Efficient Quantum-Resistant Trust Infrastructure based on HIMMO

Oscar García Morchón
Philips Group Innovation, Research
Eindhoven, The Netherlands
oscar.garcia@philips.com

Sauvik Bhattacharya
Philips Group Innovation, Research;
RWTH Aachen University
Eindhoven, The Netherlands; Aachen, Germany
sauvik.bhattacharya@philips.com

Ronald Rietman
Philips Group Innovation, Research
Eindhoven, The Netherlands
ronald.rietman@philips.com

Ludo Tolhuizen
Philips Group Innovation, Research
Eindhoven, The Netherlands
ludo.tolhuizen@philips.com

Jose Luis Torre-Arce
Philips Group Innovation, Research
Eindhoven, The Netherlands
jose.luis.torre.arce@philips.com

Maarten Bodlaender
Philips Group Innovation, Research
Eindhoven, The Netherlands
maarten.bodlaender@philips.com

ABSTRACT

Secure Internet communications face conflicting demands: while advances in (quantum) computers require stronger, quantum-resistant cryptographic algorithms, the Internet of Things demands better-performing protocols. Finally, communication links usually depend on a single root-of-trust, e.g., a certification authority which forms 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. Our solution is resilient to the capture of individual roots of trust without affecting performance, while public-key cryptography provides 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 use-cases 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 in the TLS protocol. 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.

DOI: 10.1145/1235

Keywords

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

1. INTRODUCTION

Secure and (mutually) authenticated Internet communications are absolutely essential today, especially in scenarios involving access to secure infrastructure or networks, such as in remote procedure calls [36, 25], server-server communication [25], virtual private networks (VPN), SCADA (Supervisory control and data acquisition) networks [35], Machine-to-Machine communication and in specific Internet of Things (IoT) use-cases. However, existing security infrastructures are 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 security protocols such as 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 [9]. However, most existing candidates, such as those summarized in the ETSI report in [15], are not a direct drop-in replacement due to their large key or signature sizes. For instance, the McEliece crypto system [29] requires a 1 MByte public-key, and XMSS [6] involves signatures of tens of KBytes. *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 [22]. A *third* aspect to consider is that the scope of the Internet is increasing and millions of smart objects are getting connected that typically have limited resources (energy, computation power, storage) and communicate through resource-constrained networks. *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, while (ii) still being practical and performant, in particular taking into account new application areas such as the IoT. Furthermore, (iii) the architecture should be resilient to the capture of roots of trust in order to ensure confidentiality, integrity, and authenticity of communications. At the same time, (iv) the solution should be able to accommodate trade-offs between the right to 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 [17][21] to address these problems, specifically targeted at use-cases strictly requiring privacy and (mutual) authentication, such as the ones mentioned at the beginning of this section. In our architecture, long-term public keys and parameters of devices are certified by means of HIMMO (by HIMMO TTPs) so that any pair of devices can efficiently authenticate themselves 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, in-band, initial distribution of secret HIMMO key material. Furthermore, the use of public-key cryptography (that is independent from HIMMO and thus cannot be compromised by its TTPs) in combination with HIMMO to protect the communication between parties ensures confidentiality against eavesdropping by Trusted Third Parties (TTPs), in strong contrast to how it typically is in the case of Key Pre-distribution. In contrast to a fully hierarchical PKI in which each public-key is usually certified by a single Certification Authority – thus forming a single point of failure – our architecture uses multiple TTPs to ensure robustness. For instance, multiple TTPs can be placed at different locations and run by independently affiliated authorities so that the capture of individual TTPs does not break the whole system.

Our architecture can be further configured to target different use cases, relying on HIMMO's lightweight nature and performance to scale from the constrained IoT to more traditional computer networks and the Internet. A performant configuration for use cases in which operational performance is critical (e.g., IoT or real-time communication links) can be created by using *only* HIMMO to provide a non-interactive authenticated key exchange. A generic, hybrid configuration for the Internet can be created by using a combination of HIMMO and (quantum-resistant) public-key cryptography to secure communication between parties, where the independence of the public-key cryptosystem from HIMMO protects against possible eavesdropping by TTPs. The use of multiple TTPs to certify long-term public keys for authentication in this configuration further makes it robust against the compromise of individual TTPs. Another configuration using only HIMMO to establish secure communication in the architecture is possible, specifically for use cases requiring lawful interception, but with the guarantee of multiple TTPs (that would have to unanimously cooperate to intercept communications) to preserve a trade-off with the right to privacy of legitimate users.

The remainder of this paper is structured as follows. After summarizing background work in Section 2, 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 public keys and used to create a confidential, authenticated TLS session, demonstrating integrations both with pre-quantum (ECC) and post-quantum (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 is built on a number of Certification Authorities (CAs) in charge of certifying 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 [27] 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 capable of monitoring all communications.

Architectures based on Key Pre-distribution (KPS) 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 key material that depends on R and her identity. This key material is thus unique to Alice and is kept private by her. Once two 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 common 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 [9] of the standardization of **quantum-resistant algorithms** that should replace 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] and Frodo [4], McEliece [29], Rainbow [11] and XMSS [6]. 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 [17],[19],[21]. In its basic version (see Appendix B for further details), there is a single TTP owning secret root key material $R(x, y)$. During enrollment, the TTP provides Alice with a secret function $\boldsymbol{s}_A$ 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 $\boldsymbol{s}_A(B) \approx \boldsymbol{s}_B(A)$ (with the small difference between them reconciled by exchanging some non-secret, *helper* data). HIMMO's cryptanalysis relies on lattice techniques [21], and therefore, it is considered a potential quantum-resistant candidate. Although it is not a public-key scheme, it enables features such as key generation $K_{A,B} = \boldsymbol{s}_A(B)$ using publicly-verifiable identities in the place of B. If two parties engage in 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 also be easily verified.

