Simple Logarithmic-size LSAG signature

Edsger Hughes[*]

edsgerhughes@protonmail.com

July 9, 2024

Abstract

A number of existing cryptosystems use the well-known linear-size LSAG signature concept, extending it in many ways. This article presents a simple logarithmic-size signature LS-LSAG which, despite a radical reduction in size, retains the basic code block of LSAG. Therefore, substituting LS-LSAG for LSAG requires minimal changes to almost any existing LSAG/CLSAG-based solution, making it logarithmic instead of linear.

1 Introduction

The LSAG signature [6] is linear in the anonymity set size, and this linearity persists in the solutions extending LSAG. A number of existing anonymous blockchains are based upon the Cryptonote protocol [11], thus inheriting LSAG. For example, the CLSAG signature [3] of the Monero blockchain [7] optimizes the LSAG size while still remaining in the linear class. It also enhances LSAG so that 'hinged equipment', e.g, hidden wallet balance equations, can be 'mounted'.

At the same time, there already exist logarithmic signatures based on the same cryptographic assumptions as LSAG. Some of their concepts are presented in [4, 8], [1, 10], [12]. In addition to the logarithmic size, they open a possibility for efficient batch verification which is not available in LSAG. However, if you have a working software using LSAG/CLSAG, then moving to a logarithmic scheme rises a question of compatibility. That is, how many LSAG's 'hinged equipment' has to be unmounted, reconsidered and, finally, how much code needs to be rewritten.

We propose LS-LSAG, which allows for a smooth solution to this problem. LS-LSAG is a logarithmic-size signature that retains the base code block of LSAG/CLSAG, thus requiring no major replacement of the 'equipments'. However, due to the retained block, batch verification is not included. That's the trade-off. The only prerequisite for LS-LSAG is a logarithmic-size inner-product argument module, e.g., as in [2], which is typically already installed in LSAG-based systems with hidden balances.

2 Contribution

LS-LSAG protocol is presented in Figure 2. Formally, it is a log-size ring signature without trusted setup in a prime-order group of points on an elliptic curve under the discrete-logarithm assumption. An informal property of LS-LSAG is that it is compatible with LSAG/CLSAG.

A detailed sketch of a formal proof of unforgeability of LS-LSAG, which follows from the discrete-logarithm assumption and collision-resistance of the standard hash-to-curve function, is provided in Theorem 3.

This draft does not include a comparison with the existing ring signature methods to save space, however it looks like that our method has not been used before. The contemporary methods can be understood, e.g., from the comparisons in [8, 12, 1].

As an additional remark, we show how a design approach like ours can be used in a blockchain in transitional to post-quantum period. Also, we observe that an LSAG-based solution which is not too complex can be moved smoothly onto a membership proof which supports 'rerandomization'. Thus, our signature looks compatible with both LSAG/CLSAG and full-chain membership [10] being developed by Monero.

[*] c74c6036bc144cef4702e97648821d24b1abf804b53472694d9f41d33a0f3fc1, bc1q9hke6xwf8jl4su0v35776dhey2ksxanvuj8xdt

3 LSAG code block

Assuming that the reader is aware of the LSAG, CLSAG, and Cryptonote schemes, let's go directly to what they have in common. For a ring of n public keys $\boldsymbol{P}$ and a key image I, there are n systems of Schnorr-like equations

$$\begin{cases} T_i = r_i^* G + c_i^* P_i \\ V_i = r_i^* \mathcal{H}_p(P_i) + c_i^* I \end{cases} \tag{1}$$

Here, $\{c_i^*\}_{i=0}^{n-1}$ and $\{r_i^*\}_{i=0}^{n-1}$ are some scalars that are either randomly sampled or transmitted in the signature, divided respectively into the challenge and reply parts. Specific methods for obtaining these scalars vary in different LSAG-based schemes, we will sample and generate them in our own way.

In general, each equation in the systems (1) adhers to the Schnorr signature pattern. The pattern is that, for a single Schnorr signature, the left-hand side of its equation gets evaluated after the corresponding challenge and reply are inserted in the right-hand side, and a signer can successfully sign only knowing the left-hand side ahead-of-time, which is possible only knowing private key.

In the LSAG-based schemes, the scalars $\{c_i^*\}_{i=0}^{n-1} \cup \{r_i^*\}_{i=0}^{n-1}$ are chosen in such a way as to allow accepting verification only if the left-hand side of at least one of the systems (1), say at index $i = s$, is known to signer before querying the random oracle. By the same reason as in the Schnorr scheme, this knowledge of the left-hand side implies knowledge of a private key x such that $P_s = xG \wedge I = x\mathcal{H}_p(P_s)$. Our LS-LSAG signature will follow the same principle.

