Efficient Quantum-Resistant Trust Infrastructure based on HIMMO

Oscar García Morchón*, Sauvik Bhattacharya*†, Ronald Rietman*,
Ludo Tolhuizen*, Jose-Luis Torre-Arce* and Maarten Bodlaender*
*Philips Group Innovation, Research, Eindhoven, The Netherlands
†RWTH Aachen University, Aachen, Germany
Email: oscar.garcia, sauvik.bhattacharya, ronald.rietman, ludo.tolhuizen, jose.luis.torre.arce,
maarten.bodlaender@philips.com

Abstract

Secure Internet communications face conflicting demands: advances in (quantum) computers require stronger, quantum-resistant cryptographic algorithms, while at the same time the Internet of Things demands better-performing protocols. 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 addresses these problems by proposing a hybrid infrastructure that combines the quantum-resistant HIMMO key pre-distribution scheme based on multiple Trusted Third Parties with public-key cryptography. During operation, any pair of devices can use private HIMMO key material and public keys to establish a secure and authenticated link, where their public keys are certified beforehand by multiple TTPs, acting as roots of trust. The solution is resilient to the capture of individual roots of trust without affecting performance, while public-key cryptography 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.

Our TLS proof-of-concept shows the feasibility of our proposal by integrating the above security features with minimal changes. Our TLS implementation provides classic and post-quantum confidentiality and authentication, all while adding a computation overhead of only 2.8% and communication overhead of approximately 50 bytes to a pre-quantum Elliptic Curve Diffie-Hellman ciphersuite.

Keywords

Post-Quantum Cryptography, Authentication, Root of Trust, HIMMO, TLS, Security Architecture.

I. Introduction

Secure Internet communications are a must nowadays. Otherwise, services such as doing our tax declarations on-line, private communications, 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 [34] can be used to break all public-key algorithms currently used in the Transport Layer Security (TLS) protocol. This threat has led to the recent NIST and ETSI announcement of future standardization plans of post-quantum cryptography [8]. 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 [29] would require 1 MByte public-key, and XMSS [5] involves signatures of tens of KBytes. The ETSI report in [14] provides a good summary of existing algorithms and their performance. Second, the manipulation of Certification Authorities (CAs), the Roots of Trust in existing Public-Key Infrastructure (PKI) can have catastrophic consequences, as with the hack of Diginotar Certification Authority (CA) [30] in 2011 or the alleged compromise of Gemalto's key server [21]. A third 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 crime.

Thus, the problem statement of this paper focuses on four challenges: (i) ensuring the availability of 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 (IoT). Furthermore, (iii) the architecture should be resilient to the capture of Roots of Trust in order to ensure the confidentiality, integrity, and authenticity of communications. At the same time, (iv) the solution should be able to accommodate trade-offs between the right of privacy in legitimate communications and the lawful interception of communication required in specific circumstances.

This paper contributes a hybrid architecture combining desirable features of traditional PKI and the quantum-resistant HIMMO Key Pre-distribution (KPS) scheme [16][19] 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 authenticate them

by means of symmetric-key cryptography. This addresses the need of quantum-resistance and efficiency in the solution. At the same time, the usage of public-key cryptography ensures advanced security properties such as forward-secrecy and also allows for the secure initial distribution of secret HIMMO key material in-band. Furthermore, its use in combination with HIMMO to protect the communication between parties ensures confidentiality even in the presence of Trusted Third Parties (TTPs), in strong contrast to how it typically is in the case of Key Pre-distribution. Public keys can further 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 yet is still efficient. The architecture can be configured to flexibly 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.

The remainder of this paper is structured as follows. After summarizing background work in Section II, we come to the main contributions of this paper. Section III details security needs at system and protocol level, limitations, and goals for our work. Section IV presents our hybrid architecture combining HIMMO with public-key cryptography, and accompanying protocols. Section V details our proof-of-concept based on TLS in which HIMMO is used to certify ECC and NTRU public keys and used to create a confidential, authenticated TLS session. Section VI provides an overall evaluation of the proposed solution. Section VII concludes this paper and points out future research.

II. Background

There are different types of **security architectures**. A Public-Key Infrastructure is built on a number of Certification Authorities 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 [25] rely on a Trusted Third Party (TTP) called a Key Distribution Center (KDC). The KDC enables 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 her to communicate securely 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 also rely on a TTP, which owns some *secret* root key material R. When a party, e.g., Alice enrolls with the TTP, it provides her with some key material that depends on R and her identity. This key material is thus unique to Alice and is kept private by her. Once a pair of entities have been enrolled, they can establish a common symmetric-key based on their respective key materials 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 [28] 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 [8] of the 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 [2], McEliece [29], Rainbow [10] and XMSS [5]. 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.

The HIMMO scheme has been introduced as an efficient and collusion- and quantum-resistant KPS [16],[18],[19]. In its basic version (see Appendix B for further details), there is a single TTP owning some root key material $R(x, y)$. During enrollment, the TTP provides Alice with a secret function $\mathbf{g}_A$ in a secure way, known as Alice's HIMMO key material. During operation, Alice and Bob can agree on a common key $K_{A,B}$ since $\mathbf{g}_A(B) \approx \mathbf{g}_B(A)$. HIMMO's cryptanalysis relies on lattice techniques [19], 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 $K_{A,B} = \mathbf{g}_A(B)$ using publicly-verifiable identities in the place of B that are cryptographically bound to public keys. To the best of our knowledge, HIMMO is the only known scheme with this properties. The work in [32] can be considered as a collusion-resistant KPS, but it is not quantum-resistant. Furthermore, HIMMO also supports multiple TTPs efficiently. 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 in a mutual authentication handshake and this handshake is successful, then both parties successfully verify 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.

The Transport Layer Security 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 their supported ciphersuites and engage in a protocol, usually based on public-key cryptography, that enables key agreement, (mutual) authentication, and depending on the protocol used, 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 [12]. In order to prepare for this, there have been publications in which quantum-resistant candidates have been integrated with TLS, including TLS-LWE [4] or DTLS-HIMMO [18].

