MByte public-key, and XMSS [3] involves signatures of tens of KBytes. Second, the manipulation of Certification Authorities, our Roots of Trust, in the existing Public-Key Infrastructure can have catastrophic consequences, as with the hack of Diginotar Certificate Authority [22] in 2011 or the alleged compromise of Gemalto's key server [16]. A third aspect aspect to consider is that the scope of the Internet is increasing and millions of smart objects are getting connected. These smart objects have limited resources, in particular, they communicate through resource-constrained networks and have a low energy budget and low computational power. Finally, secure communications can be used for good purposes, but there are also situations in which they are misused for performing crime.

Thus, the problem statement of this paper focuses on four challenges: (i) ensuring a secure world requires security primitives and a security architecture that can be resistant to quantum-computers, and (ii) still be practical, in particular taking into account new application areas such as the Internet of Things. Furthermore, (iii) the architecture should be resilient to the capture of Roots of Trust in order to ensure the confidentiality, integrity, and authenticity of our communications. At the same time, (iv) the solution should be able to accommodate trade-offs between the right of privacy in our communications and the lawful interception of communication required in some circumstances.

This paper contributes a hybrid architecture combining the best features of traditional public-key cryptography and the HIMMO key pre-distribution scheme [12][15] to address these problems. In our architecture, long-term public keys and parameters of the devices are certified by means of HIMMO so that any pair of devices can efficiently verify them by means of symmetric-cryptography. This addresses the need of a quantum-resistant and efficient solution. At the same time, the usage of public keys ensures advanced security properties such as forward-secrecy and also allows for the secure initial distribution of HIMMO key material in-band so that communication remains confidential even in the presence of Trusted Third Parties (TTPs). Public-keys can be certified by multiple TTPs, the roots of trust in our architecture, without affecting performance of the devices, so that our solution is resilient to the capture of some TTPs and still efficient. The architecture can be configured to fit use cases in which operational performance is critical by using only HIMMO to provide a quasi non-interactive authenticated key exchange. Finally, the usage of multiple TTPs can also provide a proper trade-off between the right on privacy and the need, in some occasions, for lawful interception of communication.

After summarizing background work in Section 2, we come to the main contributions of this paper. Section 3 details security needs at system and protocol level, limitations, and goals for our work. Section 4 presents our hybrid architecture combining HIMMO with public-key cryptography, and accompanying protocols. Section 5 details our proof-of-concept based on TLS in which HIMMO is used to certify ECC and NTRU public-keys. Section 6 provides an overall evaluation of the proposed solution. Section 7 concludes this paper and points out future research.

2 Background

There are different types of **security architectures**.

A public-key infrastructure (PKI) is built on a number of Certification Authorities (CAs) in charge of signing the public-keys of the communicating parties. In such an architecture Alice has a public/private key pair (Pk_A, Sk_A), and her public key Pk_A is signed by a CA. When Alice presents her public-key Pk_A and its certificate C_A to Bob, Bob can verify Alice's certificate, and therefore, her public-key if it was signed by a common CA. PKI scales well in many use cases and it is widely used to protect the Internet.

Security architectures based on Kerberos [19] rely on a Trusted Third Party (TTP) called Key Distribution Center (KDC). The KDC enables a secure and authenticated communication between Alice and Bob. When Alice wants to communicate with Bob, Alice will contact the KDC that will provide her with a ticket enabling secure communication with Bob. Kerberos is a widely used solution, e.g., in operating systems to enable the authentication of users and services. The main challenges with Kerberos are that the KDC is required during normal operation, and that it is able to monitor all communications.

Architectures based on a Key Pre-Distribution Scheme (KPS) also rely on a TTP. The TTP owns some secret root key material R. When the TTP enrolls a party, e.g., Alice, the TTP provides her with some key material that depends on R and her identity. Once a pair of entities have been enrolled, they can establish a common symmetric-key based on their respective key material and identities in a stand-alone manner. This shared symmetric-key can be used for achieving mutual authentication or for securely exchanging data. The main challenge of KPS since the introduction of the concept in 1987 by Matsumoto and Imai [20] has been the design of a KPS that is operationally efficient and remains secure even if many devices have been compromised.

The advent of quantum-computers has motivated the NIST announcement [6] of standardization of **quantum-resistant algorithms** that should substitute the currently standardized public-key primitives, namely RSA and ECC, used in PKI. There are four main types of quantum-resistant public-key algorithms: lattice-based, code-based, multivariate polynomials, and hash-based. Examples of these classes of algorithms are: NewHope [1], McEliece [21], Rainbow [7] or XMSS [3]. However, none of the existing proposals seems to be a drop-in replacement for existing algorithms in all use cases due to the lack of security analysis or low performance.

Recently, the HIMMO scheme has been introduced as an efficient and collusion-resistant KPS [12],[14],[15]. In its basic version, there is a single TTP owning some root key material $R(x, y)$. During enrollment, the TTP provides Alice with a function $G_A(x)$ in a secure way. During operation, Alice and Bob can agree on a common key $K_{A,B}$ since $G_A(B) \approx G_B(A)$. HIMMO's cryptanalysis relies on lattice techniques, and therefore, it is considered a potential quantum-resistant candidate and although it is not a public-key scheme, it enables some very interesting features such as key generation $K_{A,B} = G_A(B)$. HIMMO also supports multiple TTPs. This means that an entity can enroll with multiple TTPs, receive key material from each of them, and obtain its final keying material as the combination of key material received from each TTP. In this way, each of the TTPs only has a partial knowledge of the key material of the party. Building on the key generation step, if two parties engage on a mutual authentication handshake and this handshake is successful, then both parties have successfully verified each other's identities. If the identities are computed as the hash function of a set of parameters, then all those parameters can be easily verified. Appendix A provides further details about HIMMO.

