Solving the Quantum Dilemma in Public Blockchains

A Lite Paper by pQCee.com

October 2022

Abstract. Public blockchains such as Bitcoin and Ethereum face an existential crisis when faced with quantum computers. The digital signing algorithm, ECDSA, used by these blockchains is vulnerable to quantum-capable adversaries and can no longer be relied on to protect transactions and authenticate users. On the other hand, migrating blockchains to an alternative quantum-secure algorithm is a massive, resource-intensive and complex undertaking that may adversely impact users and hinder the blockchains' ability to innovate and grow.
We identify critical success factors for a successful quantum-secure solution and propose a different approach to quantum-readiness for public blockchains by augmenting the ECDSA signing process with a Proof-of-Key-Generation (PoKG) proof. The PoKG proof is secure against fake transactions created by quantum-capable adversaries and still maintain the backward-compatibility to ECDSA. We discuss the architecture needed to support the different usage scenarios and features that the PoKG solution will require to have. Our next steps are to evaluate and select a suitable platform to build and operate the PoKG solution.

Keywords: Post-Quantum Cryptography (PQC) · BIP39 · Blockchains · Elliptic Curve Digital Signing Algorithm (ECDSA).

Table of Contents

1 Introduction

Digital signing algorithms using asymmetric key cryptosystems work on the principle of a private-public key pair. In a typical communication scenario between a Sender and Receiver, the Sender generates a private key, which is kept secret, and broadcasts the corresponding public key to the Receiver. When sending a message, the Sender uses the private key to compute a digital signature on the message, and sends the digital signature with the message to the Receiver. The Receiver can use the Sender's public key to validate the digital signature to i) check that the message is actually sent by the Sender; ii) verify that the message is not modified along the way; iii) prove to others that the Sender sent the message. The security assumption for a good digital signing algorithm is that nobody, except the Sender, has knowledge of or can compute the private key, even when the public key and many valid digital signatures are made public. Popular digital signing algorithms such as Rivest-Shamir-Adleman (RSA), Digital Signing Algorithm (DSA) and Elliptic Curve Digital Signing Algorithm (ECDSA), along with their respective underlying cryptosystem – RSA, Diffie-Hellman (DH), Elliptic Curve Cryptography(ECC), rely on hard mathematical problems of integer factorization and discrete logarithm over a large finite field to ensure that the private keys can never be obtained from the public keys.

Popular blockchains such as Bitcoin [16], Ethereum [4], etc use ECDSA to protect the communication between the wallet (as the Sender) and blockchain nodes (as the Receiver). Wallets contain the ECC private keys and are used to compute the digital signature of transactions that the wallet owner wants to perform. When the blockchain nodes received a valid signed transaction, the blockchain nodes will then effect the transaction which mostly involves the transfer of assets from the associated wallet address to another payee address. Assuming wallets are securely implemented to protect the private key, nobody other than

the wallet owner can create valid signed transactions ensuring that the assets on the blockchain remain intact.

But this will all be changed when quantum deadline is reached and quantum computers become powerful enough. Shor's algorithm [20,18] running on quantum computers can exponentially speed up the process to solve integer factorization and discrete logarithm problems. This means that an adversary with a large enough quantum computer can cryptanalyze an RSA, DH or ECC public key, and obtain the corresponding private key in a matter of hours or days, thus breaking the security assumption that these digital signing algorithms hold. The implication for blockchains still using ECDSA would mean that blockchain nodes will not longer be able to differentiate between signed transactions coming from valid wallets, and signed fake transactions sent by quantum-capable adversaries, resulting in a catastrophic loss in trust and value in assets stored on blockchains.

Fortunately, quantum computers are currently not powerful enough to run Shor's algorithm to break ECDSA on blockchains. Unfortunately the National Institute of Standards and Technology (NIST) thinks that this may happen as soon as within a decade and has embarked on their post-quantum cryptography (PQC) selection exercise [1] for new algorithms. While efforts to find replacement digital signing algorithms are underway, these replacements are not backward compatible to ECDSA and their use may impact users and applications significantly [22]. Inactive users may find their assets no longer accessible and all wallet and smartcontract addresses will be changed. We should also note that any migration of algorithm on the blockchain is a massive and complex effort, requiring large amounts of resources and community involvement, and may potentially hinder the current innovation and growth trajectory that blockchains and Web3 have enjoyed over the past decade.