To the best of our knowledge, HIMMO is the only known scheme with these properties. Considering related schemes, the work in [31] can be considered as a collusion-resistant KPS, but it is not quantum-resistant. The work in [33] based on the above protocol [31] achieves authenticated key-exchange, however, being based on pairings it is less efficient than HIMMO and furthermore not quantum-resistant. Furthermore, HIMMO supports multiple TTPs efficiently, where an entity can enroll with multiple TTPs, receive key material from each of them, and obtain its final keying material as the (component-wise) combination of these individual key materials. Thus each of the TTPs only has a partial knowledge of the final key material of the entity, that is used to generate the secret key, preventing a single point of failure as in a single TTP scenario.

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 their supported ciphersuites and engage in a protocol, usually based on public-key cryptography, that enables key establishment, (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 [13]. In order to prepare for this, there have been publications in which quantum-resistant candidates have been integrated with TLS, including TLS-LWE [5] or DTLS-HIMMO [19].

In the IETF there are two initiatives to make TLS and Internet Key Exchange (IKEv2) more quantum-resistant. The hybrid TLS handshake in [32] extends TLS in a modular way with quantum-resistant key encapsulation or transport schemes, such as NTRUEncrypt [24]. In this hybrid handshake, key establishment 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 to have the final secret that protects the actual data, derived from both the classical and the post-quantum schemes, so that an attacker who 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 the greater urgency of protecting key establishment. Furthermore, there are no good signature candidates exhibiting a good security confidence and performance.

The Internet Draft in [16] aims at making the Internet Key Exchange protocol IKEv2 quantum-resistant by using pre-shared keys next to a standard key exchange, aiming for authenticated and confidential communication between an initiator and responder. To this end, the authors of [16] 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, [16] does not cover how the pre-shared keys can be pre-distributed in an efficient way.

3. GOALS & PROBLEM STATEMENT

In this section, we first describe the threat model that we consider for our security architecture (Section 3.1). Then we state the goals that our security architecture aims to achieve, in the context of security, operational and performance requirements in Section 3.2, and formally describe the problem addressed by this work in Section 3.3.

3.1 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 [8]. *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., [22] or [30]. *Finally*, these attackers are also able to compromise end devices.

With these capabilities, the goals of the attacker can be not only to eavesdrop (passive attack) on one specific or multiple communication links; but also to impersonate (active attack) a person and send fake messages on her behalf.

3.2 Design Goals

In order to prevent attackers possessing capabilities such as those described in Section 3.1 from achieving their goals, we would like a security architecture with the following **security features**:

- Confidentiality and integrity of communication links.
- Identification and authentication of all communicating parties.
- Resilience against the capture of some roots of trust.
- Protocols should be quantum-resistant.
- Advanced security features such as forward-secrecy.

Furthermore, we also wish to address the trade-off between the right to privacy and the need for lawful interception [14] of communications to prevent or resolve crimes.

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

- The architecture should facilitate a smooth transition towards quantum resistant solutions with minimal changes and overhead.
- The architecture should be applicable to as many use cases and applications as possible, e.g., be applicable not only to secure networks and traditional Internet applications but also to new ones, e.g., Internet of Things.
- Apart from security goals, performance should be as optimal 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.

3.3 Problem Statement

When we consider the above design goals in Section 3.2, the attack model, and the current state-of-the-art reviewed in Section 2, the problem statement of this paper is fourfold.

Firstly, despite there being many potential quantum-resistant candidates for key-exchange and authentication, none of them are a drop-in replacement of existing primitives, creating an urgent requirement for practical, secure and quickly-deployable quantum-resistant primitives. Current quantum-resistant signatures are especially bulky or operationally cumbersome to use. For instance, hash-based signatures involve an overhead of tens of KBs and in some settings require keeping some state [7], making private keys a critical resource, and key management extremely risky.

Secondly, the security of communication links in authenticated Internet protocols usually fully depends on a single root of trust, e.g., a certification authority, that is a single point of *complete* failure. This trust responsibility must be distributed among multiple roots of trust without affecting performance, resulting in resilience to the compromise of individual roots of trust.

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 security requirements and also exhibits good performance.

4. DESIGN

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

Firstly, use multiple 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, while preventing single points of failure and thus adding robustness.

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. The only other schemes known to resemble a fully collusion-resistant KPS are [31, 33, 10], however they are 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.

4.1 Roots of Trust in a PQ World

A fully general secure, authenticated 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 since they are 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 to prevent TTPs from becoming single points of failure. 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 efficient quantum-safe public-key authentication schemes.

Therefore, our trust infrastructure relies on multiple HIMMO TTPs that maintain secret root key material for some given security parameters. These TTPs are managed by multiple, independently located and affiliated organizations so that a security breach in one of them does not affect the rest but if required for lawful interception of communications, only a coalition of *all* TTPs could monitor communications. TTPs are in charge of certifying parameters of entities that wish to enroll and of distributing key material to them.

4.2 Enrollment & Certification

The first phase of our architecture's operation is an enrollment and certification phase during which a party can talk to any number of HIMMO roots of trust, expose its identification information and public-key, get them certified, and securely receive 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.