In the IETF there are two initiatives to make TLS and Internet Key Exchange (IKEv2) more quantum-resistant. The hybrid TLS handshake in [33] extends TLS in a modular way with quantum-resistant key encapsulation or transport schemes, such as NTRUEncrypt [23]. In this hybrid handshake, key agreement is performed by means of *both* a classical key establishment scheme, e.g., ECDH, and a post-quantum scheme, e.g., NTRUEncrypt in parallel. 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 key establishment. Another reason is that there are no good candidates exhibiting a good security confidence and performance.

The very recent Internet Draft [15] 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 Internet Protocol Security (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.

III. Goals & Problem Statement

In this section, we first describe the threat model that we assume for our security architecture (Section III-A). Then we state the goals that our security architecture aims to achieve, in the context of security, operational and performance requirements in Section III-B, and formally describe the problem addressed by this work in Section III-C.

A. Attack Model

In the advent of a post-quantum world, we consider an attack model in which attackers (not necessary the same one) can *firstly*, own and run a quantum-computer and break classical public-key cryptography algorithms with a quantum computer. *Secondly*, the attackers are able to compromise or weaken algorithms with classical attacks without others knowing it [7]. *Thirdly*, attackers can *harvest and store* exchanged messages over the Internet during a pre-quantum era and decrypt them once a quantum computer is available. *Fourthly*, they can monitor communication links. *Fifthly*, the attackers are able to modify communications between a pair of parties. *Sixthly*, they are able to compromise roots of trust such as CAs or TTPs, e.g., [21] or [30]. *Finally*, these attackers are also 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.

B. Design Goals

In order to prevent attackers possessing capabilities such as those described in Section III-A from achieving their goals and to fulfill regulations, we would like a security architecture with the following **security 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.

Furthermore, certain regulations require the capability of lawful interception of communications [13] 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 the above-mentioned security goals, but also aim at achieving as good a performance as possible in terms of bandwidth, computation cost, energy, and memory.
- The design of constituent protocols in the architecture should be modular, in order to easily add multiple ciphersuites, also quantum-resistant ones for both key exchange and digital signatures.

C. Problem Statement

When we consider the above design goals in Section III-B, the attack model, and the current state-of-the-art reviewed in Section II, the problem statement of this paper is fourfold.

Firstly, there are many potential quantum-resistant candidates for key-exchange and authentication, 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 [6]. This is a serious limitation, even more than their high overhead. It makes private keys a critical resource, and key management extremely risky. Even one instance of a signing key's reuse brings the security of the entire signature scheme into question.

Secondly, the security of communication links in Internet protocols usually fully depend 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 without a drop in performance and in this way, be resilient to the capture of individual Roots of Trust ensuring the privacy of communications.

Thirdly, the right to privacy and the lawful interception of communication are contradictory goals. However, both goals need to be accommodated in a common architecture so that users have the confidence that their data is not monitored or manipulated, but if required and agreed upon in special circumstances, access by law enforcement to such data 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.

IV. Design

This section describes our hybrid trust infrastructure and the rationale behind its design. The basic design concept is twofold:

Firstly, 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.

Secondly, use public-key cryptography (ideally quantum-resistant) to enable the secure distribution of HIMMO key material to devices and to enable further advanced security features like forward-secrecy.

The architecture can work with any type of public keys and any KPS. We use HIMMO as the KPS since it is – to the best of our knowledge, the only KPS that enables *both* quantum resistant and collusion-resistant key agreement in an efficient way. To the best of our knowledge, the only other scheme known to resemble a fully collusion-resistant KPS is [32] [9], however it is not quantum-resistant.

This design approach combines desirable security features of both Public-Key Infrastructure and Key Pre-distribution, such as efficiency, offline operation, secure initialization, low overhead, forward-secrecy, and support for multiple roots of trust. In the following subsections we discuss three aspects: (i) roots of trust, (ii) the enrollment and certification procedure for a party or device, and (iii) operation of a secure handshake between two parties.

A. Roots of Trust in a PQ World

A fully general architecture can consider roots of trust supporting (i) traditional public-key infrastructure such as the ability 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 only a TTP 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, a HIMMO TTP is not involved during operation, i.e., during actual communication of devices/nodes. Second, multiple roots of trust can also be 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 efficient quantum-safe public-key authentication schemes.

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 of distributing key material to them. Thus, a single root of trust, i.e., a 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.

B. Enrollment & 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

Figure 1. Enrollment and Certification of a communicating entity *Alice* with a root of trust, i.e., a HIMMO TTP, resulting in Alice getting her long-term public-key Pk_a certified and securely receiving private HIMMO key material $\mathbf{g}_{Pk_a}(x)$ that is bound to Pk_a. This key material is created by the HIMMO TTP using its secret root key material R.

Alice	**TTP (Root of Trust)**
$\xrightarrow{1.\ Pk_a,\ N_a}$	
	2. $\mathbf{g}_{Pk_a}(x) = R(hash(Pk_a), x)$
$\xleftarrow{3.\ E_{Pk_a}\{\mathbf{g}_{Pk_a}(x) \mid hash(\mathbf{g}_{Pk_a}(x) \mid N_a)\}}$	
4. Decrypt key material $\mathbf{g}_{Pk_a}(x)$, Verify integrity and N_a.	

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 [17], this is not the case for the Internet. The challenge is therefore how to securely distribute HIMMO key material over an open network. 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. Alice then enrolls with one or several HIMMO TTPs. To this end, she must send to each TTP this set of identification 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 for Alice by hashing the certificate and uses this to obtain the corresponding HIMMO key material. Therefore, through Alice's identity and in turn through the certificate containing her public-key, the HIMMO key material generated for Alice is now cryptographically bound to her 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), *all of which* is 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 into *one single instance* of HIMMO key material by adding the coefficients of multiple HIMMO key materials received from different TTPs, one by one.

