Faster Dual-Key Stealth Address for Blockchain-Based Internet of Things Systems

Xinxin Fan

IoTeX
xinxin@iotex.io

Abstract. Stealth address prevents public association of a blockchain transaction's output with a recipient's wallet address and hides the actual destination address of a transaction. While stealth address provides an effective privacy-enhancing technology for a cryptocurrency network, it requires blockchain nodes to actively monitor all the transactions and compute the purported destination addresses, which restricts its application for resource-constrained environments like Internet of Things (IoT). In this paper, we propose DKSAP-IoT, a faster dual-key stealth address protocol for blockchain-based IoT systems. DKSAP-IoT utilizes a technique similar to the TLS session resumption to improve the performance and reduce the transaction size at the same time between two communication peers. Our theoretical analysis as well as the extensive experiments on an embedded computing platform demonstrate that DKSAP-IoT is able to reduce the computational overhead by at least 50% when compared to the state-of-the-art scheme, thereby paving the way for its application to blockchain-based IoT systems.

Keywords: Dual-key stealth address, blockchain, Internet of Things

1 Introduction

The Internet of Things (IoT) has been connecting extraordinarily large number of smart devices to the Internet and driving the digital transformation of industry. Unfortunately, existing cloud-centric IoT systems have a number of significant disadvantages such as high system maintenance costs, slow response time, security and privacy concerns, etc. Blockchain, a form of distributed, immutable and time-stamped ledger technology, has been perceived as a promising solution to address the aforementioned problems and to securely unlock the business and operational values of IoT. The combination of blockchain and IoT facilitates the sharing of services and resources, creates audit trails and enables automation of time-consuming workflows in various applications. While combining these two technologies is creating new levels of trust, the decentralized network and public verifiability of blockchain transactions often do not provide the strong security and privacy properties required by the users.

During the past few years, quite a few cryptographic techniques such as ring signature [10], stealth address [1], and zero-knowledge proof [4] have been

employed to ensure transaction privacy for senders, receivers and transaction amount in blockchains [8,11,13,22]. This work focuses on stealth address, a privacy protection technique for receivers of cryptocurrencies. Stealth address requires the sender to create random one-time addresses for every transaction on behalf of the recipient so that different payments made to the same payee unlinkable. The most basic stealth address scheme [1] was first sketched by a Bitcoin Forum member named 'ByteCoin' in 2011, which was then improved in [13,19] by introducing the random ephemeral key pair and fixing the issue that the sender might change the mind and reverse the payment. Later on, a dual-key enhancement [18] to the previous stealth address schemes was implemented in 2014, which utilized two pairs of cryptographic keys for designated third parties (e.g., auditors, proxy servers, read-only wallets, etc.) removing the unlinkability of the stealth addresses without simultaneously allowing payments to be spent.

The dual-key stealth address protocol (DKSAP) provides strong anonymity for transaction receivers and enables them to receive unlinkable payments in practice. However, this approach does require blockchain nodes to constantly compute purported destination addresses and find the corresponding matches in the blockchain. While this process works well for full-fledged computers, it poses new challenges for resource-constrained IoT devices. Considering the limited energy budget of smart devices, we propose a lightweight variant of DKSAP, namely DKSAP-IoT, which is based on the similar idea as the TLS session resumption [2,14] and requires both the sender and receiver to keep track of the continuously updated pairwise keys for each payment session. DKSAP-IoT is able to improve the performance of DKSAP by at least 50% and reduce the transaction size simultaneously, thereby providing an efficient solution to protecting the privacy of the recipient in blockchain-based IoT systems.

The rest of the paper is organized as follows: Section 2 gives a brief overview of the elliptic curve cryptography, followed by the description of the dual-address stealth address protocol (DKSAP) in Section 3. In Section 4, we present DKSAP-IoT, a faster dual-key stealth address protocol for blockchain-based IoT systems. Section 5 analyzes the security and performance of the proposed scheme. Finally, Section 6 concludes this contribution.