1.1 Contribution

So how can we best mitigate the quantum-threat in blockchains while minimizing the migration effort, innovation disruption, and user impact? We present the pQCee's approach by using a combination of

- layering a quantum-resistant proof-of-key-generation [21][1] on Bitcoin Improvement Proposal (BIP) 39 [17] to protect ECDSA signatures; and
- operating a proof-chain for the storage, retrieval and verification these proofs.

2 Critical Success Factors

In this era where frauds and scams are prevalent whenever new changes or technology are introduced, our focus is to allow for as gradual and as little change as possible while maintain financial sustainability so that the community, consisting of end-users, developers, industry participants, researchers and regulators can have the appropriate time to understand the solution, evaluate the challenges, and allocate sufficient resources to make it happen at their own pace. We list the critical success factors for the proposed solution in Table 1.

[1] Patent-pending

Table 1. Critical success factors for quantum-secure solution

No.	Category	Requirement	Description
1.	Technical	Quantum-secure	Validator nodes receiving digitally signed transactions from wallets need to be able to differentiate between genuine transactions signed by valid users and fake transactions signed by quantum-capable adversaries.
2.	Compatibility	No deadline	There should be no arbitrary deadline set by when users need to migrate. Users can migrate at their own time, and even users who do not migrate prior to the quantum deadline can still have their assets protected.
3.	Migration	No hard-fork	To reduce confusion, there should be no big-bang or fixed cut-off when transactions will stop being recognized on one chain and subsequent transactions splintering into different parallel chains.
4.	User Experience	Usage process unchanged	To increase acceptance and adoption, the overall user experience from information retrieval to wallet authentication to transaction signing (including multi-party signatures) to calling of smartcontracts should remain unchanged. The identity component (i.e. wallet or smartcontract address) which should remain intact.
5.	Sustainable	Viable operating model	The provider of such solutions should be able to build a viable financial model (e.g. transaction fee, or SaaS recurring, etc) to allow for the sustained business operations. This is to cater to the continued, long-term quantum-security needs of public blockchains.

When applying the list of critical success factors to existing solutions, we see that they fail in one or more categories. We briefly run through the known solutions below:

– *Forking from an existing blockchain.* An example of this effort is Bitcoin Post-Quantum (bitcoinpq)[2] which operates as a hard-fork at height 555,000 from Bitcoin. Users wanting protection against quantum computers will have to move to bitcoinpq by authenticating with their existing ECDSA key and then using the new quantum-resistant eXtended Merkle signature scheme (XMSS) [12] with new wallets and addresses. Since transactions after height 555,000 on bitcoin are no longer recognized, coupled with significant changes to the user-experience, this hard-fork makes it very challenging for the entire community to trust the implementation and migrate at a given time. bitcoinpq does not seem to have gone past the testnet stage.

[2] `https://bitcoinpq.org/`

- *Starting a new blockchain.* The Quantum Resistant Ledger (theqrl)[3] project adopts a different approach by starting a brand new chain using XMSS instead of ECDSA as the underlying authentication scheme. In doing so, it takes advantage of the technological features of distributed ledger technologies but without the quantum cryptographic vulnerability. Since theqrl has to build the trust and community from scratch, including convincing users to adopt the QRL token, this has been also a very challenging journey.
- *Hybrid signatures.* A hybrid approach [7,11] taken by researchers is to combine the use of a post-quantum signature on top of the existing quantum-vulnerable ECDSA algorithm so that each signature can overcome potential security weakness in the other signature. The assumption with this approach is that there will new keys, addresses and wallets that need to be generated in advance before the quantum deadline which means that inactive users will still be significantly impacted [22].

3 Design Intuition

So how do we find a solution that can provide quantum-security, and still provide sufficient backward-compatibility for a seamless user experience? We have to assume that the ECC private key can already be compromised by the adversary, which means we have to go even earlier into the key generation process.