C. Secure Operation

Now the question is how Alice and Bob can establish a secure communication session in the above setting, i.e., each of them has public keys bound to private HIMMO key materials 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 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 (ideally quantum-resistant) to achieve strong properties such as forward-secrecy, and on HIMMO to verify the identities of the communicating parties and thus to ensure an authenticated channel.
- 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 traditional digital signatures, but HIMMO. In this case, the HIMMO key material of a party is cryptographically bound to its identity, which in turn is bound to the party's public-key. Therefore, if the exposed public keys are the right ones, HIMMO key generation at both sides will happen in a proper way. Consequently, if an authentication handshake between the parties based on their generated HIMMO keys is successful, *both* of their public keys will be automatically verified.

We note that the verification of public-keys in our proposed infrastructure is more "horizontal" than in today's public-key infrastructure that relies on an infrastructure of Certification Authorities (CAs). If a single CA is compromised, then the

Figure 2. Secure Operation for establishing a confidential, authenticated channel between two parties *Alice* and *Bob*. Authentication is achieved using HIMMO key material bound to long-term, certified public keys. Confidentiality is achieved using a combination of HIMMO and a shared session key transported using asymmetric cryptography, ideally quantum-resistant.

Alice		**Bob**
	$\xrightarrow{1.\ Pk_a,\ N_a}$	
		2. $\{k_{\mathbf{b,a}}^{\text{HIMMO}}, \text{HelperData}\} = \mathbf{g}_b(Pk_a)$, Create session key k, $k_{\mathbf{b,a}} = KDF(k_{\mathbf{b,a}}^{\text{HIMMO}}, k)$, $MAC_{b,a} = MAC_{k_{\mathbf{b,a}}}(k\|N_b\|N_a)$.
	$\xleftarrow{3.\ Pk_b,\ N_b,\ MAC_{b,a},\ E_{Pk_a}(k),\ \text{HelperData}}$	
4. Decrypt k, $k_{\mathbf{b,a}}^{\text{HIMMO}} = \mathbf{g}_a(Pk_b) + \text{HelperData}$, $k_{\mathbf{b,a}} = KDF(k_{\mathbf{b,a}}^{\text{HIMMO}}, k)$, Verify $MAC_{b,a}$ using $k_{\mathbf{b,a}}$, $MAC_{a,b} = MAC_{k_{\mathbf{b,a}}}(k\|N_a\|N_b)$.		
	$\xrightarrow{5.\ MAC_{a,b}}$	
		6. Verify $MAC_{a,b}$.

whole PKI is broken. In our proposed solution, 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 due to the nature of HIMMO's operation while using multiple TTPs. Furthermore, the fact that asymmetric-cryptography is used for key exchange ensures that TTPs cannot access the communication and enables forward-secrecy for advanced use cases, e.g., communication over the Internet.

Since the solution is TTP-based, this architecture can also be applied to diverse scenarios such as those requiring lower communication/computational overhead, e.g., the Internet of Things, or those potentially requiring lawful interception of communications (achieved via a unanimous cooperation of all TTPs) but still maintaining privacy assurances by relying on multiple TTPs. Finally, the usage of HIMMO in the eventual creation of the final key between peers ensures that their authentication and the resulting communication link are quantum-resistant.

The above general operational protocol is exemplified in a very specific operational handshake for key exchange and mutual authentication depicted in Figure 2. In this example, a single TTP is commonly supported by the communicating entities, Alice and Bob. In Step (1), Alice sends her public-key to Bob, along with a fresh nonce N_a. In Step (2), Bob computes the HIMMO key $k_{\mathbf{b,a}}^{\text{HIMMO}}$ using HIMMO and additionally randomly generates a secret symmetric session key k. Bob then securely combines both keys and uses the resulting key to compute a Message Authentication Code (MAC) based on the session key k, the nonce received from Alice and another nonce locally generated. Next in Step (3), Bob replies to Alice exchanging his long-term 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 that Bob computed the same key $k_{\mathbf{b,a}}^{\text{HIMMO}}$ and therefore, verify Bob's public key. Furthermore, Bob also sends the session key k to Alice, encrypted using her public-key.

After decrypting and recovering the session key k, Alice independently regenerates $k_{\mathbf{b,a}}^{\text{HIMMO}}$ and obtains the combined key in a manner similar to Bob. She achieves this by using non-secret HIMMO *helper data* sent to her by Bob, to reconcile any small difference between her own and Bob's computed HIMMO keys. Following this, both Alice and Bob possess the same symmetric HIMMO key $k_{\mathbf{b,a}}^{\text{HIMMO}}$. Finally, in Step (5) Alice sends her MAC to Bob so that he can verify Alice's long-term public-key and the generated session key k in Step (6). Next, Alice and Bob can proceed to communicate in a secure, authenticated way, using the final key created by combining $k_{\mathbf{b,a}}^{\text{HIMMO}}$ and k to protect their communication. The structure of this handshake can further be extended to support more generic use-cases and advanced security properties such as forward-secrecy, via the use of ephemeral public keys to transport the session key between Alice and Bob. Similar hybrid handshakes can be constructed using an (ephemeral) Diffie-Hellman key exchange.

V. 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 Internet applications. We use HIMMO in our proof of concept since it

Figure 3. TLS Handshake including the design proposed in this paper, including negotiation of a common set of trusted TTPs and establishment of a confidential, authenticated channel protected by a combination of a classic secret, a quantum-safe secret $CliS$ and a quantum-safe HIMMO key. The HIMMO key is created using a combination of key materials obtained from multiple, negotiated TTPs and the peer's long-term public-key, ideally quantum-resistant.