As an extension, composite keys can be used. For instance, CLSAG operates with keys represented as vectors. They move into our scheme unchanged, along with the corresponding random weights. For simplicity, we denote them as P_i's, implying that instead of $G, P_i, \mathcal{H}_p(P_i), I$ in (1), for LS-LSAG, there will be some expressions from the source scheme. Furthermore, there may be additional equations in (1), e.g., related to wallet balances. They come from the source scheme to LS-LSAG in the same way as the considered ones. Showing of these composite keys and additional equations is omitted to avoid cluttering.

Taking all the above into account, let us define 'LSAG code block' as the following set of equations which corresponds to n systems (1)

$$\left[\begin{array}{l} \boldsymbol{T} = \{rG + cP_i\}_{i=0}^{n-1} \\ \boldsymbol{V} = \{r\mathcal{H}_p(P_i) + cI\}_{i=0}^{n-1} \end{array}\right. \tag{2}$$

In this block, we instantly reflect the fact that in LS-LSAG there will be only two scalars r and c such that $\{c_i^*\}_{i=0}^{n-1} = \{c\}_{i=0}^{n-1}$ and $\{r_i^*\}_{i=0}^{n-1} = \{r\}_{i=0}^{n-1}$.

In all other respects the expressions in (2) are assumed to be the same as in the source LSAG-based scheme. The omitted components implicitly move into (2). If besides the set $\{r_i^*\}_{i=0}^{n-1}$ another scalar set $\{\tilde{r}_i^*\}_{i=0}^{n-1}$ is used in the additional equations of the source scheme, then it can be replaced by a corresponding scalar $\tilde{r}$. The replacement principle will be the same as for r.

4 Signature LS-LSAG

Setup procedure for LS-LSAG is shown in Figure 1. It simply creates a set of $n + 3$ linearly independent generators, one of which is the predefined group generator G. This set is used in all instances of the signature.

```
Setup($G \in \mathbb{G}$, $n \in \mathbb{N}$)
$\boldsymbol{B} \leftarrow\!\!\$\ \mathbb{G}^n,\ D, H \leftarrow\!\!\$\ \mathbb{G}$
$\texttt{cgen} = \{\boldsymbol{B}, D, H, G\} \in \mathbb{G}^{n+3}$
return cgen
```

Figure 1: LS-LSAG setup

The LS-LSAG scheme is presented in Figure 2 in the usual prove-and-verify notation. The random oracle is modeled by the hash function $\mathcal{H}_s$. When input to the random oracle is the entire transcript created up to the point of querying, then the oracle output is denoted, as usual, as a challenge.

To obtain linearly independent group elements, the standard collision-resistant hash-to-curve function $\mathcal{H}_p$ is used. It is assumed, as usual, that $\mathcal{H}_p$ deterministically models a random oracle on the curve, and thus there is no winning adversary in the DL assumption game for $\mathcal{H}_p$ image of any set.

At the final step, the zk-WIP$_{\vec{1}^n}$ inner-product argument procedure from the Bulletproofs-plus paper [2] is played, with $\lambda = 1$. Although, any other zero-knowledge inner-product argument protocol can be played instead.

Linking procedure is assumed to be a mere comparison of key images, it is not shown.

Informally speaking, in Figure 2 the prover randomly generates the left-hand side (T, V) of the system (1) under the signing index s. Then, the prover commits to (T, V) by hashing it to a point on the curve. The resulting commitment A is blinded with the random αH and sent to verifier. The verifier makes a challenge c, and the prover replies with r so that now the left-hand sides of all n systems (1) can be computed. Particularly, for the s-th system, its left-hand side turns out to be the opening of the commitment A, that is, $(T_s, V_s) = (T, V)$.

Now, both the prover and the verifier hash all the left-hand sides (T_i, V_i) into a point vector $\boldsymbol{A}$ of length n. Clearly, A is a blinded version of A_s. Considering $\boldsymbol{A}$ as a basis, the prover convinces the verifier that A is a non-trivial linear combination of the points from $\boldsymbol{A}$.

To accomplish this, the prover creates one-hot vector $\boldsymbol{a}$ where s-th element is nonzero, and using the inner-product argument shows that $\langle \boldsymbol{a}, \mathbf{1}^n \rangle = 1$ holds. Concretely, the prover shows that the element $W = A + e \langle \mathbf{1}^n, \boldsymbol{B} \rangle + eD$ is a commitment to the vectors $\boldsymbol{a}$ and $e\mathbf{1}^n$ over $\boldsymbol{A} \cup \boldsymbol{B} \cup \{D\}$ such that $\langle \boldsymbol{a}, e\mathbf{1}^n \rangle = e$. Here, D is a separate generator for inner-product, as required by the inner-product argument, and e is an auxiliary randomness which prevents the prover from adding to A any points not from $\boldsymbol{A}$.