3.1 Understanding Key Generation

BIP39 [17] is a common standard used for key generation and recovery for many blockchain wallets. During key generation, the following is performed by a valid user using the wallet:

1. *Obtaining the secret seed.*
   - If generating a new key, the wallet will first randomly choose a sequence of words from a pre-defined wordlist of 2048 words[4].
   - If recovering an existing key, the user is prompted to enter the previously generated sequence of words.
   - The secret seed can be further concatenated with an optional secret password chosen by the user. The combination of the sequence of words with the optional secret password will make up the secret seed.
2. *Key derivation.*
   - Wallet will perform a PBKDF2 key derivation [14] on the secret seed to obtain a 256bit hash result. This hash result is used as the ECC private key for signing transactions using ECDSA.
   - The ECC public key can also be computed from the ECC private key based on the FIPS186-5 standard [6].

[3] `https://www.theqrl.org/`

[4] There exists different sets of wordlists implemented, but they mostly follow the same principle as BIP39.

- The wallet address associated with the ECC private key is also computed by performing a hash[5] of the ECC public key.

3. *Key recovery.* Optionally, the user can request the wallet to display the secret seed so that the sequence of words can be copied onto a piece of paper and stored securely offline.

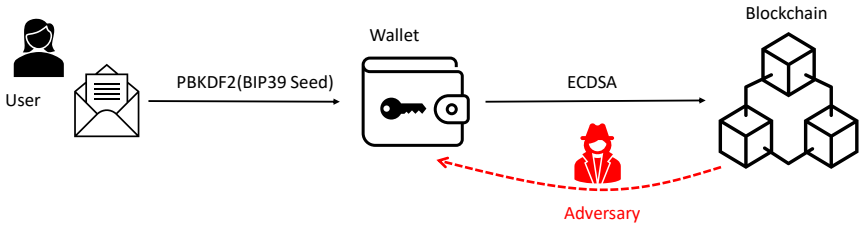

Fig. 1. Distinction between valid user and adversary

Figure 1 shows the distinction between a quantum-capable adversary, who obtains the ECC private key by cryptanalyzing the ECC public key, and the valid user, who obtains the ECC private key by PBKDF2 hashing the BIP39 secret seed. Note that it is **not** feasible for the adversary to compute the secret seed via any quantum cryptanalysis technique. This is because Grover's algorithm [10] running on a quantum computer can only maximally provide a quadratic speedup in searching for the secret seed. It is therefore recommended to use at least 24 words in the secret seed, or supplement the sequence of words with a long-enough password.

3.2 Proof-of-key-generation (PoKG)

We make use of this distinction that quantum-capable adversaries do not have knowledge of the BIP39 secret seed to enable the blockchain to differentiate between valid users and adversaries. If an incoming signed transaction includes a proof-of-key-generation (PoKG) showing that the private key used to digitally sign the transaction is derived from a secret seed, then the blockchain validator nodes will know that this transaction originated from a valid user and not from a quantum-capable adversary.

[5] Different blockchains use different hash algorithms here. For example, Bitcoin uses a combination of $SHA256()$ with $RIPEMD160()$ while Ethereum uses $KECCAK256()$

This PoKG can be achieved by applying a Multi-party computation (MPC)-in-the-head non-interactive zero-knowledge proof [13] circuit to derive the ECC private key from the BIP39 secret seed, and tying it to the existing signed transaction. Possible constructions of MPC-in-the-head that can be used include ZKBoo [8], ZKB++ [5], and KKW [15]. Since MPC-in-the-head is not based on any algebraic primitives which make it quantum-secure, the adversary will not be able to retrieve the BIP39 secret seed from the PoKG. We describe the algorithms for PoKG proof generation (see Algorithm 1) and verification (see Algorithm 2) in the appendix.

Note that the PoKG design is not limited to ECDSA and can be adapted for other ECC-based signature schemes including Edwards-curve Digital Signing Algorithm (EdDSA) [2] , Schnorr signatures including multi-signatures [19], Boneh–Lynn–Shacham (BLS) signatures [3], threshold ECDSA [9], etc.