Client		**Server**
1. ClientHello = Ciphersuites$_c$ + TTPList$_c$[1] + ...		
	2. ClientHello →	
		3. ServerHello = Ciphersuites$_s$ + TTPList$_s$ + ...
	← 4. ServerHello	
	⋮	
5. $\{k^{\text{HIMMO}}_{\mathbf{c,s}}, \text{HelperData}\} = \mathbf{g}^1_c(QshPk_s) + \mathbf{g}^2_c(QshPk_s) + \ldots$		
6. ClientKeyExchange = $\{E_{\text{ClassicPk}_s}(\text{ClassicS}) \vert E_{\text{QshPk}_s}(\text{CliS}) \vert \text{HelperData}\}$[2]		
	7. ClientKeyExchange →	
		8. Decrypt ClassicS, CliS.
	⋮	
		9. $k^{\text{HIMMO}}_{\mathbf{c,s}} = \mathbf{g}^1_s(QshPk_c) + \mathbf{g}^2_s(QshPk_c) + \ldots +$ HelperData.
10. PMS[3]= $\{\text{Classic} \vert \text{CliS} \vert k^{\text{HIMMO}}_{\mathbf{c,s}}\}$.		10. PMS = $\{\text{Classic} \vert \text{CliS} \vert k^{\text{HIMMO}}_{\mathbf{c,s}}\}$.
11. MasterSecret = PRF(PMS).		11. MasterSecret = PRF(PMS).
	12. Finished$_a$ →	
		13. Verify Finished$_a$.
	← 14. Finished$_b$	
15. Verify Finished$_b$.		

[1] List of supported HIMMO TTP IDs as per RFC 6066 [11].
[2] ClassicS, CliS are the classic secret and client QSH (assuming [33]) secret respectively.
[3] Pre-master Secret used to generate the Master Secret by means of the TLS pseudo-random function PRF.

enables for the efficient distribution of pairwise keys, enabling implicit verification of credentials, and supporting multiple TTPs. We further use ECDH and ECDSA as the pre-quantum cryptographic primitives for key exchange and digital signatures and NTRUEncrypt [23] is taken as a post-quantum candidate for public-key encryption.

Figure 3 shows the complete TLS handshake between a client, *Client*, and a server, *Server* when we apply the design explained in Section IV. 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 [11] since it defines extensions to the *ClientHello* and *ServerHello* messages in which 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" [11]. 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 IV, then they exchange the identities of their supported TTPs. Thus, in Step (1) of the handshake, Client creates a *ClientHello* message including its supported roots of trust, i.e., HIMMO TTPs, as described in RFC 6066 [11], that is then sent to Server in Step (2). In step (3), Server searches for a common set of HIMMO TTPs that it then includes in the *ServerHello* message and sends back to Client 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 already possess HIMMO 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 [33] 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 Client and Server to independently generate pairwise HIMMO keys in Steps (5) and (9) (one key for each common TTP agreed upon), which they then use to independently derive the *final* HIMMO key k^{HIMMO} by means of a pseudo-random function of all previously generated HIMMO keys. In the specific example we describe in Figure 3, Client and Server use each other's quantum-resistant public-key to generate k^{HIMMO} in steps (5) and (9). This offers stronger, quantum-security. However, classic public-keys may also be used for this purpose to ease widespread

adoption. 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 further enhance security by computing the Master Secret used for protecting the TLS record layer not only from a single Pre-master secret *PMS*, but from both the classical key and the HIMMO key. This is similar to [33] and [15]. Note that this Master Secret is equal to the final session key $k_{\mathbf{b,a}}$ in Section IV-C and Figure 2 that is derived from the regular session key and the HIMMO key, and is used to protect and authenticate the final communication during Secure Operation. 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 Client and Server to calculate a Message Authentication Code 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 their own HIMMO key material, and can therefore be considered equal to the $\text{MAC}_{b,a}$, exchanged as an authentication token in Section IV-C during the Secure Operation phase (see Figure 2). This verification by Server and Client will fail unless they computed the right k^{HIMMO} using an authentic certificate/public-key of the other party and proper key material, and its success *automatically* authenticates both Server and Client to each other.

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 also being able to achieve advanced features such as forward-secrecy and non-repudiation (through the added incorporation of standard digital signatures). Lawful interception use-cases would require excluding the exchange of public keys during operation and relying on multiple TTPs (and thus HIMMO) only to provide a proper privacy trade-off. Note that the TLS extension presented in [18] also enables key establishment and mutual authentication based on multiple HIMMO TTPs.

VI. Evaluation

In this section, we begin with an analysis of the security provided by our hybrid architecture in Section VI-A, followed by a quantitative evaluation of its performance in Section VI-B. Section VI-C presents a qualitative discussion of the architecture's contributions and quantitatively compares it with existing post-quantum key-exchange and authentication schemes. Finally, Section VI-D discusses its applicability in various real-world use-cases.

A. Security Analysis

In this subsection, we look at our hybrid architecture and analyze its security. A core contribution of this architecture is imparting quantum-safe authentication and confidentiality to communication channels in a scalable and efficient manner. The private information possessed by a party, i.e., the secret HIMMO key material plays a key part in this and must therefore be securely allocated to the party beforehand. Such confidential distribution of HIMMO key material is achieved by the Enrollment phase, as it allows parties to get their public keys certified by (multiple) root(s) of trust and securely receive key materials from them, in a manner analogous to traditional Public-Key Infrastructure. While allocating a HIMMO key material to an enrolling party, a root of trust encrypts it using the recipient party's public-key before transmitting the key material to it, thereby ensuring confidentiality of the private key material. Furthermore, the transmission of a cryptographically-secure hash of the key material along with the actual key material itself allows its integrity to be checked by the recipient party. Freshness of the key material received by a party is also ensured and message-replay attacks are prevented by the transmission of freshly-generated nonces. However, an existing challenge during the Enrollment phase is the achievement of correct authentication or verification of an enrolling party's identity by a root of trust. We note that this problem also exists in current PKI, and has led to initiatives such as the Certificate Transparency audit framework [26]. Finally, we emphasize once again that quantum-safe asymmetric cryptography should ideally be used during Enrollment, to make sure that the key material of enrolled parties remains confidential. Using classic public-key cryptography suffices *only* for the short-term to ease widespread adoption, but opens up ways for quantum-capable attacks in the long-term, such as the recovery of a party's HIMMO key material by an attacker via eavesdropping on the communication between the party and root of trust, leading to more serious attacks such as identity impersonation [1].