2 Preliminaries

An elliptic curve E over a field $\mathbb{F}$ is defined by the Weierstrass equation:

$$E(\mathbb{F}) : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ and the curve discriminant $\Delta \neq 0$. The set of solutions $(x, y) \in \mathbb{F} \times \mathbb{F}$ satisfying the above equation along with the identity element $\mathcal{O}$, or point-at-infinity, form an abelian group under the addition operation $+$ (i.e., the chord-and-tangent group law). It is this abelian group that is used in the construction of elliptic curve cryptosystems. Given an elliptic curve point $G \in E(\mathbb{F})$ and an integer k, the scalar multiplication kG is defined by the

addition of the point G to itself $k-1$ times, i.e.,

$$kG = \underbrace{G + G + \cdots + G}_{k-1 \text{ additions}}.$$

The scalar multiplication is the fundamental operation in elliptic curve based cryptographic protocols such as the Elliptic Curve Diffie-Hellman (ECDH) key agreement [16] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [16], etc. The security of elliptic curve cryptosystems is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [6,7]. This problem involves finding the integer k $(0 < k < n)$ given a point kG, where n is the group order of $E(\mathbb{F})$. The 15 elliptic curves have been recommended by NIST in the FIPS 186-2 standard for U.S. federal government [3], which are also contained in the specification defined by the Standards for Efficient Cryptography Group (SECG) [17]. For example, the curve used in Bitcoin is called secp256k1 with parameters specified by SECG [17]. For more details about elliptic curve cryptography, the interested reader is referred to [5].

3 Dual-Key Stealth Address Protocol (DKSAP)

The first full working implementation of DKSAP was announced by a developer known as rynomster/sdcoin in 2014 for **ShadowSend** [18], a capable, efficient and decentralized anonymous wallet solution. The DKSAP has been realized in a number of cryptocurrency systems since then, including **Monero** [8], **Samourai Wallet** [15], **TokenPay** [20], just to name a few. The protocol takes advantage of two pairs of cryptographic keys, namely a 'scan key' pair and a 'spend key' pair, and computes a one-time payment address per transaction, as illustrated in Fig. 1.

When a sender A would like to send a transaction to a receiver B in a stealth mode [18], DKSAP works as follows:

1. The receiver B has a pair of private/public keys (v_B, V_B) and (s_B, S_B), where v_B and s_B are called B's 'scan private key' and 'spend private key', respectively, whereas $V_B = v_B G$ and $S_B = s_B G$ are the corresponding public keys. Note that none of V_B and S_B ever appear in the blockchain and only the sender A and the receiver B know those keys.
2. The sender A generates an ephemeral key pair (r_A, R_A) with $R_A = r_A G$ and $0 < r_A < n$, and sends R_A to the receiver B.
3. Both the sender A and the receiver B can perform the ECDH protocol to compute a shared secret:

$$c_{AB} = H(r_A v_B G) = H(r_A V_B) = H(v_B R_A),$$

   where $H(\cdot)$ is a cryptographic hash function.
4. The sender A can now generate the destination address of the receiver B to which A should send the payment:

$$T_A = c_{AB} G + S_B.$$

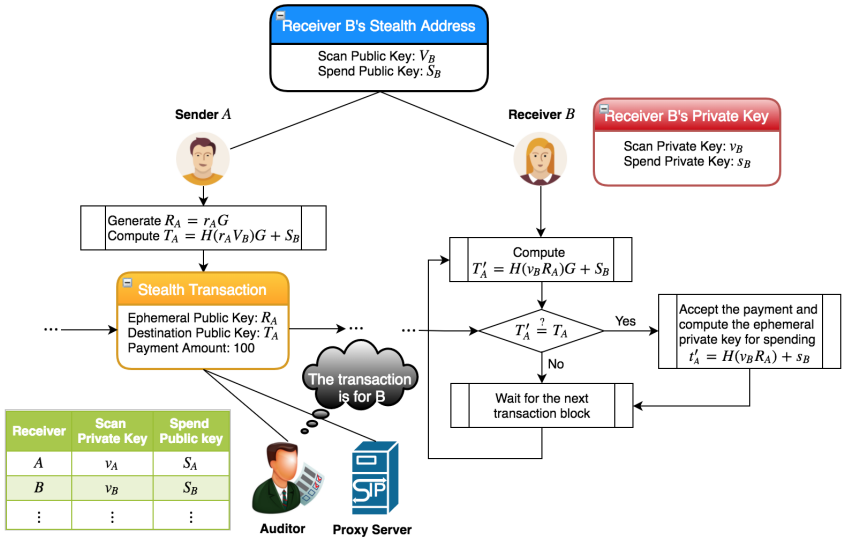

Fig. 1. The Dual-Key Stealth Address Protocol (DKSAP)

Note that the one-time destination address T_A is publicly visible and appears on the blockchain.