It is imperative that the HIMMO key material for a party remains secret. While for IoT scenarios, an out-of-band channel for distributing key material is a good option [18], this is not the case for the Internet. The challenge is therefore to securely distribute such key material over an open network. We propose to solve this using the following general enrollment approach: When an entity, Alice, wants to get enrolled

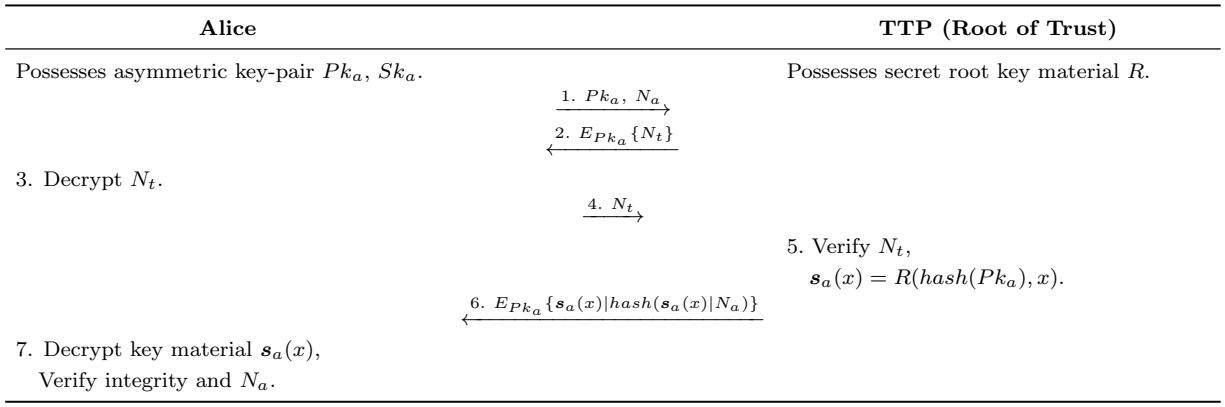

Figure 1: Enrollment and Certification of a communicating entity *Alice* with a root of trust, i.e., a HIMMO TTP, resulting in Alice getting her long-term public-key Pk_a certified and securely receiving private HIMMO key material $\boldsymbol{s}_{Pk_a}(x)$ that is bound to Pk_a. This key material is created by the HIMMO TTP using its secret root key material R. N_a and N_t are random, ephemeral nonces that prevent replay attacks and enable verification of the enrollee's identity respectively.

in the system, she generates a public/private key-pair (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 Pk_a. This is demonstrated in Step (1) of Figure 1, that depicts Alice's enrollment with a single TTP.

When the TTP receives Alice's information, it can further verify it for consistency and optionally add additional parameters, e.g., an expiration date, creating a certificate of Alice's identity. Before this, the TTP ascertains Alice's identity (to thwart would-be impersonators) by using the public-key received to encrypt and send a random, fresh nonce N_t to Alice (Figure 1, Step (2)), that she must then decrypt and send back to the TTP (Steps (3,4)). Next, given Alice's certificate, the TTP computes a HIMMO identity for Alice by hashing the certificate (containing *at least* her public-key Pk_a) and uses this to extract the corresponding HIMMO key material $\boldsymbol{s}_a(x)$ for her in Step (5). Therefore, through Alice's identity and in turn through the certificate containing Pk_a (or through Pk_a directly as in Figure 1), $\boldsymbol{s}_a(x)$ generated for Alice is now cryptographically bound to Pk_a. In Step (6), the TTP can use Alice's public-key Pk_a to encrypt and send the computed HIMMO key material $\boldsymbol{s}_a(x)$ to her so that only the owner of Pk_a can decrypt $\boldsymbol{s}_a(x)$. The TTP includes a hash of the key material in the encrypted message along with a nonce N_a originating from Alice that acts as an authentication token. This is decrypted and verified by Alice in Step (7), following which she can use her new HIMMO key material.

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 either without being combined, or combined into *one single instance* of HIMMO key material by adding the components/coefficients of multiple HIMMO key materials received from different TTPs, one by one. In case of multiple TTPs, some communication between them may be required to ensure that they all generate the same HIMMO identity for Alice.

4.3 Secure Operation

Following the successful enrollment of two entities Alice and Bob with one or multiple roots of trust, they should be able to establish a secure communication session in the above setting, i.e., each of them has public keys Pk_a and Pk_b bound to private HIMMO key materials $\boldsymbol{s}_a(x)$ and $\boldsymbol{s}_b(x)$. In case of multiple TTPs, these are combinations of the individual key materials generated by the TTPs involved. We believe that many security protocols can be designed in general (a few examples of which can be found in Appendix C), through the following 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.

The handshake between Alice and Bob resulting in a confidential, authenticated channel is an adapted form of the Bellare-Rogaway AKEP[1] [3], with the final session key that protects both confidentiality and authentication of the communication between Alice and Bob being derived from an additional random key established using asymmetric cryptography, apart from the HIMMO key (and thus independent from HIMMO and its TTPs).