Next, we discuss the security of the second step in the architecture, i.e., Secure Operation. Both confidentiality and authentication of the communication channel established between Alice and Bob during this step is due to the final key derived from *both* the HIMMO key k^{HIMMO} and the session key k exchanged using public-key cryptography. Authentication of the two parties' long-term public keys is implicitly and mutually achieved by a successful agreement of the HIMMO key between the parties, since it is derived from *both* the two public keys of the communicating parties *and* their private HIMMO keying materials. Therefore, a successful key agreement automatically verifies both parties' public keys to each other, since this agreement *never* works unless *both* parties perform this step using authentic, certified public keys and proper HIMMO

key materials that have been cryptographically bound to their certified public keys. The same idea also prevents conventional attacks such as a Man-in-the-Middle (MITM) attack on the handshake between Alice and Bob, since the authentication involves *both* of their private key materials and certified public keys (exchanged via verifiable certificates), thus making it impossible for the attacker to either impersonate Alice or Bob to the other party.

The final key and therefore the actual communication between the parties, receives combined protection from both HIMMO and public-key cryptography. The Leftover Hash Lemma [24] ensures that given an input X, one can extract Y bits that are almost uniformly distributed, where Y is asymptotic to the min-entropy of X. However, an attacker who *partially* knows X, will have almost no knowledge about Y. Thus, as long as *at least one* of the two secrets k^{HIMMO} and k has enough entropy and remains secret, the final key remains secure and thus all subsequent communications between Alice and Bob remain confidential *and* authenticated. Even if the attacker possesses quantum computing capabilities and is able to recover k, he fails to learn the key k^{HIMMO}. In [19], it has been shown that all attacks on HIMMO found so far lead to lattice-based cryptanalysis, thereby HIMMO provides quantum-security to any scheme that uses keys derived using it. Furthermore, [20] explicitly contains an analysis of quantum-capable attacks on the HIMMO scheme.

Thus, the overall communication benefits from both classic- and quantum-secure confidentiality and authentication. The architecture is also flexible enough to incorporate additional strong security properties such as forward-secrecy with minimal changes – such as, using an ephemeral public/private key-pair to transport the session key k between the communicating parties during the Secure Operation phase (see Figure 2). This ensures that k and therefore the final key (which is derived from a combination of k and the HIMMO key) remains secure even in the case of a long-term key's compromise in the future. This in turn means that the confidentiality and authentication of the communication (both secured using the final key) also remain equally secure.

Finally, we note that due to HIMMO's use of multiple TTPs and consequently our architecture's use of multiple roots of trust, the level of security can be increased n-fold (when using HIMMO key material obtained from n roots of trust during the Enrollment phase), with almost no corresponding increase in computation or communication cost due to the nature of HIMMO's operation when using multiple TTPs. Anything short of a *total* compromise of *all* roots of trust would affect neither confidentiality nor authentication. As long as even one root of trust remains unbroken, the final private HIMMO key material of the enrolled parties and therefore HIMMO symmetric keys generated by them remain secret, in turn ensuring that the final session or Master secret and therefore the final communication channel also remains secure.

B. Performance Evaluation & Benchmarks

To validate our research, we implement the protocol described in Section V by extending the CyaSSL library and using TLSv2. The TLS protocol is executed between two machines within the same Local Area Network. Each machine/PC uses Windows 7 Enterprise (64 bit), with an Intel Core i5-5300U @ 2.30 GHz. Software was compiled for the $\mathrm{x86\_64}$ architecture with full optimization ($/\mathrm{Ox}$) using the Microsoft Visual C compiler.

Table I 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. From the handshake times in Table I, it can be noted that the increase in running times of the TLS handshake for two ciphersuites – the classic ECDHE_ECDSA and the post-quantum ciphersuite NTRU_RSA, when extended with HIMMO as per the protocol in Section V is minimal, even while using three TTPs for higher security and ensuring that the confidentiality and authentication assured by the handshake remains immune to compromise of individual TTPs. Extending the classical ECDHE_ECDSA with a single-TTP HIMMO implementation results in a TLS handshake that is slower by only 2.8% for the HIMMO security parameter $t = 4000$ (50.5 ms for the extended ciphersuite as compared to 49.1 ms for classic ECDHE+ECDSA). Note that this value of t is quite conservative [19]; the drop in speed of the TLS handshake is minimal by comparison. For the post-quantum ciphersuite NTRU_RSA, the increase in computation overhead is 7.8% for a single-TTP implementation (31.6 ms for the single-TTP extended ciphersuite as compared to 29.3 ms for the original NTRU+RSA one) and 19.1% for a three-TTP implementation (34.9 ms), both cases for $t = 4000$. Note that a further major optimization is possible for the multiple-TTP implementation: 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 optimization is done, we expect a timing similar to the usage of a single of TTP irrespective of the number of TTPs used.

To further evaluate the proposal, we give insights on the performance of HIMMO standalone. Table II 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 machine. It is worth noting that an increase of the security parameter t to a very high value, that directly determines the dimension of the

Ciphersuite	Handshake time (ms)	Handshake size (Bytes)
ECDHE_ECDSA_WITH_AES_256_CBC_SHA (nistp256)	49.1	3277
NTRU_RSA_WITH_AES_256_CBC_SHA (EES439EP1)	29.3	5381
ECDHE_ECDSA_WITH_AES_256_CBC_SHA_HIMMO256	50.5	3330
NTRU_RSA_WITH_AES_256_CBC_SHA_HIMMO256	31.6	5448
NTRU_RSA_WITH_AES_256_CBC_SHA_HIMMO256_3_TTP	34.9	5640

Table I: TLS performance for different ciphersuites, including existing classical and hybrid post-quantum ciphersuites extended with HIMMO key generation using key material bound with certified public keys as per our design (256-bit HIMMO key with a large security parameter $t = 4000$). Extensions using both single-TTP and multi-TTP HIMMO implementations are shown. The Elliptic Curve key-exchange and signature implementations use the NIST P-256 curve, providing 128-bits of security. NTRUEncrypt key-exchange use the NTRU parameter set EES439EP1, with a 608 byte public key at 128-bit security.

t	Key Generation & Verification (ms)	Exchanged Data (B) (32 *B identity* + *Helper data*)	Key Material size (B)	Key Material Generation (ms)
2000	0.314	64	256	63
4000	0.670	71	577	134
8000	1.480	75	1217	296

Table II: 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 t is much higher than 2000, the recommended value for this security parameter in [19]

lattice that needs to be reduced in order to find a close/short vector [19], 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.