5. Depending on whether the wallet is encrypted, the receiver B can compute the same destination address in two different ways:

$$T'_A = c_{AB}G + S_B = (c_{AB} + s_B)G.$$

The corresponding ephemeral private key is

$$t'_A = c_{AB} + s_B,$$

which can only be computed by the receiver B, thereby enabling B to spend the payment received from A later on.

In DKSAP, the receiver B needs to actively scan the blockchain transactions, calculate the purported destination address and compare it with the one in each block until a match is found. In the case that an auditor or a proxy server exists in the system, the receiver B can share the 'scan private key' v_B and the 'spend public key' S_B with the auditor/proxy server so that those entities can scan the blockchain transactions on behalf of the receiver B. However, they are not able to compute the ephemeral private key t'_A and spend the payment from the sender A.

4 Faster Dual-Key Stealth Address Protocol for Internet of Things (DKSAP-IoT)

In this section, we describe a faster dual-key stealth address protocol called DKSAP-IoT, which is dedicatedly designed for blockchain based IoT systems.

4.1 Design Rationale

In DKSAP, the receiver B scans the blockchain and calculates the purported destination address for each transaction, which requires computations of two scalar multiplications, including one random-point scalar multiplication with the ephemeral public key R_A and one fixed-point scalar multiplication with the base point G. For resource-constrained IoT devices, computing two scalar multiplications continuously for each blockchain transaction is going to drain battery power of smart devices dramatically. Furthermore, containing an ephemeral public key in each stealth payment increases the size of the transaction and incurs additional communication overhead for IoT devices as well.

Motivated by the TLS session resumption techniques [2,14], we aim to accelerate the process for receivers finding the matched destination address by extending the lifetime of the shared secret between senders and receivers. While both the session ID [2] and the session ticket [14] are fixed in TLS for a given period of time between the client and server, the sender does need to generate a one-time destination address for each payment sent to the same recipient in our case. To this end, both the sender and receiver will apply the cryptographic hash function to their shared secret for subsequent N transactions before the sender initiates a shared secret update with a fresh ephemeral public key. This key evolving process is shown in Fig. 2, which leads to the design of DKSAP-IoT, a faster dual-key stealth address protocol for blockchain based IoT systems, as detailed in the next subsection.

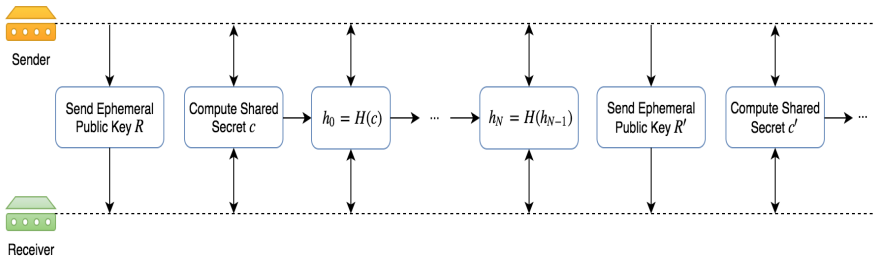

Fig. 2. The Key Evolving Process between the Sender and Receiver in DKSAP-IoT

4.2 DKSAP-IoT Specification

DKSAP-IoT is similar to DKSAP except that whenever the sender and receiver establish a shared secret using ECDH it will be continuously and pseudoran-

domly updated with a cryptographic hash function and used in their subsequent N stealth transactions. Both the sender and receiver maintain the transaction state (i.e., shared secret, counter, etc.) locally and update it after each stealth transaction. A high-level description of DKSAP-IoT is depicted in Fig. 3.