The Transport Layer Security (TLS) protocol is used today to protect most of the communications in the Internet. TLS runs between a client and a server and involves the exchange of a number of messages during an initial handshake. In this handshake, client and server exchange the supported ciphersuites and engage in a protocol, usually based on public-key cryptography, that enables key agreement, (mutual) authentication, or forward secrecy. Due to the advent of quantum-computers, all available ciphersuites will be broken with the exception of the one based on pre-shared keys [9]. In order to prepare for this, there have been publications in which quantum-resistant candidates have been integrated with TLS, including TLS-LWE [2] or DTLS-HIMMO [14].

In the IETF there are two initiatives to make TLS and IKEv2 more quantum-resistant. The hybrid TLS handshake [23] extends TLS in a modular way with quantum-resistant Key Encryption Schemes, such as NTRUEncrypt [18]. In this hybrid handshake, key agreement is performed by means of a classical key establishment scheme, e.g., ECDH, and a post-quantum scheme, e.g., NTRUEncrypt. The goal is that the final secret used to protect the actual data is derived from both schemes, the classical one and the post-quantum one, so that an attacker that is recording message exchanges now cannot later decrypt them once a quantum computer is available. This hybrid handshake does not include quantum-resistant signatures due to several reasons. One of them is that it is less urgent than the key establishment. Another reason is that there are no good candidates exhibiting a good security confidence and performance. The very recent Internet Draft [11] aims at making IKEv2 quantum-resistant by using pre-shared keys. Here, the authors propose the usage of pre-shared keys next to a standard key exchange in the Internet Key Exchange protocol IKEv2, part of IPSec. Their motivation is to make sure that the communication between an initiator and a responder remains confidential and is authenticated. To this end, they propose an optional key exchange and mutual authentication handshake based on pre-shared keys. This optional key exchange would happen next to the normal key exchange in IKEv2 providing post-quantum security. However, this draft does not cover how the pre-shared keys can be pre-distributed in an efficient way.

3 Design Goals and Problem Statement

3.1 Design goals

In the advent of a post-quantum world, we consider an **attack model** in which attackers (not necessary the same one):

- can own and run a quantum-computer and break classical public-key cryptography algorithms with a quantum computer.
- are able to compromise or weaken algorithms with classical attacks without others knowing it [5].
- can harvest and store exchanged messages over the Internet during a pre-quantum era and decrypt them once a quantum computer is available.
- can monitor communication links.
- are able to modify communications between a pair of parties.
- are able to compromise roots of trust such as CAs or TTPs, e.g., [16] or [22].
- are able to compromise end devices.

With these capabilities, the goals of the attacker can be to just eavesdrop (passive attacker) on a specific communication link, or on all of them; but also to impersonate (active attacker) a person or even send fake messages on her behalf. In order to prevent such attackers from achieving their goals and fulfill regulations, we would like a security architecture with the following **features**:

- Confidentiality, authenticity, and integrity of communication links.
- Identification and authentication of communicating parties.
- The architecture should resist the capture of some roots of trust.
- Protocols should be quantum-resistant.
- Communication parties should be able to have advanced security features such as forward secrecy.

Certain regulations require the capability of lawful interception of communications [10] to prevent or resolve crimes. The architecture should therefore allow for a proper trade-off between the right on privacy and the need for lawful interception of communication.

In addition to above security goals, it is of key importance to also consider **operational and performance goals**:

- The architecture should facilitate the transition towards quantum resistant solutions with minimal changes and overhead.
- The architecture should be widely usable to ensure that as many use cases and applications as possible are secured, e.g., be applicable not only to traditional Internet applications but also to new ones, e.g., Internet of Things.
- The design should ensure above security goals, but also aim at achieving as good a performance as possible in terms of bandwidth, computation cost, energy, and memory.
- The protocols should be modular to easily add multiple ciphersuites, also quantum-resistant ones for both key exchange and digital signatures.

3.2 Problem Statement

When we consider the above design goals in Section 3.1 and the background reviewed in Section 2, the problem statement of this paper is fourfold.

First, there are many potential quantum-resistant candidates, however, none of them seems to be a drop-in replacement of existing primitives. In particular, signatures, are relatively bulky or are operationally cumbersome to use. For instance, hash-based signatures involve an overhead of tens of KBs and in some settings require keeping some state [4].

Second, communication links in Internet protocols usually depend on the trust put on a single root of trust, e.g., a certification authority, that if manipulated or misbehaving, gives an open door to modify or even monitor all digital communications. Here the question is how to share that trust responsibility among multiple roots of trust and in this way, be resilient to the capture of Roots of Trust ensuring the privacy of our communications.

Third, the right of privacy and the lawful interception of communication are contradictory goals and the question is how to accommodate both goals in a common architecture so that users have the confidence that their data is not monitored or manipulated, but if required and agreed, access can be enabled.

Finally, existing protocols and cryptographic primitives are already considered to be too heavy for the Internet of Things, in particular, concerning bandwidth consumption, energy requirements, and strict timing needs. Therefore, it is a challenge to find an overall solution that fulfills all requirements and also exhibits good performance.

4 Design

This section describes our efficient and flexible hybrid trust infrastructure and its rationale. The basic idea behind this architecture is simple:

- Use HIMMO TTPs as roots of trust to enable efficient certification and verification of the public-keys and other credentials of devices in one-to-one communication scenarios.
- Use the public-keys to enable the secure distribution of HIMMO key material and advanced security features like forward secrecy.