C. Discussion & Comparison

The solution proposed in this paper exhibits advantageous properties compared with other architectures as shown in Table III. *Firstly*, the usage of HIMMO to create the final session key that protects a secure channel between any pair of devices makes the operational communication link between them **quantum-resistant**, independent of the public-key cryptosystem. *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 roots of trust** 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 incorporation of multiple TTPs also provides a solid solution for use cases in which **lawful interception** of communication is required. However at the same time, this would not infringe upon the **right to privacy** of legitimate communications since it would require a unanimous cooperation among all TTPs involved in the communication, acting as a check against the possible abuse of power.

Furthermore, if wished, the proposed solution can operate with public keys (ensuring **confidentiality against TTPs' monitoring**) and even provide **forward-secrecy**. Most importantly, a HIMMO key material assigned to a node is bound to its public-key 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, through the use of any digital certificate or public-key as HIMMO identity and the use of existing public keys to retrieve HIMMO key material from multiple TTPs.

Supported Features	Traditional PKI	Kerberos	KPS	QSH-TLS [33]	DTLS-HIMMO [18]	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	Yes (high overhead)	Yes	Yes (low overhead)
Non-interactive operation	No	No	Yes	No	Yes	No
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)
Scalability	High	Medium	Medium	Medium	High	Higher

Table III: Comparison of the hybrid infrastructure presented in this paper with existing security architectures, for a number of security and operational features.

Thus, the proposed architecture is highly **scalable**, even if an out-of-band channel is not available as in the Internet, and is applicable to a wide range of use cases. For instance, the hybrid TLS handshake in [33] focuses on traditional Internet applications only and does not support authentication by multiple roots of trust. The DTLS-HIMMO work [18] 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 considering secure key-exchange between communicating parties, we consider the **Frodo** [3] and **NewHope** [2] post-quantum key-exchange schemes for TLS, which combine their LWE [31] and R-LWE [27] based key-exchange respectively, with classical ECDH to create hybrid TLS ciphersuites in a manner similar to ours. While extending ECDHE with Frodo's LWE-based key-exchange, a computation overhead of 8.7 ms is added to the baseline, i.e., the TLS handshake time for the classic ECDHE+ECDSA ciphersuite. Similarly, NewHope adds a computation overhead of 2.3 ms to the ECDHE ciphersuite's handshake time. In comparison, our solution of combining ECDHE with a post-quantum key-exchange (i.e., HIMMO key generation from certified public keys) adds a computation overhead of only 1.4 ms to the ECDHE baseline. Regarding communication overhead, Frodo causes an increase of 22.6 Kilobytes to the handshake size when used to extend the ECDHE ciphersuite, while NewHope causes an increase of 3.89 Kilobytes [3]. In comparison, our HIMMO-based implementation used to extend the same ECDHE ciphersuite causes an increase of only 53 bytes (the size of HIMMO helper data exchanged for key reconciliation) for a single-TTP implementation. Even while using a multi-TTP implementation which provides higher security, the size of the HIMMO helper data exchanged between parties for key reconciliation only grows *logarithmically* with the increase in number of TTPs (i.e., roots of trust) and not linearly. When a N-TTP implementation is used, the final key is derived as a combination of N intermediate HIMMO keys, each derived from one key material instance (obtained from one root of trust) with HIMMO parameters t and m. However, if the HIMMO parameter t is same for all intermediate key materials, then the final HIMMO key corresponds to a symbolic HIMMO key material with parameters t and Nm. The size u of the helper data for this HIMMO key would then be $u = \lceil \log_2(4Nmt + 1) \rceil$ (see Appendix B), which can be rounded to show that the size u grows as $\lceil \log_2(4Nmt) \rceil$. Hence, even with a growth in the number N of TTPs, u would still grow only logarithmically with N, leading to a highly scalable behavior of the resulting communication overhead when using this multi-TTP HIMMO implementation in our scheme. For more details about the exact nature of HIMMO helper data that leads to this desirable behavior, see Appendix B. Furthermore, we note that the capabilities of the evaluation platform used by the authors of [3] to obtain the above performance data for Frodo and NewHope are significantly different from (and superior to) our own, and take this into account while comparing our scheme's performance with Frodo and NewHope. We also note that our solution further achieves post-quantum authentication unlike the former two schemes, which offer quantum-resistance for only key-exchange. Finally, our solution is also capable of using *only* HIMMO to achieve *both* post-quantum key-exchange *and* authentication, which potentially results in performance gains that are magnitudes of order better than what results from using HIMMO in a hybrid manner along with comparatively slower schemes like ECDHE.