Theorem 1 (Completeness, linkability)**.** *LS-LSAG is complete and linkable.*

Proof. Follows trivially from the scheme. □

Theorem 2 (Anonymity)**.** *LS-LSAG is anonymous.*

Proof. (Sketch) The transcripts of LS-LSAG and LSAG differ only in the commitment A and sub-transcript of zk-WIP$_{\vec{1}^n}$. The scalar challenges and the replies $r, \{r_i^*\}_{i=0}^{n-1}$ are not counted, as they are independent and uniformly distributed.

As the commitment A is hiding, and as zk-WIP$_{\vec{1}^n}$ is zero-knowledge, anonymity of LS-LSAG reduces to the anonymity of LSAG. □

Signature LS-LSAG($m \in \{0,1\}^*$, $\boldsymbol{P} \in \mathbb{G}^n$; $x \in \mathbb{F}$, $s \in [0, n-1]$)

Prover $\mathcal{P}$ **Verifier $\mathcal{V}$**

.... Step 0: $\mathcal{P}, \mathcal{V}$ validate the ring ....

$$\textbf{assert } \forall i, j, i \neq j : P_i \neq 0, P_i \neq P_j$$

..... Step 1: $\mathcal{P}$ builds the key image I and commits to (T, V) at the index s .....

assert $P_s = xG$

$I = x\mathcal{H}_p(P_s)$

$t, \alpha \leftarrow\$ \mathbb{F}$

$T = tG, \;\; V = t\mathcal{H}_p(P_s)$

$A = \mathcal{H}_p(\texttt{cgen} \| m \| \boldsymbol{P} \| I \| T \| V) + \alpha H$

.... Step 2: $\mathcal{V}$ receives I, A, makes a challenge c and gets a reply r ....

$$\xrightarrow{I, A}$$

$$\xleftarrow{c} \qquad c \leftarrow\$ \mathbb{F}$$

$$r = t - cx \qquad \xrightarrow{r}$$

.... Step 3: $\mathcal{P}, \mathcal{V}$ calculate the LSAG code block ....

$$\begin{bmatrix} \boldsymbol{T} = \{rG + cP_i\}_{i=0}^{n-1} \\ \boldsymbol{V} = \{r\mathcal{H}_p(P_i) + cI\}_{i=0}^{n-1} \end{bmatrix}$$

.... Step 4: $\mathcal{P}, \mathcal{V}$ calculate hashes of all (T_i, V_i) ....

$$\boldsymbol{A} = \{\mathcal{H}_p(\texttt{cgen} \| m \| \boldsymbol{P} \| I \| T_i \| V_i)\}_{i=0}^{n-1}$$

.... Step 5: $\mathcal{P}, \mathcal{V}$ build the commitment W using one more challenge ....

$$\xleftarrow{e} \qquad e \leftarrow\$ \mathbb{F}$$

$$W = A + e\langle \mathbf{1}^n, \boldsymbol{B} \rangle + eD$$

...... Step 6: $\mathcal{P}$ builds one-hot vector $\boldsymbol{a}$ s.t. A is a commitment to $\boldsymbol{a}$ over $\boldsymbol{A}$ ......

$$\boldsymbol{a} = \begin{cases} a_s = 1 \\ \forall i \in [0, n-1], i \neq s \; : \; a_i = 0 \end{cases}$$

.... Step 7: $\mathcal{P}$ convinces $\mathcal{V}$ that $\langle \boldsymbol{a}, e\mathbf{1}^n \rangle = e$ using Bulletproofs-plus ....

$$\textbf{play } \text{zk-WIP}_{\vec{1}^n}(\boldsymbol{A}, \boldsymbol{B}, D, H, W; \; \boldsymbol{a}, e\mathbf{1}^n, \alpha)$$

Figure 2: LS-LSAG protocol

5 Unforgeability

To understand soundness of the scheme, let's rewind it. At the end, verifier is convinced that the commitment A is a known to the prover non-trivial linear combination of points from $\boldsymbol{A}$. Rewind to the challenge c and resume with a different random value c'.

For the case, if at least one of n systems (1) has its left-hand side remaining the same for c and c', the private key can be recovered from it the way this is usually done for Schnorr-like schemes.

Consider the opposite case, where none of the systems (1) has its left-hand side for c equal to its left-hand side for c'. It's easy to see that left-hand sides at distinct indexes cannot match. Thus, there is no duplicate entries in the combined set of $2n$ left-hand sides for c and c'.