Authentication of public keys happens without traditional digital signatures, but HIMMO. The HIMMO key material $\boldsymbol{s}_a(x)$ of a party A is cryptographically bound to its identity,

[1]authenticated key-exchange protocol

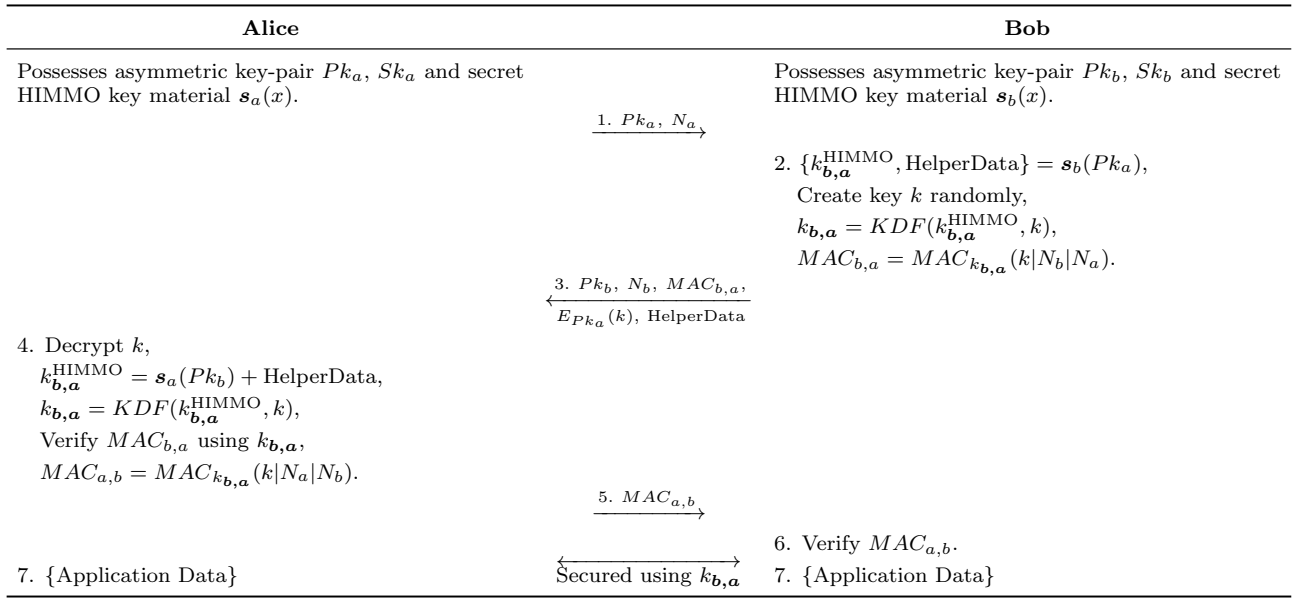

Figure 2: Secure Operation for establishing a confidential, authenticated channel between two parties *Alice* and *Bob*. Authentication is achieved using HIMMO key material bound to long-term, certified public keys. Confidentiality is achieved using a combination of HIMMO and a shared session key transported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

which in turn is bound to the party's public-key Pk_a. Successful agreement of a HIMMO key between two parties using authentic key materials and authentic public-keys results in a successful mutual authentication.

Prevention of single points of failure (as in traditional PKI) is accomplished by performing this authentication with key material generated by multiple TTPs, preventing impersonation of public keys even if some TTPs are compromised. Furthermore, this does not affect performance due to the nature of HIMMO's operation while using multiple TTPs. The use of asymmetric-cryptography for key exchange ensures that TTPs cannot access the communication and enables forward-secrecy for advanced use cases, e.g., communication over the Internet.

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 Pk_a to Bob, along with a fresh nonce N_a. In Step (2), Bob computes the HIMMO key $k^{\text{HIMMO}}_{\boldsymbol{b,a}}$ using HIMMO and additionally generates a secret, random key k. Bob then securely combines both keys into a session key $k_{\boldsymbol{b,a}}$, using it to compute a Message Authentication Code (MAC) based on the random key k, the nonce N_a received from Alice and another nonce N_b locally generated. Next in Step (3), Bob replies to Alice exchanging his long-term public-key Pk_b, his own nonce N_b, 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 HIMMO key $k^{\text{HIMMO}}_{\boldsymbol{b,a}}$ 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 random key k, Alice independently regenerates $k^{\text{HIMMO}}_{\boldsymbol{b,a}}$, 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 $k^{\text{HIMMO}}_{\boldsymbol{b,a}}$, the same random key k and thereby the same session key $k_{\boldsymbol{b,a}}$. Finally, in Step (5) Alice sends her MAC to Bob so that he can verify Alice's long-term public-key and the random key k in Step (6). Next, Alice and Bob can proceed to communicate in a secure, authenticated way in Step (7), using the final key $k_{\boldsymbol{b,a}}$ created by combining $k^{\text{HIMMO}}_{\boldsymbol{b,a}}$ 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. To illustrate this, Appendix C depicts three embodiments of this handshake – a generic version with any suitable public-key key-establishment scheme PKE and key pre-distribution scheme KPS, a version instantiated with the post-quantum NTRU key-exchange and with HIMMO and finally, a final version instantiated with the forward-secure, classic ECDH key-agreement and with HIMMO as the KPS scheme.

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. We use HIMMO in our proof of concept since it enables efficient distribution of pairwise keys, implicit verification of credentials, and supports multiple TTPs/roots of trust. We further use ECDH and NTRUEncrypt [24] as the pre- and post-quantum cryptographic primitives for key-establishment, and ECDSA as the pre-quantum cryptographic primitive for authentication.