Next, considering authentication, we quantify the gains of our solution by comparing it with two signature schemes considered quantum-resistant, namely **XMSS** [5] and **NTRUMLS** [22]. 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 [14]. 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 [14]. We note that computationally, the actual post-quantum authentication step in our solution, i.e., the generation of the HIMMO symmetric key, is 65% faster than that of XMSS (0.67 ms compared to XMSS's 1.95 ms) and 15% faster than that of NTRUMLS (0.67 ms compared to NTRUMLS's 0.79 ms), considering the relatively secure NTRUMLS parameter set 743 and a high HIMMO security parameter value of $t = 4000$. Furthermore, Figure 4 compares the communication overhead during TLS authentication, comparing the amount of bytes exchanged by classic and existing post-quantum signatures with that exchanged by our HIMMO-based solution. Since this consists of only HIMMO helper data as compared to full-fledged digital signatures in the former case, our solution has a strong advantage

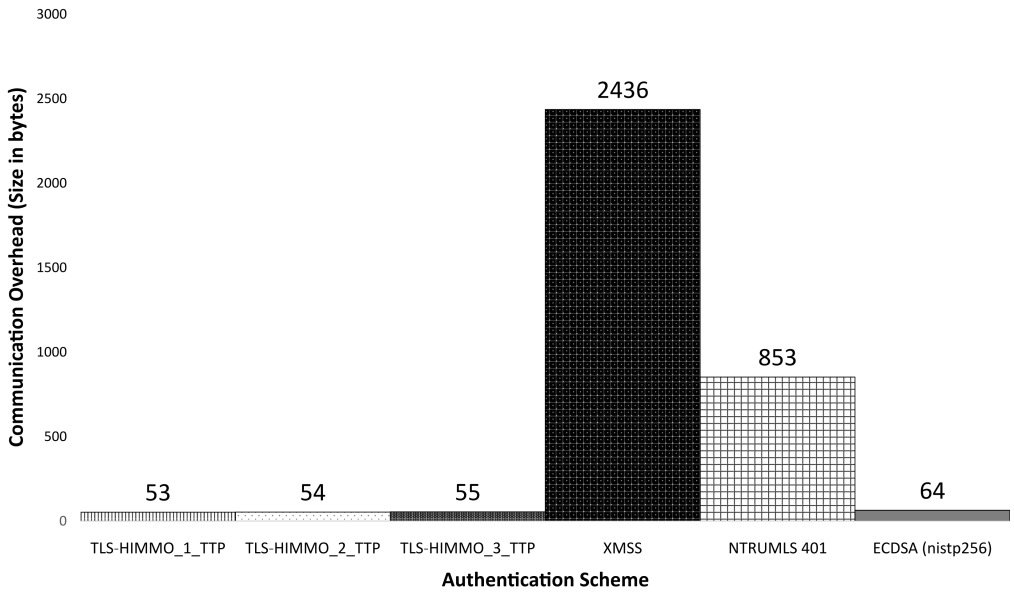

Figure 4. Comparison of communication overhead in TLS authentication phase when using: (a) HIMMO key generation-based authentication with one TTP, (b) with two TTPs, (c) with three TTPs (all using 256-bit HIMMO key with a large security parameter $t = 4000$), (d) XMSS with 82 bit security, (e) NTRUMLS with parameter set 401 [22] and 112 bit security, (f) classic ECDSA with the curve $\mathrm{nistp256}$. Data exchanged for authentication purposes when using our HIMMO-based solution amounts to only HIMMO helper data for key reconciliation.

regarding communication overhead or signature sizes. Figure 4 also shows that even achieving higher security through the use of *multiple roots of trust* (i.e., TTPs) comes at virtually no added cost, due to the logarithmic growth in the size of HIMMO helper data with the number of TTPs. In comparison, for traditional digital signatures such as XMSS and NTRUMLS (albeit quantum-resistant) to achieve the same increase in security, usage of certifications from multiple CAs is necessary. This involves a linear growth of computation overhead (key-pair generation, signature generation, signature verification) as well as communication overhead (exchange of multiple signatures), with the number of CAs involved.

Thus, the main advantage of our approach relates to the **implicit verification of public keys** that involves minimal overhead, even with stronger security conditions such as multiple roots of trust (see Section VI-B).

D. Applicability of our Trust Infrastructure

We discuss how our proposed solution can be used to secure four real-world, practical usage scenarios. With this section, we demonstrate the distinct advantages and flexibility offered by this hybrid security architecture's design.

1) Easy Transition towards Quantum-Resistant Security Protocols: 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. An entity can just send its classical public-key or certificate to a TTP (or several of them), and receive a HIMMO key material, associated to the public-key or certificate, in a secure way (assuming that the public-key cryptography used for communication between the root of trust, i.e., TTP and entity is secure against attacks). Once this is done, changes and additional overhead in protocols like TLS are minimal. 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 could 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 (which can however be prevented by the use of quantum-resistant asymmetric cryptography for all node-TTP communications, as recommended in Sections IV-B and VI-A); or (b) compromising all TTPs.

2) Efficient Verification of Public-Keys: Another challenge is the efficient certification and verification of public keys to ensure a secure Internet (in a post-quantum world). Although KPS in general, or HIMMO in particular, are not digital signature schemes in the classic definition, they can fulfill the required functionality in one-to-one communication scenarios as in the case of TLS, Secure Shell (SSH), IKE, or many others. Our proposal allows for the verification of public-keys with very low cost while being more resilient as multiple roots of trust, i.e., TTPs are supported so that the validity of a public-key, as certified by multiple TTPs, can be simultaneously verified.

3) Lawful Interception with multiple TTP Guarantees: An important use case refers to the lawful interception of communications [13]. While the proposed architecture supports forward-secrecy, in some applications, this might not be applied and the trust of lawful interception would be distributed among multiple TTPs managed by different organizations. This approach

would ensure that a single organization cannot monitor or modify communications, but if required, all TTPs could unanimously agree to cooperate on intercepting a given communication link. It is important to note here that nothing short of a *unanimous* agreement of all TTPs would allow this, acting as a check against possible abuse of this power.

4) Efficient Management of Symmetric-Keys and Credentials in the Internet of Things: The Internet of Things is another challenging scenario that can be addressed by means of this architecture. In the Internet of Things, each device will require information to establish a secure link with other peers and let other devices verify its credentials. This is a problem today since this involves a significant number of keys and credentials, and handling them is a cumbersome task. As [18], the proposed architecture addresses this in a future-proof way since 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.