The architecture can work with any type of public-keys and any KPS. We use HIMMO as it is – to the best of our knowledge – the only KPS that enables quantum resistant and collusion-resistant key agreement in an efficient way. This approach combines the best security features of both worlds, such as efficiency, offline operation, secure initialization, low overhead, forward secrecy, and support for multiple roots of trust. In the following we discuss three aspects: (i) roots of trust, (ii) enrollment and certification, and (iii) operation.

4.1 Roots of Trust in a PQ World

A fully general architecture can consider (i) roots of trust supporting traditional public-key infrastructure able to sign digital certificates containing public-keys and (ii) TTP solutions, either requiring an online interaction to enable a pair of parties to securely communicate with each other (similar to Kerberos) or with a TTP only involved during the registration of a party (such as with HIMMO).

In this paper, our architecture relies on roots of trust based on HIMMO TTPs due to several reasons. First, its TTP is not involved during operation. Second, multiple roots of trust are also supported in an efficient way. Third, the operational features of HIMMO fit in many IoT use cases: low bandwidth consumption, fast operation, low RAM needs. Finally, there is a lack of good quantum-safe PK-based signature solutions.

Therefore, our trust infrastructure relies on a number of HIMMO TTPs that keep secret root key material for some given security parameters. TTPs are managed by one or several independent organizations at at a single or multiple locations so that a security breach in one of them does not affect the rest. The TTPs are in charge of certifying parameters of entities that wish to enroll and distributing key material to them. Thus, a single root of trust, TTP, is not able to monitor or modify communications; but only a coalition of TTPs could do it if required for lawful interception of communications.

4.2 Enrollment and Certification

In our architecture, we consider an enrollment and certification phase during which a party can talk to any set of HIMMO roots of trust, expose its identification information and public-key, get them certified, and securely get HIMMO key material. This is similar to the certification of public-keys in today's PKI or the distribution of key material in existing KPS. Note that during the certification of a public-key in PKI, the public-key is signed by the CA and the certificate can be sent back in the clear. In a KPS, however, it is of key importance to ensure the confidentiality of the key material.

While for IoT scenarios, an out-of-band channel is a good option [13], this is not the case for the Internet. The challenge is how to distribute HIMMO key material in a secure way. To this end, this paper proposes the following general enrollment approach: When an entity, Alice, wants to get enrolled in the system, she generates a pair of public/private keys (ideally quantum-resistant), and combines the public-key with any other identification information. Then Alice enrolls with one or several HIMMO TTPs. To this end, she must send to each TTP this information, that must contain at least her public-key. When the TTP receives Alice's information, it can further verify it for consistency and add additional parameters, e.g., an expiration date, creating a certificate of Alice's identity. Given this certificate, the TTP computes a HIMMO identity by hashing the certificate and obtains the corresponding HIMMO key material. Therefore, the HIMMO key material is bound to the received public key. Now the TTP can use the same received public-key to encrypt the computed HIMMO key material so that only the owner of this public-key can decrypt the key material. In case of multiple TTPs, some communication between them may be required to ensure that they all generate the same HIMMO identity for Alice.

A very specific instantiation of the above general concept is depicted in Figure 1 in which we show the enrollment with a single TTP. In Step (1), Alice sends her public-key and a nonce. In Step (2), the TTP computes the HIMMO key material whose identifier is the hash of the public-key. In Step (3), the TTP sends a message to Alice containing the HIMMO key material and the hash of it concatenated with the received nonce (that acts as an authentication token), everything encrypted with Alice's public-key. In Step (4), Alice can decrypt her HIMMO key material that will serve to verify her public-key and checks the received authentication token to verify the integrity and freshness of the message.

We note that the final HIMMO key material associated to Alice can include the HIMMO key material received from multiple TTPs. This information can be stored without being combined, or combined by adding the coefficients of the HIMMO key material one by one.

Fig. 1. Enrollment and Certification

Alice		**TTP**
	$\xrightarrow{1.Pk_a,N_a}$	
		$2.G_{Pk_a}(x) = R(hash(Pk_a),x)$
	$\xleftarrow{3.E_{Pk_a}(G_{Pk_a}(x)\vert hash(G_{Pk_a}(x)\vert N_a))}$	
$4.Pk_a, G_{Pk_a}(x)$		

4.3 Secure Operation

Now the question is how Alice and Bob can establish a secure communication in the above setting, i.e., each of them has public-keys bound to HIMMO key material generated by a number of TTPs. We believe that many security protocols can be designed, in general, following the following general steps.

- Alice exposes her supported TTPs to Bob.
- Bob decides whether Alice's supports at least a minimum number of TTPs that he also trusts. If not, Bob aborts. Otherwise, Bob exchanges these TTPs identifiers with Alice so that she can also verify whether that set of TTPs provides her with enough trust.
- Alice and Bob engage in a handshake for key exchange and mutual authentication that relies on public-key cryptography to achieve strong properties such as forward secrecy and HIMMO to ensure an authenticated channel and to verify the identities of the communicating parties.

– Upon the establishment of a secure channel, Alice and Bob exchange data in a secure way.

Note that the verification of the public-keys happens without requiring digital signatures, but HIMMO. In this case, the HIMMO key material of a party is cryptographically bound to its identity. Therefore, if the exposed public-keys are the right ones, HIMMO key generation at both sides will happen in a proper way and both parties will automatically verify their public-keys if an authentication handshake based on the generated HIMMO keys is successful. Since HIMMO supports multiple TTPs, this verification can simultaneously be done with key material generated by multiple TTPs, so that even if one of the TTPs is compromised, public-keys cannot be impersonated. Most importantly this is achieved with no additional computational or communication cost during the operational phase. The fact that asymmetric-cryptography is used for key exchange ensures that TTPs cannot access the communication and enables forward secrecy for advanced uses cases, e.g., Internet communication. Since the solution is TTP-based, this architecture can also be applied to scenarios requiring lower communication/computational overhead, e.g., Internet of Things, or potentially requiring lawful interception of communications but still requiring privacy assurances by relying on multiple TTPs. The usage of HIMMO ensures that the authentication of peers and the resulting communication link are quantum-resistant.