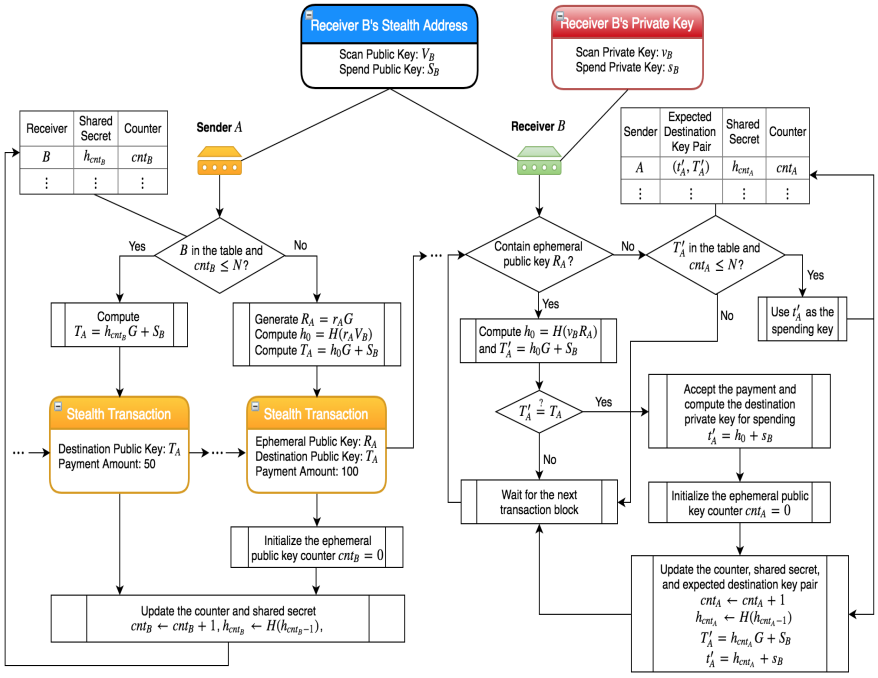

Fig. 3. The Dual-Key Stealth Address Protocol for IoT (DKSAP-IoT)

In a blockchain-based IoT system, two smart devices A and B can process a transaction in a stealth mode using DKSAP-IoT as described below:

1. The receiver device B is pre-installed with a 'scan key' pair (v_B, V_B) and a 'spend key' pair (s_B, S_B) as in DKSAP, where $V_B = v_B G$ and $S_B = s_B G$.
2. For sending a transaction to B, the sender device A first checks whether B is in A's receiver list. If B is in the list and the counter value cnt_B is less than N (i.e., A has communicated with B before), A retrieves the shared secret h_{cnt_B} from the table and computes the destination public key:

$$T_A = h_{cnt_B} G + S_B.$$

The stealth transaction that only contains the destination public key T_A as well as the payment amount is then added into the blockchain. In the case

that B is not in the list or the counter value is greater than N, A generates a fresh ephemeral public key $R_A = r_A G$ and calculates the shared secret $h_0 = H(r_A V_B)$ as well as the destination public key as in DKSAP:

$$T_A = h_0 G + S_B.$$

Here the stealth transaction is composed of the ephemeral public key R_A, the destination public key T_A and the payment amount. After putting the transaction on the blockchain, the sender A will initialize the ephemeral public key counter $cnt_B = 0$. In both cases, the counter cnt_B and the shared secret h_{cnt_B} will be updated as well:

$$cnt_B \leftarrow cnt_B + 1, \;\; h_{cnt_B} \leftarrow H(h_{cnt_B - 1}).$$

Note that only the counter cnt_B is updated when it reaches N.