Client		Server
1. ClientHello = $\text{Ciphersuites}_\text{c}$ + TTPList_c[1]+ . . .		
	2. ClientHello →	
		3. ServerHello = $\text{Ciphersuites}_\text{s}$ + TTPList_s + . . .
	← 4. ServerHello	
	⋮	
5. $\{k^{\text{HIMMO}}_{\mathbf{c,s}}, \text{HelperData}\} = \boldsymbol{s}^1_c(QshPk_s) + \boldsymbol{s}^2_c(QshPk_s) + \ldots$		
6. ClientKeyExchange = $\{E_{\text{ClassicPk}_\text{s}}(\text{ClassicS}) \vert E_{\text{QshPk}_\text{s}}(\text{CliS}) \vert \text{HelperData}\}$[2]		
	7. ClientKeyExchange →	
		8. Decrypt ClassicS, CliS.
	⋮	
		9. $k^{\text{HIMMO}}_{\mathbf{c,s}} = \boldsymbol{s}^1_s(QshPk_c) + \boldsymbol{s}^2_s(QshPk_c) + \ldots +$ 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.		
16. {Application Data}	↔ Secured using MasterSecret	16. {Application Data}

[1] List of supported HIMMO TTP IDs as per RFC 6066 [12].
[2] ClassicS, CliS are the classic secret and client QSH (assuming [32]) secret respectively.
[3] Pre-master Secret used to generate the TLS Master Secret by means of the TLS pseudo-random function PRF.

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.

Figure 3 shows the complete TLS handshake between a client, *Client*, and a server, *Server* extended with our design in Section 4. The required additions to support our extensions 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 [12] since it defines extensions to the *ClientHello* and *ServerHello* messages in which supported certification authorities are exchanged. 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 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 [12], 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, they then exchange certificates or public keys whose hashes are HIMMO identifiers for which they already possess HIMMO key material. If only classical ciphersuites are considered, then such certificates correspond to classical primitives, e.g., ECDSA, that are exchanged normally via the *ClientCertificate* and *ServerCertificate* messages. If post-quantum keys are supported, then an extension header in the *ClientHello* message is used (as specified in the hybrid TLS handshake Internet Draft [32]) to exchange additional, post-quantum public keys. The exchanged certificates or public keys and corresponding 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 $k^{\text{HIMMO}}_{\boldsymbol{c,s}}$ 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 $k^{\text{HIMMO}}_{\boldsymbol{c,s}}$ 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 that we exchange via the *ClientKeyExchange*. This is the only protocol field that would require standardization.

To utilize the combined security of HIMMO, the classic and/or post-quantum key-establishment schemes, the TLS Master Secret used for protecting the TLS record layer is computed from a Pre-master secret *PMS* that is a combination of the classical key *Classic*, the post-quantum key

CliS and the HIMMO key $k_{c,s}^{\text{HIMMO}}$. This is similar to [32] and [16]. Since the HIMMO key $k_{c,s}^{\text{HIMMO}}$ is one of the inputs to the Master Secret in Steps (10) and (11) via the Pre-master Secret *PMS*, it ensures the confidentiality and authentication of the messages exchanged in the record layer. Most importantly, it also implicitly verifies the certificates and public keys of Client and Server since the Master Secret is used in the computation of a Message Authentication Code in the *Finished* message, that both parties must independently compute, exchange and verify (in steps (13) and (15)). This MAC is thus bound to the $k_{c,s}^{\text{HIMMO}}$ that both parties independently 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 during the Secure Operation phase (see Figure 2). The MAC's verification by Server and Client will fail unless they computed the right $k_{c,s}^{\text{HIMMO}}$ 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 authentication using HIMMO can be done using *multiple TTPs*, while also being capable of forward-secrecy and non-repudiation (through the added incorporation of standard digital signatures). Lawful interception use-cases would require the client and server to exclude the exchange of public keys during operation and thus rely only on HIMMO and multiple TTPs to provide a proper privacy trade-off.

6. EVALUATION

In this section, we begin with an analysis of the security provided by our hybrid architecture in Section 6.1, followed by a quantitative evaluation of its performance in Section 6.2. Section 6.3 presents a qualitative discussion of the architecture's contributions and quantitatively compares it with the state of the art.

6.1 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 (and verification) of freshly-generated nonces. A preventive measure against impersonators trying to enroll with the root of trust is achieved by the use of encrypted nonces at the start of Enrollment. 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 and to prevent recovery of an enrolling party's secret HIMMO key material by a quantum-capable attacker (leading to identity impersonation [1]). Using classic public-key cryptography suffices *only* for the short-term to ease widespread adoption.

Next, we discuss the security of the Secure Operation handshake between two parties. *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 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. In [32], it is argued using the Leftover Hash Lemma [26] that if a secret is composed of two other, independent secrets, an attacker cannot recover the combined secret as long as one of the constituent secrets is unknown to him. As a result, even if the attacker possesses quantum computing capabilities and is able to recover k, he fails to learn the key $k_{b,a}^{\text{HIMMO}}$, thus preventing him from learning the session key $k_{b,a}$. In [21], 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 random key k between the communicating parties during the Secure Operation phase (see Figure 7 in Appendix C for an instantiation).

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 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.

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 1: 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.

6.2 Performance Evaluation & Benchmarks

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 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 `x86_64` architecture with full optimization (`/0x`) using the Microsoft Visual C compiler.

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 after adding HIMMO next to them using a single TTP. The HIMMO parameter b is chosen to be 32. Several HIMMO systems are used in parallel to obtain a 256 bit key. The additional time or bandwidth consumption is minimal while enabling the verification of public keys and generation of a symmetric-secret. Furthermore, the increase in running times of the TLS handshake for the classic ECDHE_ECDSA and the post-quantum ciphersuite NTRU_RSA, when extended with HIMMO where the HIMMO key is computed from key material assigned by three TTPs (as per the protocol in Section 5) is minimal. Each TTP is managed by a different organization or has independent locations, thus preventing a single point of failure. 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 [21]; 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 $t = 4000$. A further optimization is possible for the multiple-TTP implementation: instead of sequentially creating and combining three HIMMO sub-keys, it is possible to generate a single HIMMO key after adding the coefficients/components of the key materials obtained from different TTPs. 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.