The previous general operational protocol is exemplified in a very specific operational handshake for key exchange and mutual authentication depicted in Figure 2. In this scheme, a single TTP is supported. In Step (1), Alice sends her public key and a nonce to Bob. In Step (2), Bob computes the HIMMO key using HIMMO and a Diffie-Hellman key using public-key cryptography. Bob then securely combines both keys and uses the resulting key to compute a message authentication code (MAC) based on the HIMMO key, the received nonce and another nonce locally generated. Next in Step (3), Bob replies to Alice exchanging his public key, his own nonce, and the computed MAC so that in Step (4) Alice can perform equivalent steps at her side and verify the received MAC, and therefore, verify Bob's public key and that Bob computed the same key. Finally, in Step (5) Alice sends her MAC to Bob so that he can verify Alice's public-key and the generated key in Step (6). Next, Alice and Bob can proceed to communicate in a secure way. This handshake is just an example: ephemeral public-keys can be exchanged for forward secrecy or the key exchange can be based on a key encapsulation mechanism as we show in our proof-of-concept.

5 Proof-of-Concept with TLS: TLS-HIMMO

As a proof of concept, we illustrate the above generic architecture by instantiating it in the context of the TLS protocol as it is used in the Internet today to protect most of the Internet applications. We use HIMMO in our proof of concept since it enables for the efficient distribution of pairwise keys, enabling implicit verification of credentials, and supporting multiple TTPs. Finally, we

Fig. 2. Secure Operation

Alice		**Bob**
	$\xrightarrow{1. Pk_a, N_a}$	
		$2. K_{b,a}^{HIMMO} = G_b(Pk_a)$ $K_{b,a}^{DH} = Pk_a \cdot Sk_b$ $K_{b,a} = KDF(K_{b,a}^{HIMMO}, K_{b,a}^{DH})$ $MAC_{b,a} = MAC(K_{b,a} \vert N_b \vert N_a)$
	$\xleftarrow{3. Pk_b, N_b, MAC_{b,a}}$	
$4. K_{a,b}^{HIMMO} = G_a(Pk_b)$ *Verify* $MAC_{b,a}$ $K_{a,b}^{DH} = Pk_b \cdot Sk_a$ $K_{a,b} = KDF(K_{a,b}^{HIMMO}, K_{a,b}^{DH})$ $MAC_{a,b} = MAC(K_{a,b} \vert N_a \vert N_b)$		
	$\xrightarrow{5. MAC_{a,b}}$	
		6. Verify $MAC_{a,b}$

further use ECDH and ECDSA as the pre-quantum cryptographic primitives for key exchange and digital signatures and NTRUEncrypt [18] is taken as a post-quantum candidate for public-key encryption.

Figure 3 shows the complete TLS handshake between a client, Alice, and a server, Bob when we apply the design explained in Section 4. The required additions to support this operational handshake in TLS are not many, and the main one refers to the capability of agreeing on a common set of roots of trust, TTPs, and agreeing on this type of their use. For this, we consider RFC 6066 [8] since it defines extensions to the *ClientHello* and *ServerHello* messages in which the supported certification authorities are exchanged. The current motivation in RFC 6066 is to: *prevent multiple handshake failures involving TLS clients that are only able to store a small number of CA root keys due to memory limitations.* We reuse this extension for our purposes as follows:

If a client and server have registered with some HIMMO TTPs and obtained HIMMO key material as described in Section 4, then they will exchange their supported TTPs. Thus, in Step (1) of the handshake, the client Alice creates a *ClientHello* message including its supported roots of trust, i.e., HIMMO TTPs or CAs, as described in RFC 6066 [8], that is then sent to the server Bob in Step (2). In step (3), Bob searches for a common set of HIMMO TTPs that it then includes in the *ServerHello* message and sends back to Alice in Step (4). If they agree on the usage of HIMMO TTPs, then this means that they will exchange certificates or public-keys whose hashes are HIMMO identifiers for which they have key material. If we consider that only classical ciphersuites are supported, then we consider classical certificates, e.g., ECDSA, that are exchanged in a standard way by means of the *ClientCertificate* and *ServerCertificate* messages. If we consider that post-quantum keys are supported, then we consider the specification of the hybrid TLS handshake Internet Draft [23] since it describes an extension header in the *ClientHello* message in which additional public-keys are exchanged. The exchanged certificates or public keys and their key material associated to common TTPs allow Alice and Bob to independently generate pairwise HIMMO keys in Steps (5) and (9) (one key for each common TTP agreed upon), which they then independently derive the *final* HIMMO key $K^{\text{HIMMO}}$by means of a pseudo-random function of all previously generated HIMMO keys. Note that HIMMO requires a key reconciliation step in which HelperData is exchanged from the client and server. In our implementation, we exchange this information in the *ClientKeyExchange* and is the only protocol field that would require standardization.