3. Upon receiving a stealth transaction, the receiver B first checks whether the transaction contains an ephemeral public key R_A. If it is, B computes the purported shared secret and destination public key:

$$h_0 = H(v_B R_A), \;\; T'_A = h_0 G + S_B.$$

If the purported destination public key T'_A matches the received one (i.e., $T'_A = T_A$), B accepts the payment from A and computes the corresponding private key for spending:

$$t'_A = h_0 + s_B.$$

B also sets the ephemeral public key counter cnt_A to be 0, updates the counter and shared secret, and precomputes the expected destination key pair for the next stealth transaction from A:

$$cnt_A \leftarrow cnt_A + 1, \;\; h_{cnt_A} \leftarrow H(h_{cnt_A - 1}), \tag{1}$$

$$T'_A = h_{cnt_A} G + S_B, \;\; t'_A = h_{cnt_A} + s_B. \tag{2}$$

When B receives a stealth transaction without an ephemeral public key, B will check whether the received destination public key T_A is contained in its list of senders. If a match is found and the value of the counter cnt_A is less than or equal to N, B retrieves the corresponding destination private key t'_A as the spending key and updates the transaction state information accordingly with the equations (1) and (2). Again only the counter cnt_A is updated when it reaches N.

In DKSAP-IoT, stealth transactions are divided into two categories depending on whether ephemeral public keys are included in the blocks. For the first stealth transaction between two blockchain nodes, the receiver needs to conduct the same operations as DKSAP, followed by a more efficient preparation process for the next transaction. For the subsequent N stealth transactions between the same peers, generating a fresh ephemeral key is no longer needed on the sender side. Meanwhile, the receiver only performs a fast table look-up as well

as transaction state updates, which facilitates the receiver to quickly filter out the designated transactions.

Given the 'scan private key' v_B and the 'spend public key' S_B, the auditor/proxy server is able to calculate all the destination addresses for the receiver B, thereby tracking or forwarding all the transactions to B. However, both the auditor or the proxy server cannot derive the corresponding ephemeral private keys and spend the funds.

5 Security Analysis and Performance Evaluation

In this section, we analyze the security and performance of DKSAP-IoT and report its implementation on a Raspberry Pi 3 Model B, a good representative of moderately resource-constrained embedded devices.

5.1 Security Analysis

DKSAP-IoT follows the same threat model as DKSAP, in which the adversary aims to determine the corresponding recipients by observing the transactions on the blockchain. DKSAP-IoT provides the following security properties:

- **Receiver Anonymity**: DKSAP-IoT offers strong anonymity for receivers and ensures the unlinkability of payments received by the same payee. For each payment to a stealth address, the sender computes a new normal address T_A on which the funds ought to be received. Given two destination addresses $T_A^{(i)} = h_i G + S_B$ and $T_A^{(j)} = h_j G + S_B$ $(0 \leq i, j \leq N)$ for the same receiver B, the adversary is not able to link them thanks to the difficulty of ECDLP.
- **Forward Privacy**: DKSAP-IoT provides forward secrecy due to the usage of a cryptographic hash function for updating the shared secret continuously for N stealth transactions. If the adversary compromises the device and obtains h_l for the l^{th} $(0 < l < N)$ stealth transaction, he/she is still not able to link previous transactions because of the properties of the hash function.

Since both the sender and receiver need to locally maintain the state information for their peers in DKSAP-IoT (See Fig. 3), these tables, together with the device private keys, should be stored in the encrypted form for mitigating the risk that IoT devices might get compromised. Considering that the hardware AES engine is widely available on many IoT devices, the computational overhead for encrypting/decrypting those sensitive information is quite small.

5.2 Performance Evaluation

Computational and Communication Overhead. We assume that a sender is going to send N stealth transactions to a receiver using blockchain. Let RP, FP and H denote the computation of a random-point scalar multiplication, a fixed-point scalar multiplication and a cryptographic hash function, respectively.

Table 1. Computational Overhead of DKSAP and DKSAP-IoT for Sending N Stealth Transactions between Two Blockchain Nodes

Scheme	Sender			Receiver		
	#RP	#FP	#H	#RP	#FP	#H
DKSAP	N	$2N$	N	N	N	N
DKSAP-IoT	1	$N+1$	N	1	N	N

Table 1 gives a comparison between the DKSAP and DKSAP-IoT in terms of their computational overhead.