[21, 17] present insights into the efficient, lightweight performance of HIMMO. It is demonstrated that an increase of the HIMMO security parameter t 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 [21], still 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 affordable. IoT applications have different performance criteria: communication bandwidth, energy consumption, timing, and memory. For all of these HIMMO is optimal. By construction, HIMMO has low communication and energy needs since it is identity-based, and since wireless communication consumes most of the energy budget. Timing is also optimal, although key material size is on the high side. Even without optimizations, memory is the least relevant criteria: IoT devices are getting bigger memories; computers and phones have sufficient memory capabilities.

6.3 Discussion & Comparison

The solution proposed in this paper exhibits advantageous properties compared with other architectures as shown in Table 2. *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**. *Secondly*, the use of an independent public-key cryptosystem ensures **secure distribution of key material** during the initial enrollment and prevents eavesdropping by rogue TTPs during Secure Operation. The architecture supports **multiple roots of trust** (based on HIMMO TTPs) with low overhead, ensuring that the system remains secure even if individual roots of trust are compromised. Operationally, if a secure channel between parties is established based on HIMMO *only*, this results in a non-interactive method for key agreement and authentication, ideal for **IoT scenarios**. Together with the incorporation of multiple TTPs, this configuration of our architecture (i.e., based on *only* HIMMO) provides a solid solution for use cases in which **lawful interception** of communication is required. At the same time, this does not infringe upon the **right to privacy** of legitimate communications since it would require a unanimous cooperation among all TTPs involved in the communication, preventing possible abuses of power.

Supported Features	Traditional PKI	Kerberos	KPS	QSH-TLS [32]	DTLS-HIMMO [19]	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	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)
Scalability	High	Medium	Medium	Medium	High	Higher

Table 2: Comparison of the hybrid infrastructure presented in this paper with existing security architectures, for a number of security and operational features.

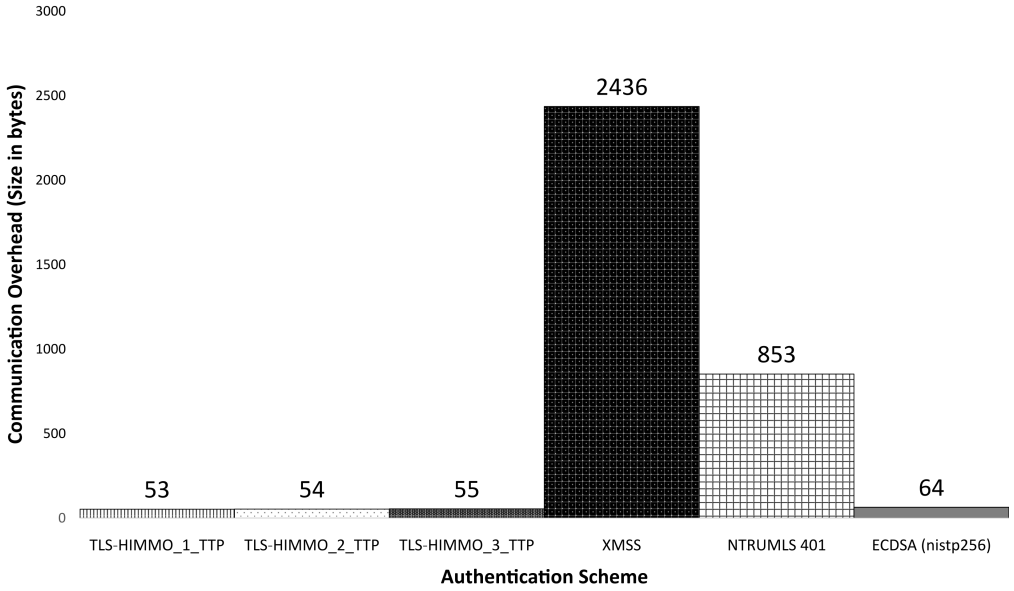

Figure 4: Comparison of communication overhead in TLS authentication phase when using: (a) HIMMO key generation-based authentication with one TTP, (b) with two TTPs, (c) with three TTPs (all using 256-bit HIMMO key with a large security parameter $t = 4000$), (d) XMSS with 82 bit security, (e) NTRUMLS with parameter set 401 [23] and 112 bit security, (f) classic ECDSA with the curve `nistp256`. Data exchanged for authentication purposes when using our HIMMO-based solution amounts to only HIMMO helper data for key reconciliation.

Considering authentication, 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, allowing a node to verify public parameters of other devices with very low overhead while supporting multiple roots of trust. Upgrading to any public-key cryptosystem is simple, through the use of the corresponding digital certificate or public-key as the HIMMO identity.

Thus, the proposed architecture is highly **scalable**, even if an out-of-band channel is not available as is the case in the Internet, and is applicable to a wide range of use cases. In contrast, the hybrid TLS handshake in [32] focuses on traditional Internet applications only and does not support authentication by multiple roots of trust. The DTLS-HIMMO work [19] addresses IoT use cases but lacks a solution for in-band enrollment or the capability of providing forward-secrecy.

Considering post-quantum authentication, we quantify the gains of our solution by comparing it with two signature schemes considered quantum-resistant, namely **XMSS** [6] and **NTRUMLS** [23]. XMSS with an h value equal to 8 (i.e., the height of the XMSS hash tree) allows creating at most 256 signatures, each approximately 2436 Bytes long and having a signature verification time of 1.95 ms [15]. 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 [15]. 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 sig-

natures with that exchanged by the authentication in 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 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, since 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 essentially grows logarithmically with n. For more details about the exact nature of HIMMO helper data that leads to this desirable behavior, see Appendix B.