Now, HIMMO is used to verify that the exchanged certificates and public-keys are valid and to increase security level by computing the Master Secret used for protecting the TLS record layer not only from a single Pre-master secret *PMS*, but from the classical key and the HIMMO key. This is similar to [23] and [11]. Therefore, if the HIMMO key K^{HIMMO} is one of the inputs to the Master Secret in Steps (10) and (11) via the Pre-master Secret *PMS*, then K^{HIMMO} will ensure the confidentiality and authentication of the messages exchanged in the record layer, but most importantly, it will also implicitly verify the certificates, and therefore, the public-keys, of client and server since the Master Secret is used in the computation of the *Finished* message. This is because the TLS handshake requires Alice and Bob to calculate a message authentication code (MAC) over the entire handshake using the Master Secret, which they then exchange via the *Finished* messages and verify, in steps (13) and (15). This MAC is thus bound to the K^{HIMMO} that both parties computed using the certificates of the other party and the own HIMMO key material. This verification by Bob and Alice will fail unless they computed the right K^{HIMMO} using an authentic certificate/public key of the other party and proper key material.

Finally, note that the above TLS extension uses HIMMO to verify the certificates and public-keys of client and server enabling an efficient way for certificate verification and key establishment supported by multiple TTPs while achieving advanced features such as forward secrecy or non-repudiation. Lawful interception use cases would require excluding the exchange of public-keys during operation and relying on multiple TTPs only to provide a proper privacy trade-off. Note that the TLS extension presented in [14] also enables key establishment and mutual authentication based on multiple HIMMO TTPs.

6 Evaluation

6.1 Development environment and performance

To validate our research, we implement the protocol described in Section 5 by extending the CyaSSL library and using TLSv2. The TLS protocol is executed

Fig. 3. TLS Handshake including the design proposed in this paper

Alice		**Bob**
1. ClientHello = $\text{TLSversion} + \text{Random}_\text{a} + \text{SessionID}_\text{a}$ $+\text{Ciphersuites}_\text{a} + \text{CompressionMethods}_\text{a}$ $+\text{ExtType} + \text{CAList}_\text{a} + \text{TTPList}_\text{a}$.[1]		
	$\xrightarrow{\text{2. ClientHello}}$	
		3. ServerHello = $\text{TLSversion} + \text{Random}_\text{b} + \text{SessionID}_\text{b}$ $+\text{Ciphersuites}_\text{b} + \text{CompressionMethods}_\text{b}$ $+\text{ExtType} + \text{CAList}_\text{b} + \text{TTPList}_\text{b}$.
	$\xleftarrow{\text{4. ServerHello}}$	
	$\vdots$	
5. $\{K_{a,b}^{HIMMO}, \text{HelperData}\} = G_a^1(Pk_b) + G_a^2(Pk_b) + \ldots$.		
6. ClientKeyExchange = $\{E_{\text{ClassicPk}_\text{b}}(\text{ClassicS}) \mid E_{\text{QshPk}_\text{b}}(\text{CliS})$ $\mid \text{HelperData}\}$.[2]		
	$\xrightarrow{\text{7. ClientKeyExchange}}$	
		8. Recover ClassicS, CliS.
	$\vdots$	
		9. $K_{a,b}^{HIMMO} = G_b^1(Pk_a) + G_b^2(Pk_a) + \ldots + \text{HelperData}$.
10. PMS[3] = $\{\text{Classic} \mid \text{CliS} \mid K_{a,b}^{HIMMO}\}$.		10. PMS = $\{\text{Classic} \mid \text{CliS} \mid K_{a,b}^{HIMMO}\}$.
11. MasterSecret = $PRF(\text{PMS})$.		11. MasterSecret = $PRF(\text{PMS})$.
	$\xrightarrow{\text{12.Finished}_\text{a}}$	
		13. Verify Finished_a.
	$\xleftarrow{\text{14.Finished}_\text{b}}$	
15. Verify Finished_b.		

[1] List of supported HIMMO TTP and CA IDs according to RFC 6066.

[2] ClassicS, CliS are the classic secret and client QSH (assuming [23]) secret respectively.

[3] Pre-master Secret used to generate the Master Secret by means of the TLS pseudo-random function PRF.

between two machines within the same Local Area Network. The PC uses Windows 7 Enterprise, with an Intel Core i5-3437U @ 1.90 GHz.

Table 1 compares the performance of different TLS handshakes, using pre- and post-quantum ciphersuites. Pre-quantum ciphersuites use ECDHE-ECDSA while post-quantum ciphersuites rely on NTRU_RSA (NTRU is used to encrypt a secret that is exchanged; the NTRU public-key is signed with RSA). We compare them when we add HIMMO next to them using a single TTP. The HIMMO parameters b and B are chosen to be 32. Several HIMMO systems are used in parallel to obtain a 256 bit key. In this case, we observe that the additional time or bandwidth consumption is minimal while enabling the verification of public-keys and generation of a symmetric-secret. We also show the case in which the HIMMO key is computed from the HIMMO key material assigned by three TTPs, that could represent TTPs managed by different organizations or located at different places. In our implementation, key computation with multiple TTPs is sequential, i.e., it computes three HIMMO keys instead of a single combined key after adding the coefficients of the key material one by one first. Once this is done, we expect a timing similar to the usage of a single of TTP independently of the number of TTPs.

	Handshake time (ms)	Handshake size (Bytes)
ECDHE_ECDSA_WITH_AES_256_CBC_SHA	59	3547
ECDHE_ECDSA_WITH_AES_256_CBC_SHA_HIMMO256	62	3590
NTRU_RSA_WITH_AES_256_CBC_SHA	42	5651
NTRU_RSA_WITH_AES_256_CBC_SHA_HIMMO256	45	5694
NTRU_RSA_WITH_AES_256_CBC_SHA_HIMMO256_3_TTP	51	6260

Table 1: TLS performance for different ciphersuites (256-bit HIMMO key with a large security parameter $\alpha = 8000$).