From Table 1, one can see that DKSAP-IoT is able to reduce the number of RP and FP by $N-1$ on the sender side, respectively, when compared to the DKSAP. Moreover, DKSAP-IoT can also save $N-1$ RP on the receiver side. With respect to the communication overhead, the sender in DKSAP-IoT only needs to contain a fresh ephemeral public key in the first stealth transaction, thereby saving the transmission of $N-1$ elliptic curve points.

Software Implementation. To validate the performance improvements of DKSAP-IoT, we implemented an optimized elliptic curve cryptography library, namely libsect283k1, using the 283-bit binary Koblitz curve specified in [17]:

$$E(\mathbb{F}_{2^{283}}) : y^2 + xy = x^3 + 1,$$

where the binary field $\mathbb{F}_{2^{283}}$ is defined by $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$. The library was written in C and compiled using the GNU C Compiler (GCC). A number of efficient techniques, such as the lambda coordinates [9], the window τNAF method [5], the pre-computation [21], etc., have been utilized to optimized the performance of the libsect283k1 library. Moreover, BLAKE-256 [12] is chosen as the hash function in our library due to its high performance cross multiple computing platforms. When running our library on a Raspberry Pi 3 Model B, the timings for the computation of RP, FP and H are shown in Table 2.

Table 2. Timings for Computing RP, FP and H on a Raspberry Pi 3 Model B

RP	FP	H
3.67 ms	3.12 ms	5.26 μs

Note that the computation of the hash function is about three orders of magnitude faster than that of the scalar multiplication over an elliptic curve. Therefore, using the hash function to update the shared secret and extend its lifetime brings significant performance benefits for IoT devices. Fig. 4 compares the performance of the DKSAP and DKSAP-IoT on both sender and receiver sides for sending $N = 10, 20$ and 30 stealth transactions, respectively.

From Fig. 4, one notices that the overall cost of DKSAP-IoT is less than 50% of DKSAP, mainly because extending the lifetime of the shared secret with a

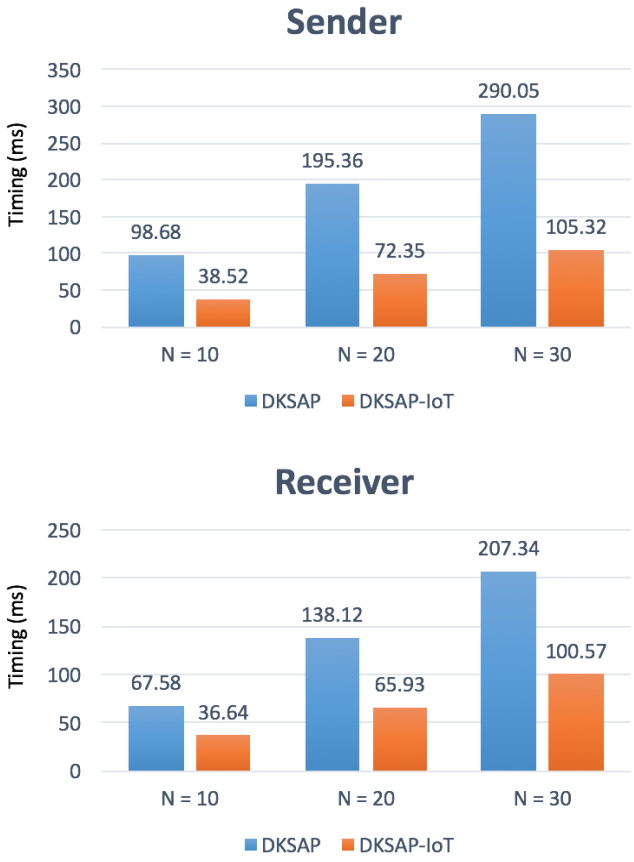

Fig. 4. Performance Comparison of the DKSAP and DKSAP-IoT for Sending $N = 10, 20$ and 30 Stealth Transactions

cryptographic hash function enables both the sender and receiver to reduce the number of RP from N to 1. Moreover, the computation of the hash function is almost negligible compared to the scalar multiplication over the elliptic curve. In addition, DKSAP-IoT can save the transmission of $72 \cdot (N-1)$ bytes for N stealth transactions. For resource-constrained IoT devices, the improved performance and reduced transaction size by DKSAP-IoT leads to significant power savings and extended battery life.