4 Overall PoKG Solution

In order to realize the PoKG design into a workable solution, we need to implement two components:

- *PoKG Wallet Plugin.* The wallet plugin extends the functionality of the existing wallet, e.g. Metamask, to perform the PoKG proof generation as per Algorithm 1.
- *PoKG proof chain.* The proof chain serves as a smart repository to support the storage, validation and retrieval of the PoKG proofs generated.

4.1 Proposed Architecture

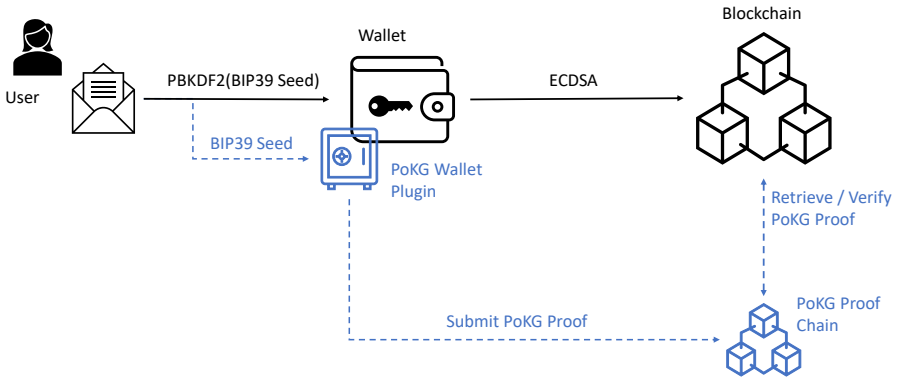

Fig. 2. Proposed solution architecture with additional components

Figure 2 shows the overall architecture showing how the additional components, marked in blue, are used to make existing public blockchains quantum-secure. Whenever the wallet is needed to sign a transaction, the BIP39 secret seed entered by the user is used by PoKG Wallet Plugin to generate the PoKG proof. This proof is submitted to the PoKG Proof Chain for storage. During transaction verification, validator nodes on the blockchain can connect to the PoKG Proof Chain to retrieve the associated PoKG proof, or rely on the PoKG Proof Chain to carry out the verification of the PoKG proof. We briefly describe the functionality of the PoKG Proof Chain in Section 4.3.

4.2 Usage Scenarios

We envision quantum-readiness for existing public blockchains to happen over three broad steps:

Step i) *Setup.* The PoKG proof chain needs to be set up to receive incoming PoKG proofs. The PoKG wallet plugin needs to be built for existing wallets. Users not using BIP39 secret seeds or using secret seeds that are of less than 24 words are encouraged to to upgrade their protection.

Step ii) *Migration.* Users will upgrade their wallets to include the PoKG wallet plugin. Validator nodes can start to check for PoKG poofs for optional additional security.

Step iii) *Post-Quantum.* Validator nodes will enforce the PoKG check for transactions. Users who have not upgraded their wallets with the PoKG wallet plugin can still do so to get access to their assets. Two possible usage scenarios how the PoKG solution can be used by existing blockchains are:

- ***Business-as-usual (BAU).*** Blockchains can continue to operate "as-is" while relying on the additional PoKG proof chain to provide the additional quantum-security protection for transactions.
- ***ECDSA Migration.*** Blockchains can choose to support new NIST PQC algorithms [1] for transaction security, and rely on PoKG for migration of assets from legacy ECDSA wallets to the new PQC wallets.

The benefits of the PoKG solution directly address the critical success factors that are discussed in Section 2. We analyze them below in Table 2.

4.3 PoKG Proof Chain

The PoKG Proof Chain is intended to run on an agnostic Layer-1 chain for the purpose of receiving the PoKG proofs from various wallets and storing them for subsequent retrieval by different blockchain nodes or verification on behalf of these nodes. It can also be used to support PoKG proofs from ECDSA signing and verification in non-blockchain use-cases such as document signing, X.509 certificates, etc.