To further evaluate the proposal, we give insights on the performance of HIMMO standalone. Table 2 shows the computation time of a HIMMO key, as well as the size of the necessary key material, on an Intel Core i5-3437U @ 1.90 GHz. It is worth noting that an increase of the security parameter α to a very high value, that directly determines the dimension of the lattice that needs to be reduced in order to find a close/short vector [15], does not have a huge impact in the CPU needs while bandwidth consumption remains practically constant and equal to a few bytes. Memory increases, but for application areas such as the Internet, a storage requirement of a couple of megabytes is completely affordable. We also remark that for Internet of Things applications there are different performance criteria: communication bandwidth, energy consumption, timing, and memory. For all these parameters HIMMO is excellent. By construction, HIMMO has low communication needs since it is identity-based. This is also the reason why it requires little energy, since wireless communication consumes most of the energy budget. Timing is also good. Key material size is on the high side.Even if this can be optimized, memory is the least relevant criteria: IoT devices are getting bigger memories; computers and phones have enough space.

α	Key generation and verification (ms)	Exchanged Data (B) (32 *B identity+Helper data*)	Key material size (kB)	Key material generation (ms)
2000	0.314	64	256	63
4000	0.670	71	577	134
8000	1.480	75	1217	296
16000	3.110	80	2561	622
32000	6.685	87	5633	1337

Table 2: HIMMO performance for generating a 256 bit key.This key is computed from several HIMMO instances for $b = B = 32$. HIMMO is highly performing and communication cost remains very low even if the security parameter α is much higher than 2000, recommended security parameter in [15]

6.2 Comparison, discussion, and applicability

The solution proposed in this paper exhibits advantageous properties compared with other architectures as shown in Table 3. First, the usage of HIMMO makes the operational communication link between any pair of devices quantum-resistant, independently of the public-key method. Secondly, it provides a solid solution for securely distributing key material during the initial enrollment phase. Since the roots of trust are based on HIMMO-TTPs, the proposed architecture supports multiple TTPs with low overhead, ensuring that the system remains secure even if some roots of trust are compromised, thus preventing an attacker from impersonating identities. From an operational perspective, if the secure links are established based on HIMMO only, then this solution ensures a quasi non-interactive method for key agreement and authentication, ideal for IoT scenarios. This last property together with the multiple TTPs also provides a solid solution for use cases in which lawful interception of communication is required. Furthermore, if wished, the proposed solution can operate with public-keys (so that TTPs cannot monitor the communication) and even provide forward secrecy. Most importantly, a HIMMO key material assigned to a node is linked to its public-keys so that the HIMMO key material acts as the node's implicit digital certificate. This allows a node to verify public parameters of other devices with very low overhead while supporting multiple roots of trust. The fact that HIMMO is based on identities makes the upgrade of existing pre-quantum protocols simple, just use any digital certificate or public-key as HIMMO identity and use existing public-keys to retrieve HIMMO key material from multiple TTPs. Thus, the proposed architecture scales well, even if an out-of-band channel is not available as in the Internet, and is applicable to the widest range of use cases. For instance, the hybrid TLS handshake focuses on traditional Internet applications only and does not support authentication by multiple roots of trust. The DTLS-HIMMO work [14] addressed IoT use cases but was lacking a solution for in-band enrollment or the capability of providing forward secrecy.

To quantify the gains of our approach, we consider two signatures schemes considered quantum-resistant, namely XMSS [3] and NTRUMLS [17]. XMSS with an h value equal to 8 (i.e., the height of the XMSS hash tree) allows creating up to 256 signatures, while still involving signatures of around 2436 Bytes and having a signature verification time of 1.95 ms. NTRUMLS for parameters 401 and 743 involve signatures of around 853 and 1765 Bytes with verification times of 0.33 and 0.79 ms, respectively. Thus, the main advantage of our approach relates to the implicit verification of public-keys that involves minimal overhead.

Supported Features	**Traditional PKI**	**Kerberos**	**KPS**	QSH-TLS [23]	DTLS-HIMMO [14]	**This paper**
Quantum-safe confidentiality	Depends on PK	Yes	Yes	Yes	Yes	Yes
Quantum-safe authentication	Depends on PK	Yes	Yes	No	Yes	Yes
Enrollment	Secure in-band	Out-of-band	Out-of-band	Secure in-band	Out-of-band	Secure in-band
Multiple roots of trust	Yes (high overhead)	No	Yes	No	Yes	Yes (low overhead)
Non-interactive operation	No	No	Yes	No	Yes	Yes
Forward secrecy	Yes	No	No	Yes	No	Yes
Authentication overhead	High (explicit signature)	High (online interaction)	Low (symmetric)	High (explicit signature)	Low (symmetric)	Low (symmetric)
Non-repudiation	Yes	No	No	Classical	No	Yes
Scalability	High	Medium	Medium	QH	High	Highest

Table 3: Architecture comparison

Finally, we discuss how our architecture applies to four use cases. The first challenging use case is how to start improving security of existing security protocols to make them quantum-safe or to deal with unknown weaknesses while minimizing changes in Standards or an increase to the communication overhead. The identity-based nature of the proposed solution easily enables this, for example changes and additional overhead in protocols like TLS are minimal. An entity can just send its classical public-key or certificate to a TTP, and receive a HIMMO key material, associated to the public-key or certificate, in a secure way. If multiple TTPs are used, the security is further increased. This improves security today since the communication link is protected and authenticated by HIMMO too. An attacker can only break the communications in the future if he manages to do one of the following: (a) eavesdropping on the communication of the key material assigned to a node by all the TTPs and decrypting it in the future once a quantum computer is available; or (b) compromising all TTPs.