6 Conclusion

In this paper, we propose an efficient dual-key stealth address protocol DKSAP-IoT for blockchain-based IoT systems. Motived by the TLS session resumption

techniques, we apply a cryptographic hash function to continuously update a shared secret between two communication peers and extend the lifetime of this shared secret for additional N transactions. Both the sender and receiver need to maintain the state information locally in order to keep track of the pairwise session keys. The security analysis shows that DKSAP-IoT provides receiver anonymity and forward privacy. When implementing DKSAP-IoT on a Raspberry Pi 3 Model B, we demonstrate that DKSAP-IoT can achieve at least 50% performance improvement when compared to the original DKSAP, besides significant reduction of the transaction size in the block. Our work is another logic step towards providing strong privacy protection for blockchain-based IoT systems.

References

1. ByteCoin, "Untraceable transactions which can contain a secure message are inevitable", *Bitcoin Forum*, `https://bitcointalk.org/index.php?topic=5965.0`.
2. T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2". RFC 5246 (Proposed Standard), `https://tools.ietf.org/html/rfc5246`, 2008.
3. Federal Information Processing Standards Publication 186-2, "Digital Signature Standard (DSS)", National Institute of Standards and Technology, 2000.
4. S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", *SIAM Journal on Computing*, Vol. 18, No. 1, pp. 186-208, 1989.
5. D. Hankerson, A. J. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag, Secaucus, 2003.
6. N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, Vol. 48, No. 177, pp. 203-209, 1987.
7. V. S. Miller, "Use of Elliptic Curves in Cryptography", *Advance in Cryptology - CRYPTO'85*, LNCS 218, H. C. Williams (ed.), Berlin, Germany: Springer-Verlag, pp. 417-426, 1986.
8. Monero. "Stealth Address", `https://getmonero.org/resources/moneropedia/stealthaddress.html`.
9. T. Oliveira, J. López, D. F. Aranha, and F. Rodríguez-Henríquez, "Two is the Fastest Prime: Lambda Coordinates for Binary Elliptic Curves", *Journal of Cryptographic Engineering*, Vol. 4, Iss. 1, pp. 3-17, 2014.
10. R. L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret", *Advances in Cryptology - ASIACRYPT 2001*, LNCS 2248, C. Boyd (ed.), Berlin, Germany: Springer-Verlag, pp. 552-565, 2001.
11. Rynomster and Tecnovert, "Shadow: Zero-knowledge Anonymous Distributed Electronic Cash via Traceable Ring Signatures", `https://coss.io/documents/white-papers/shadowcash.pdf`.
12. M-J. Saarinen and J-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)". RFC 7693 (Informational), `https://tools.ietf.org/html/rfc7693`, 2015.
13. N. van Saberhagen, "CryptoNote v 2.0", `https://cryptonote.org/whitepaper.pdf`.
14. J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State". RFC 5077 (Proposed Standard), `https://tools.ietf.org/html/rfc5077`, 2008.

15. Samourai. https://samouraiwallet.com/index.html.
16. Standards for Efficient Cryptography Group, *SEC 1: Elliptic Curve Cryptography, Version 2.0*, http://www.secg.org/sec1-v2.pdf.
17. Standards for Efficient Cryptography Group, *SEC2: Recommended Elliptic Curve Domain Parameters version 2.0*, http://www.secg.org/sec2-v2.pdf.
18. The Shadow Project. "Dual-key Stealth Addresses", in part of Shadow Documentation, https://doc.shadowproject.io/#dual-key-stealth-addresses.
19. P. Todd, "[Bitcoin-development] Stealth Addresses", https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg03613.html.
20. TokenPay. "TokenPay – The World's Most Secure Coin (Whitepaper)", https://www.tokenpay.com/whitepaper.pdf.
21. W. Yu, S. A. Musa, G. Xu, and B. Li, "A Novel Pre-Computation Scheme of Window τNAF for Koblitz Curves", IACR Cryptology ePrint Archive, Report 2017/1020, https://eprint.iacr.org/2017/1020, 2017.
22. Zcash. https://z.cash/technology/index.html.