VII. Conclusions

We presented a hybrid security architecture combining public-key cryptography and key pre-distribution schemes. We chose HIMMO as the KPS instantiation since it is the only known scheme that is both efficient and collusion- and quantum-resistant. The hybrid architecture combines desirable 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 already be deployed to improve performance of Internet communications in today's pre-quantum world while providing a path towards a quantum-resistant, yet highly efficient Internet. Our TLS proof-of-concept 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 low computational overhead. Thanks to the identity-based nature of the HIMMO scheme, the exchanged amount of data and time required for key generation remain practically constant even if the HIMMO security parameters are made extremely large.

References

[1] Carlisle Adams. *Impersonation Attack*, pages 596–596. Springer US, Boston, MA, 2011.

[2] Erdem Alkim, Lo Ducas, Thomas Poeppelmann, and Peter Schwabe. Post-Quantum Key Exchange – A New Hope, 2015. Cryptology ePrint Archive, Report 2015-1092.

[3] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. Cryptology ePrint Archive, Report 2016/659, 2016. http://eprint.iacr.org/2016/659.

[4] 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.

[5] 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.

[6] Denis Butin, Stefan-Lukas Gazdag, and Johannes Buchmann. Real-world post-quantum digital signatures. In *Cyber Security and Privacy*, pages 41–52. Springer, 2015.

[7] 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.

[8] 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.

[9] Yu Chen, Qiong Huang, and Zongyang Zhang. Sakai-ohgishi-kasahara identity-based non-interactive key exchange revisited and more. Cryptology ePrint Archive, Report 2014/310, 2014. http://eprint.iacr.org/2014/310.

[10] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.

[11] D. Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions, 2011. RFC6066.

[12] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), December 2005.

[13] 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.

[14] ETSI. GR/QSC-001 - V1.1.1 - Quantum-Safe Cryptography (QSC); Quantum-safe algorithmic framework , July 2016. http://www.etsi.org/deliver/etsi_gr/QSC/001_099/001/01.01.01_60/gr_QSC001v010101p.pdf.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Oscar Garcia-Morchon, Ronald Rietman, Igor Shparlinski, and Ludo Tolhuizen. Results on polynomial interpolation with mixed modular operations and unknown moduli. Cryptology ePrint Archive, Report 2015/1003, 2015. http://eprint.iacr.org/2015/1003.

[21] 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.
[22] 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.
[23] 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.
[24] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 12–24, New York, NY, USA, 1989. ACM.
[25] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), 1993. RFC1510.
[26] B. Laurie, A. Langley, and E. Kasper. Certificate transparency. RFC 6962, RFC Editor, June 2013.
[27] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2010.
[28] 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.
[29] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978.
[30] 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.
[31] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.
[32] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*, pages 135–148, 2000.
[33] 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.
[34] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

APPENDIX A
LIST OF ABBREVIATIONS

PKI	Public-Key Infrastructure
KPS	Key Pre-distribution
ECC	Elliptic Curve Cryptography
TLS	Transport Layer Security
SSH	Secure Shell
IPSec	Internet Protocol Security
ECDH	Elliptic Curve Diffie Hellman
NIST	National Institute of Standards and Technology
ETSI	European Telecommunications Standards Institute
CA	Certification Authority
IoT	Internet of Things
TTP	Trusted Third Party
KPS	Key Pre-distribution
MAC	Message Authentication Code
LWE	Learning With Errors
R-LWE	Ring - Learning With Errors
KDC	Key Distribution Center
IKEv2	Internet Key Exchange
HIMMO	Hiding Information and Mixing Modular Operations
MITM	Man-in-the-Middle

APPENDIX B
HIMMO

Although the body of this paper uses a simplified description of HIMMO as introduced the Background section, this appendix details HIMMO as described in [19] for the sake of completeness .

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. A matrix $\boldsymbol{A}$ is denoted by a capital bold letter. A vector $\boldsymbol{v}$ is denoted by a small bold letter. HIMMO has several system parameters that determine the security and performance of HIMMO, viz. b, the key length, N, an odd reduction integer (of bit length $3b$), and integers t and m. We remark that t 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:

- For $1 \leq i \leq m$, a secret, random integer q_i, s.t. $\langle q_i \rangle_{2^b} = \langle N \rangle_{2^b}$ and $N - 2^{2b} \leq q_i < N$.
- For $1 \leq i \leq m$, a secret, random symmetric $t \times t$ matrix $\boldsymbol{R}_i$ over $\mathbb{Z}_{q_i}$.

In the **key material extraction phase**, the TTP provides node $x \in [0, 2^b)^t$ with a secret vector $\boldsymbol{s}_x = \langle \sum_{i=1}^{m} \langle \boldsymbol{x} \boldsymbol{R}_i \rangle_{q_i} \rangle_N$.

A later phase comprises a **key establishment protocol** in which if node x wishes to communicate with node y, it computes the key $s_{x,y} = \langle \langle \boldsymbol{s}_x \boldsymbol{y}^T \rangle_N \rangle_{2^b}$ and determines $k_{x,y} = \left\lfloor \frac{s_{x,y}}{2^u} \right\rfloor$ and $h = \langle s_{x,y} \rangle_{2^u}$ where $u = \lceil \log_2(4mt+1) \rceil$. Then node x sends h to node y. Finally, node y computes $k_{x,y}$ from $k_{y,x}$ and h as $\left\lfloor \frac{\langle k_{y,x} + \lambda N \rangle_{2^b}}{2^u} \right\rfloor$ where $\lambda = \{N^{-1}(h - k_{y,x} \}_{2^u}$.

See [19], [16] for more details.

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, can obtain a node's short identity as $x_{short} = H(certificate)$, where H is a public hash function. The components in $\boldsymbol{x}$ can be obtained from the short identity as $\boldsymbol{x}[k] = H(x_{short}|k)$ where $|$ indicates concatenation. 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 x'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 x.

Using multiple TTPs was introduced by Matsumoto and Imai [28] for KPS and can also be elegantly supported by HIMMO [19] [16]. 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 functions 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.