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 6.2).

7. 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. Due 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.

8. REFERENCES

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

[2] E. Alkim, L. Ducas, T. Poeppelmann, and P. Schwabe. Post-Quantum Key Exchange – A New Hope, 2015. Cryptology ePrint Archive, Report 2015-1092.

[3] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution*, pages 232–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[4] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. 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.

[5] J. 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.

[6] 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.

[7] D. Butin, S.-L. Gazdag, and J. Buchmann. Real-world post-quantum digital signatures. In *Cyber Security and Privacy*, pages 41–52. Springer, 2015.

[8] S. Checkoway, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, H. Shacham, and M. 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, Aug. 2014. USENIX Association.

[9] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on Post-Quantum Cryptography. NISTIR8105, 2016. http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf.

[10] Y. Chen, Q. Huang, and Z. 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.

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

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] O. Garcia-Morchon, R. Rietman, S. Sharma, L. Tolhuizen, and J.-L. T. 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.

[19] O. Garcia-Morchon, R. Rietman, S. Sharma, L. Tolhuizen, and J.-L. T. 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.

[20] O. Garcia-Morchon, R. Rietman, I. Shparlinski, and L. 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] O. García-Morchón, R. Rietman, L. Tolhuizen, J. Torre-Arce, M. Lee, D. Gómez-Pérez, J. Gutierrez, and B. Schoenmakers. Attacks and parameter choices in HIMMO, 2016. Cryptology ePrint Archive, Report 2016-152.

[22] 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`.

[23] J. Hoffstein, J. Pipher, J. Schanck, J. Silverman, and W. Whyte. Transcript Secure Signatures Based on Modular Lattices. In M. Mosca, editor, *Post-Quantum Cryptography*, volume 8772 of *Lecture Notes in Computer Science*, pages 142–159. Springer International Publishing, 2014.

[24] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring-based public key cryptosystem. In J. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Berlin Heidelberg, 1998.

[25] IBM. IBM AIX: An executive guide to IBM's strategy and roadmap for the AIX Operating System on Power Systems, An IBM White Paper. Technical report, IBM, December 2015.

[26] 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.

[27] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), 1993. RFC1510.

[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] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978.

[30] J. Prins. Diginotar certificate authority breach *Operation Black Tulip*, September 2011. www.enisa.europa.eu/media/news-items/operation-black-tulip.

[31] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*, pages 135–148, 2000.

[32] J. 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.

[33] M. Scott. Migrating SOK to a type-3 Pairing. Technical report, Miracl Labs, October 2016.

[34] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[35] C. R. Taylor, C. A. Shue, and N. R. Paul. A deployable scada authentication technique for modern power grids. In *Energy Conference (ENERGYCON), 2014 IEEE International*, pages 696–702, May 2014.

[36] R. Thurlow. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 5531, RFC Editor, May 2009. http://www.rfc-editor.org/rfc/rfc5531.txt.

APPENDIX

A. LIST OF ABBREVIATIONS

PKI	Public-Key Infrastructure
KPS	Key Pre-distribution
ECC	Elliptic Curve Cryptography
TLS	Transport Layer Security
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
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
KDC	Key Distribution Center
IKEv2	Internet Key Exchange
HIMMO	Hiding Information and Mixing Modular Operations
MITM	Man-in-the-Middle

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 [21] 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 $k_{x,y} = \left\lfloor \frac{s_{x,y}}{2^u} \right\rceil$ 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 $\left\lfloor \frac{\langle k_{y,x} + \lambda N \rangle_{2^b}}{2^u} \right\rceil$ where $\lambda = \{N^{-1}(h - k_{y,x}\}_{2^u}$.

See [21], [17] 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 [21] [17]. 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.

C. SECURE OPERATION INSTANTIATIONS

Following the description of the Secure Operation step in Section 4.3, we describe here three embodiments to demonstrate the flexibility of its design. Figure 5 depicts a generic form with any suitable public-key key-establishment and key pre-distribution scheme; Figure 6 depicts one with NTRU and HIMMO, and Figure 7 depicts an instantiation with ephemeral ECDH and HIMMO. In the final embodiment, a combination of ephemeral ECDH and HIMMO achieves key-establishment (achieving forward secrecy and quantum-safe confidentiality) while HIMMO key material bound to static, long-term ECDSA public keys achieves quantum-safe authentication.

Alice		Bob
Possesses asymmetric key-pair Pk_a, Sk_a for scheme PKE and secret KPS key material $\boldsymbol{s}_a(x)$ bound to Pk_a.		Possesses asymmetric key-pair Pk_b, Sk_b for scheme PKE and secret KPS key material $\boldsymbol{s}_b(x)$ bound to Pk_b.
	1. $Pk_a,\ N_a$ $\longrightarrow$	
		2. $\{k^{\text{KPS}}_{\boldsymbol{b,a}}, \text{ReconciliationData}\} = \boldsymbol{s}_b(Pk_a)$, Create key k randomly, $k_{\boldsymbol{b,a}} = KDF(k^{\text{KPS}}_{\boldsymbol{b,a}}, k)$, $MAC_{b,a} = MAC_{k_{\boldsymbol{b,a}}}(k\|N_b\|N_a)$.
	$\longleftarrow$ 3. $Pk_b,\ N_b,\ MAC_{b,a},$ $E_{Pk_a}(k),\ \text{ReconciliationData}$	
4. Decrypt k, $k^{\text{KPS}}_{\boldsymbol{b,a}} = \boldsymbol{s}_a(Pk_b) + \text{ReconciliationData}$, $k_{\boldsymbol{b,a}} = KDF(k^{\text{KPS}}_{\boldsymbol{b,a}}, k)$, Verify $MAC_{b,a}$ using $k_{\boldsymbol{b,a}}$, $MAC_{a,b} = MAC_{k_{\boldsymbol{b,a}}}(k\|N_a\|N_b)$.		
	5. $MAC_{a,b}$ $\longrightarrow$	
		6. Verify $MAC_{a,b}$.
7. {Application Data}	$\longleftrightarrow$ Secured using $k_{\boldsymbol{b,a}}$	7. {Application Data}