Table 2. Critical success factors for quantum-secure solution

No.	Requirement	How is requirement fulfilled by the PoKG solution?
1.	Quantum-secure	Incoming signed transactions received by the validator nodes will have an associated PoKG proof which cannot be generated by a quantum-capable adversary
2.	No deadline	Users can continue to use their existing wallets or upgrade them with the PoKG plugin to include the PoKG proof generation. Before the quantum-deadline, validator nodes only require the ECDSA signature for transaction security. After the quantum-deadline, validator nodes will require the additional PoKG proof for transaction security, ensuring that assets are protected.
3.	No hard-fork	Existing public blockchains continue to be used and no parallel blockchain is created
4.	Usage process unchanged	When used in the "BAU" scenario, the reuse of the existing BIP39 secret seeds and ECDSA keys ensures that there are no significant changes to the user experience or wallet address. The expected impact to users are some additional processing overheads at the wallet during the computation of the PoKG proof, and the need to use BIP39 secret seeds for users who do not use them.
5.	Viable operating model	The operation of the PoKG proof chain will allow the operator to impose a storage, utilization or verification fee for each PoKG proof.

We choose to operate the PoKG Proof Chain as a distributed ledger, instead of simply relying on native storage technologies such as databases, directory services, or distributed file systems (e.g. IPFS), so that we can implement flexible pricing mechanisms, impose privacy and access controls and carry out autonomous smartcontract-like services on behalf of the signer or verifier. We identify the following features which are required for the PoKG Proof Chain:

- *Quantum-secure Access and Consensus.* The PoKG Proof Chain should not be vulnerable to quantum attacks.
- *Privacy & Anonymity Protection.* PoKG proofs that are kept in the chain should not reveal any information about the message that was signed. Necessary access controls can also be imposed for private transactions.
- *Beneficiary Party Pays.* In different use-cases, the ultimate beneficiary of a signed transaction can vary. The service fee to be imposed for the use of the PoKG Proof Chain should therefore be flexible enough to charge the signer, verifier, and/or a third-party via a per-transaction, per-seat, per-volume and/or per-period model.
- *Flexible Storage Options.* Not all PoKG proofs need to be stored in perpetuity. PoKG proofs can be allowed to expire or be batched retrieved into other storage while retaining the proof of existance of such proofs.
- *Transaction Authorization.* As an added layer of value-add, the PoKG Proof Chain can perform transaction authorization (e.g. approval by transaction

value limits, maker-checker approvals, restricting transaction types, etc) on behalf of the calling nodes or wallets.

At this point of writing, we are still evaluating suitable Layer-0 and Layer-1 blockchains which can be used as the base platform for PoKG Proof Chain. More details will be provided in a subsequent White Paper on the PoKG Proof Chain.

5 Conclusion

We have shown that although the use of ECDSA makes blockchains vulnerable to quantum attacks, it is possible to have a non-intrusive backward-compatible solution to layer in a PoKG proof to prevent quantum-capable adversaries from impersonating as valid users and creating fake transactions. This is especially relevant for public blockchains which may not have the resources to migrate their several million wallets and non-technically savvy users.

In our next steps, we will be working with and evaluating several blockchains to identify the suitable platform to build the PoKG Proof Chain, as well as to design an appropriate financial model to operate the PoKG Proof Chain and deliver the solution. For enquiries, please contact info@pQCee.com,

Acknowledgement. We would like to thank the anonymous reviewers for their comments and suggestions to this paper.

References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. rep., National Institute of Standards and Technology Gaithersburg, MD (2022)
2. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of cryptographic engineering **2**(2), 77–89 (2012)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: International conference on the theory and application of cryptology and information security. pp. 514–532. Springer (2001)
4. Buterin, V., et al.: Ethereum: A next-generation smart contract and decentralized application platform (2014)
5. Chase, M., Derler, D., Goldfeder, S., Katz, J., Kolesnikov, V., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Wang, X., Zaverucha, G.: The picnic digital signature algorithm: Update for round 2 (2019)
6. FIPS, P.: 186-4: Federal information processing standards publication. Digital Signature Standard (DSS). Information Technology Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, MD (2013)
7. Ghinea, D., Kaczmarczyck, F., Pullman, J., Cretin, J., Kölbl, S., Misoczki, R., Picod, J.M., Invernizzi, L., Bursztein, E.: Hybrid post-quantum signatures in hardware security keys. Cryptology ePrint Archive, Paper 2022/1225 (2022), `https://eprint.iacr.org/2022/1225`, `https://eprint.iacr.org/2022/1225`