Another challenge is the efficient certification and verification of public-keys to ensure a secure Internet (in a post-quantum world). Although a KPS, HIMMO in particular, is not a general digital signature scheme, it can fulfill the required functionality in one-to-one communication scenarios as is the case of TLS, SSH, IKE, or many others. The hybrid architecture achieves this at very low cost while being more resilient as multiple TTPs are supported.

A third use case refers to the lawful interception of communications. While the proposed hybrid architecture supports forward secrecy, in some applications, this might interfere with lawful interception. In these scenarios, a key that is solely based on the HIMMO key material from multiple TTPs managed by multiple organizations provides an more secure alternative to a single root of trust. This approach ensures that a communication link can only be intercepted if all TTPs agree.

The Internet of Things is maybe the most challenging scenario that the hybrid architecture addresses. In the Internet of Things, devices will require information to establish a secure link with other peers and let other devices verify their credentials. This is difficult with today's standards, as handling all those keys and credentials is a cumbersome task, small devices often lack resources, and pairwise key solutions do not support full peer-2-peer connectivity. As [14], the proposed architecture addresses this in a future-proof way: the same TTP infrastructure – which can efficiently certify and verify public-keys in the Internet – can also create and distribute the HIMMO key material to the devices in the Internet of Things and efficiently enable secure communications.

7 Conclusions

We presented a hybrid security architecture combining public-key cryptography and key pre-distribution schemes. We chose HIMMO as KPS since it is the only known scheme that is both efficient and collusion resistant. The hybrid architecture combines the best features of both worlds: it is scalable, supports secure enrollment and efficient support for multiple roots of trust. Due to its non-interactive key agreement, the scheme is ideal for use cases with strict timing constraints, limited bandwidth or a tight energy budget, as often encountered in the IoT. The hybrid scheme also finds a broader application in the Internet since it allows for efficient verification of public-keys with minimal overhead, while providing advanced features such as forward secrecy.

The architecture can be used already today to improve performance of Internet communications in a pre-quantum world, but it also provides a path towards a quantum-resistant, yet highly efficient, Internet. Our TLS implementation proves this by showing how existing non-quantum resistant communications based on ECC or quantum-safe public-keys such as NTRU public-keys can be verified with an increased communication overhead of less than 1 % and very limited computational overhead. Thanks to the identity-based nature of the HIMMO scheme, the exchanged data amount of data and key generation speed remain practically constant even if the HIMMO security parameters are made extremely large.

References

1. Erdem Alkim, Lo Ducas, Thomas Poeppelmann, and Peter Schwabe. Post-Quantum Key Exchange – A New Hope, 2015. Cryptology ePrint Archive, Report 2015-1092.
2. J.W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem, 2014. `http://eprint.iacr.org/2014/599`.
3. J. Buchmann, E. Dahmen, and A. Hulsing. XMSS: A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions, 2011. `http://eprint.iacr.org/2015/1003`.
4. Denis Butin, Stefan-Lukas Gazdag, and Johannes Buchmann. Real-world post-quantum digital signatures. In *Cyber Security and Privacy*, pages 41–52. Springer, 2015.
5. Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the Practical Exploitability of Dual EC in TLS Implementations. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 319–335, San Diego, CA, August 2014. USENIX Association.
6. Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on Post-Quantum Cryptography. NISTIR8105, 2016. `http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf`.
7. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.
8. D. Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions, 2011. RFC6066.
9. P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), December 2005.
10. ETSI. TS 101 331 - V1.3.1 - Lawful Interception (LI); Requirements of Law Enforcement Agencies , October 2009. `https://archive.org/details/etsi_ts_101_331_v01.03.01`.
11. S. Fluhrer, D. McGrew, and P. Kampanakis. An Extension for Postquantum Security using Preshared Keys for IKEv2, 2015. `https://tools.ietf.org/html/draft-fluhrer-qr-ikev2-00`.
12. O. García-Morchón, D. Gómez-Pérez, J. Gutierrez, R. Rietman, B. Schoenmakers, and L. Tolhuizen. HIMMO: A Lightweight Collusion-Resistant Key Predistribution Scheme, 2015. Cryptology ePrint Archive, Report 2014-698, Version of Aug. 18, 2015.
13. O. Garcia-Morchon, R. Rietman, S. Sharma, L. Tolhuizen, and J.-L. Torre Arce. A comprehensive and lightweight security architecture to secure the IoT throughout the lifecycle of a device based on HIMMO, 2015. NIST Ligthweight Cryptography Workshop, July 20,21, 2015, `http://csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session8-garcia-morchon-paper.pdf`.
14. O. Garcia-Morchon, R. Rietman, S. Sharma, L. Tolhuizen, and J.-L. Torre Arce. DTLS-HIMMO: Achieving DTLS Certificate Security with Symmetric Key Overhead. In *Computer Security – ESORICS 2015*, volume 9326 of *Lecture Notes in Computer Science*, pages 224–242, 2015.