Figure 5: Secure Operation for establishing a confidential, authenticated channel between two parties *Alice* and *Bob*. Authentication is achieved by using key material (corresponding to any suitable quantum-safe key pre-distribution scheme KPS) bound to long-term, certified public keys (corresponding to any suitable public-key key establishment scheme PKE). Confidentiality is achieved using a combination of a key established using KPS and another established using PKE.

Alice		Bob
Possesses NTRU key-pair Pk_a, Sk_a and secret HIMMO key material $\boldsymbol{s}_a(x)$ bound to Pk_a.		Possesses NTRU key-pair Pk_b, Sk_b and secret HIMMO key material $\boldsymbol{s}_b(x)$ bound to Pk_b.
	1. $Pk_a,\ N_a$ $\longrightarrow$	
		2. $\{k^{\text{HIMMO}}_{\boldsymbol{b,a}}, \text{HelperData}\} = \boldsymbol{s}_b(Pk_a)$, Create key k randomly, $k_{\boldsymbol{b,a}} = KDF(k^{\text{HIMMO}}_{\boldsymbol{b,a}}, k)$, $MAC_{b,a} = MAC_{k_{\boldsymbol{b,a}}}(k\|N_b\|N_a)$.
	$\longleftarrow$ 3. $Pk_b,\ N_b,\ MAC_{b,a},$ $E_{Pk_a}(k),\ \text{HelperData}$	
4. Decrypt k, $k^{\text{HIMMO}}_{\boldsymbol{b,a}} = \boldsymbol{s}_a(Pk_b) + \text{HelperData}$, $k_{\boldsymbol{b,a}} = KDF(k^{\text{HIMMO}}_{\boldsymbol{b,a}}, k)$, Verify $MAC_{b,a}$ using $k_{\boldsymbol{b,a}}$, $MAC_{a,b} = MAC_{k_{\boldsymbol{b,a}}}(k\|N_a\|N_b)$.		
	5. $MAC_{a,b}$ $\longrightarrow$	
		6. Verify $MAC_{a,b}$.
7. {Application Data}	$\longleftrightarrow$ Secured using $k_{\boldsymbol{b,a}}$	7. {Application Data}

Figure 6: Secure Operation for establishing a confidential, authenticated channel between two parties *Alice* and *Bob*. Authentication is achieved by using HIMMO key material bound to long-term, certified NTRU [24] public keys. Confidentiality is achieved using a combination of keys established using HIMMO and NTRU.

Alice		**Bob**
Possesses ECDH public parameters g, p and secret HIMMO key material $\boldsymbol{s}_a(x)$ (bound to long-term ECDSA public-key Pk_a).		Possesses ECDH public parameters g, p and secret HIMMO key material $\boldsymbol{s}_b(x)$ (bound to long-term ECDSA public-key Pk_b).
	$\xrightarrow{1.\ Pk_a,\ N_a}$	
		2. $\{k^{\text{HIMMO}}_{\boldsymbol{b,a}}, \text{HelperData}\} = \boldsymbol{s}_b(Pk_a)$, Choose ephemeral, random, secret b, $B = (g^b) \bmod p$.
	$\xleftarrow{3.\ Pk_b,\ N_b,\ MAC_{b,a},\ E_{k^{\text{HIMMO}}_{\boldsymbol{b,a}}}(B),\ \text{HelperData}}$	
4. $k^{\text{HIMMO}}_{\boldsymbol{b,a}} = \boldsymbol{s}_a(Pk_b) + \text{HelperData}$, Decrypt B, Choose ephemeral, random, secret a, $A = (g^a) \bmod p$, $k^{\text{DH}} = (B^a) \bmod p$, $k_{\boldsymbol{a,b}} = KDF(k^{\text{HIMMO}}_{\boldsymbol{b,a}}, k^{\text{DH}})$, $MAC_{a,b} = MAC_{k_{\boldsymbol{b,a}}}(A\|B\|N_a\|N_b)$.		
	$\xrightarrow{5.\ E_{k^{\text{HIMMO}}_{\boldsymbol{b,a}}}(A),\ MAC_{a,b}}$	
		6. Decrypt A, $k^{\text{DH}} = (A^b) \bmod p$, $k_{\boldsymbol{a,b}} = KDF(k^{\text{HIMMO}}_{\boldsymbol{b,a}}, k^{\text{DH}})$, Verify $MAC_{a,b}$.
7. {Application Data}	$\longleftrightarrow$ Secured using $k_{\boldsymbol{b,a}}$	7. {Application Data}

Figure 7: Secure Operation for establishing a forward-secret, confidential, authenticated channel between two parties *Alice* and *Bob*. Authentication is achieved by using HIMMO key material bound to long-term, static, certified ECDSA public keys. Confidentiality is achieved using a combination of keys established using HIMMO and ephemeral ECDH, thus providing forward-secrecy.