8. Giacomelli, I., Madsen, J., Orlandi, C.: Zkboo: Faster zero-knowledge for boolean circuits. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 1069–1083 (2016)
9. Groth, J., Shoup, V.: Design and analysis of a distributed ecdsa signing service. Cryptology ePrint Archive (2022)
10. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Physical review letters **79**(2), 325 (1997)
11. Holcomb, A., Pereira, G., Das, B., Mosca, M.: Pqfabric: a permissioned blockchain secure from both classical and quantum attacks. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–9. IEEE (2021)
12. Hülsing, A., Butin, D., Gazdag, S., Rijneveld, J., Mohaisen, A.: XMSS: eXtended Merkle signature scheme. Online: `https://tools.ietf.org/html/rfc8391` [accessed: April 2022] (2018)
13. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. pp. 21–30. ACM (2007)
14. Kaliski, B.: PKCS# 5: Password-based cryptography specification version 2.0. Tech. rep. (2000)
15. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 525–537 (2018)
16. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Online: `https://bitcoin.org/bitcoin.pdf` [accessed: August 2022] (2008)
17. Palatinus, M., Rusnak, P., Voisine, A., Bowe, S.: BIP 0039: Mnemonic code for generating deterministic keys. Online: `https://en.bitcoin.it/wiki/BIP_0039` [accessed: August 2022] (2013)
18. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. arXiv preprint quant-ph/0301141 (2003)
19. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Conference on the Theory and Application of Cryptology. pp. 239–252. Springer (1989)
20. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)
21. Tan, T.G., Zhou, J.: Layering quantum-resistance into classical digital signature algorithms. In: International Conference on Information Security. pp. 26–41. Springer (2021)
22. Tan, T.G., Zhou, J.: Migrating blockchains away from ECDSA for post-quantum security: Impact on users and applications. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer (2022)

Appendix

Algorithm 1: ECDSA signing with PoKG proof generation.

begin
  $G \leftarrow$ ECC base point; $P \leftarrow$ ECC order;
  $W \leftarrow$ Sequence of words; $P \leftarrow$ User password;
  $T \leftarrow$ Blockchain Transaction;
  Generate signature random r;
  Compute r^{-1} where $r * r^{-1} \equiv 1 \; mod \; P$;
  Compute $R = (R_x, R_y)$ where $R \equiv r \cdot G \; mod \; P$;
  Enumerate PoKG proof $\pi =$ **begin**
    Zero-knowledge computation of private key K_s where $K_s = PBKDF2(W||P)$;
    Zero-knowledge computation of signature s where $s \equiv r^{-1} * (Hash(T) + R_x * K_s) \; mod \; P$;
    Commit $\sigma = \{R_x, s\}$ in the proof;
  end
  destroy r, r^{-1}, K_s;
  output π as the PoKG proof;
  return σ as the transaction signature;
end

Algorithm 2: PoKG proof verification

begin
  $G \leftarrow$ ECC base point; $P \leftarrow$ ECC order; $K_p \leftarrow$ ECC public key;
  $R_x, s \leftarrow$ signature σ; $\pi \leftarrow$ PoKG proof; $T \leftarrow$ Blockchain transaction;
  Compute s^{-1} where $s * s^{-1} \equiv 1 \; mod \; P$;
  Compute $u_1 = s^{-1} * Hash(T) \; mod \; P$;
  Compute $u_2 = s^{-1} * R_x \; mod \; P$;
  Compute $V = (V_x, V_y)$ where $V = u_1 \cdot G + u_2 \cdot K_p \; mod \; P$;
  if $V_x \neq R_x$ **then**
    return "Failed Signature Verification"
  end
  else
    Verify PoKG proof $\pi =$ **begin**
      Check that R_x, s is committed in the proof;
      Check that zero-knowledge computation of s using a $PKBDF2()$ derivation of an unknown secret is correct;
      if *Check Failed* **then**
        return "Failed PoKG Proof Verification"
      end
    end
  end
  return success;
end