15. O. García-Morchón, R. Rietman, L. Tolhuizen, J.L. Torre-Arce, M.S. Lee, D. Gómez-Pérez, J. Gutierrez, and B. Schoenmakers. Attacks and parameter choices in HIMMO, 2016. Cryptology ePrint Archive, Report 2016-152.
16. Gemalto. Gemalto presents the findings of its investigations, 2015. `http://www.gemalto.com/press/Pages/Gemalto presents the findings of its investigations into the alleged hacking of SIM card encryption keys.aspx`.
17. Jeff Hoffstein, Jill Pipher, JohnM. Schanck, JosephH. Silverman, and William Whyte. Transcript Secure Signatures Based on Modular Lattices. In Michele Mosca, editor, *Post-Quantum Cryptography*, volume 8772 of *Lecture Notes in Computer Science*, pages 142–159. Springer International Publishing, 2014.
18. Jeffrey Hoffstein, Jill Pipher, and JosephH. Silverman. NTRU: A ring-based public key cryptosystem. In JoeP. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Berlin Heidelberg, 1998.
19. J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), 1993. RFC1510.
20. T. Matsumoto and H. Imai. On the key predistribution system: a practical solution to the key distribution problem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, LNCS 293, pages 185–193. Springer, 1988.
21. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978.
22. JR Prins and Business Unit Cybercrime. Diginotar certificate authority breach *Operation Black Tulip*, September 2011. `www.enisa.europa.eu/media/news-items/operation-black-tulip`.
23. J.M. Schanck, W. Whyte, and Z. Zhang. Quantum-Safe Hybrid (QSH) Ciphersuite for Transport Layer Security (TLS) version 1.3. IETF, 2015. `https://tools.ietf.org/html/draft-whyte-qsh-tls13-01`.

A Appendix: HIMMO

This appendix provides further details about HIMMO as described in [15]. For all integers x and all positive integers N, we denote with $\langle x\rangle_N$ the integer in the interval $[0, N-1]$ that is equivalent to x modulo N, i.e., that differs an integer multiple of N. Moreover, we denote with $\{x\}_N$ the integer in the interval $(-\frac{N}{2}, \frac{N}{2}]$ that is equivalent to x modulo N. HIMMO has several system parameters, viz. B, the bit length of node identifiers, b, the key length, N, an odd reduction integer, and integers α and m. Other system parameters are nonnegative integer functions $\phi_0, \ldots, \phi_\alpha$ on $\{0, 1, \ldots, 2^B - 1\}$. We remark that α plays a key role both in the security and performance of the system.

As in any key pre-distribution scheme, HIMMO relies on at least a trusted third party (TTP), and several phases can be distinguished during its operation.

In the **set-up phase**, the TTP, given the system parameters, secretly and randomly generates the following root key material:

- m distinct random moduli $q_1, q_2, \ldots, q_m$ of the form $q_i = N - 2^b\beta_i$, where $0 \leq \beta_i < 2^B$ and at least one of $\beta_1, \ldots, \beta_m$ is odd.

- for $1 \leq i \leq m$ and $0 \leq j \leq k \leq \alpha$, a random integer $R^{(i)}_{j,k}$ with $0 \leq R^{(i)}_{j,k} \leq q_i - 1$, and for $k < j \leq \alpha$, $R^{(i)}_{j,k} = R^{(i)}_{k,j}$

In the **key material extraction phase**, the TTP provides node $\xi \in [0, 2^B)$ with coefficients $G_0(\xi), \ldots, G_\alpha(\xi)$, where for $0 \leq k \leq \alpha$

$$G_k(\xi) = \langle \sum_{i=1}^{m} \langle \sum_{j=0}^{\alpha} R^i_{j,k} \phi_j(\xi) \rangle_{q_i} \rangle_N. \tag{1}$$

A later phase comprises a **key establishment protocol** in which if node ξ wishes to communicate with node η, it computes the key $K_{\xi,\eta}$ defined as

$$K_{\xi,\eta} = \langle G^\xi(\eta) \rangle_{2^b}, \text{ where } G^\xi(\eta) = \langle \sum_{k=0}^{\alpha} G_k(\xi) \phi_k(\eta) \rangle_N. \tag{2}$$

In general, for $\xi, \eta \in [0, 2^B)$, the generated keys $K_{\xi,\eta}$ and $K_{\eta,\xi}$ need not be equal, but they are close. In order to arrive at a common key, node ξ sends to node η some helper data, for instance, the s least significant bits of $K_{\xi,\eta}$, where $s := \lceil \log_2(2\Delta + 1) \rceil$. From these s bits and $K_{\eta,\xi}$, node η can determine $K_{\xi,\eta}$, see [15][12] for more details. The remaining $b - s$ most significant bits of $K_{\xi,\eta}$ then serve as a common key for ξ and η.

Implicit certification and verification of credentials is further enabled on top of the HIMMO scheme since HIMMO is based on identities. A node that wants to register with the system provides the TTP with information to be certified, e.g., device type, manufacturing date, etc. We will call this the node certificate. The TTP, which can also add further information to the node's certificate such as a unique node identifier or the issue date of the key material and its expiration date, obtains the node's identity as $\xi = H(certificate)$, where H is a public hash function. In this way, HIMMO does not only allow for the direct secure exchange of a message M, but also for implicit certification and verification of ξ's credentials because the key material assigned to a node is linked to its certificate by means of H. If the output size of H is long enough, e.g., 256 bits, the input (i.e., the certificate) contains a unique node identifier, and if H is a secure one-way hash function, then it is infeasible for an attacker to find any other set of information leading to the same identity ξ.

Using multiple TTPs was introduced by Matsumoto and Imai [20] for KPS and can also be elegantly supported by HIMMO [15] [12]. In this set-up, a number of TTPs provides a node with key material linked to the node's identifier during the key material extraction phase. Upon reception, the device combines the different key material by adding the coefficients of the key generating polynomials modulo N. Without increasing the resource requirements at the nodes, this scheme enjoys two interesting properties. First, privacy is enhanced since a single TTP cannot eavesdrop the communication links. In fact, all TTPs should collude to monitor the communication links. Secondly, compromising a sub-set of TTPs does not break the overall system.