Now, consider two respective sets $\boldsymbol{A}$ and $\boldsymbol{A}'$ of $\mathcal{H}_p$ hashes of these left-hand sides, for c and c'. The hashes are all different, otherwise a collision for $\mathcal{H}_p$ is found. For each of the sets $\boldsymbol{A}$ and $\boldsymbol{A}'$, the point A is a non-trivial linear combination of its elements. Eliminating A, we get a non-trivial sum of hash-to-curve images that equals to zero. This implies existence of a winning adversary for the DL assumption game.

Thus, only the first case is possible, where one of the systems (1) has its left-hand side unchanged and the private signing key is recovered from this. It's easy to see that the key image is properly built in this case as well.

Theorem 3 (Unforgeability)**.** *LS-LSAG is unforgeable.*

Proof. (Sketch) Following the definition and theorem for unforgeability in [3], we only need to show that an adversary has only a negligible probability to successfully sign with an alien key image.

Suppose that an adversarial signer knows a private key y such that $yG \notin \boldsymbol{P}$, and produces an acceptable signature with the ring $\boldsymbol{P}$ and key image $I_A = y\mathcal{H}_p(yG)$. We will show that this signer becomes a successful adversary for the DL game.

We assume that $\mathcal{H}_p$ models a random oracle on the curve and is collision-resistant, particularly, second-preimage-resistant, e.g., follows the idea of [5]. It should be noted that, since yG is also a signing key for another successful signature, it holds that $y \neq 0$ due to the validation at Step 0 in Figure 2.

As zk-WIP$_{\vec{1}^n}$ has witness-extended emulation, the signer has an non-negligible probability of rewinding the forged signature to the challenge c and obtaining another successful forgery for another independently and uniformly sampled value c', for the same $\boldsymbol{P}, I_A, A$.

Let $\mathbf{F} = \{T_i, V_i\}_{i=0}^{n-1} = \{rG + cP_i, r\mathcal{H}_p(P_i) + cI_A\}_{i=0}^{n-1}$ be a set of the left-hand sides of n systems (1) for the transcript with c. The ring is validated for $\forall i, j, i \neq j : P_i \neq P_j$, this implies $T_i \neq T_j$, and hence $\forall i, j, i \neq j : \mathbf{F} \ni (T_i, V_i) \neq (T_j, V_j) \in \mathbf{F}$. That is, all elements in $\mathbf{F}$ are distinct. The same holds for the set $\mathbf{F}'$ related to c'.

If $\exists i, j \in [0, n-1] : \mathbf{F} \ni (T_i, V_i) = (T'_j, V'_j) \in \mathbf{F}'$, then $V_i = V'_j$ implies that, in the case of $i = j$, it holds that $(r - r')\mathcal{H}_p(P_i) + (c - c')I_A = 0$, and hence, as $I_A = y\mathcal{H}_p(yG), y \neq 0, yG \neq P_i$, the signer wins the DL game.

For the opposite case of $i \neq j$, the equality $r\mathcal{H}_p(P_i) - r'\mathcal{H}_p(P_j) + (c - c')I_A = 0$ follows from $V_i = V'_j$. As $P_i \neq P_j, P_i \neq yG, P_j \neq yG, I_A = y\mathcal{H}_p(yG), y \neq 0$, the signer wins the DL game again. As a result, since the DL assumption still holds, all elements of the set $\mathbf{F} \cup \mathbf{F}'$ are distinct with overwhelming probability.

In both transcripts, for c and c', at the Steps 5, 6, 7, the verifier is convinced that the commitment A is a non-trivial linear combination of the $\mathcal{H}_p$ images of $\mathbf{F}$ and $\mathbf{F}'$, respectively. By equating these two linear combinations to each other and thus eliminating A, the signer obtains a non-trivial linear combination of the $\mathcal{H}_p$ images of the elements from $\mathbf{F} \cup \mathbf{F}'$. Since, by the above, all the elements in $\mathbf{F} \cup \mathbf{F}'$ are different, their images are independently and uniformly sampled from $\mathbb{G}$. This means that, again, the signer wins the DL game.

Thus, if the signer successfully signs with I_A, then it wins in the DL game, which is what we meant to prove. Therefore, by the theorem about equivalence between non-slanderability and unforgeability in [3], our signature is unforgeable. □

6 Efficiency

Size of the zk-WIP$_{\vec{1}^n}$ argument is $2\log_2(n)+5$, therefore size of LS-LSAG is $2\log_2(n)+8$. For instance, for a ring of 32 addresses, LS-LSAG takes 576 bytes. For a ring of 512 addresses, its size is 832 bytes.

Let's compare verification complexities of LS-LSAG and LSAG. Since there is only one response r, LS-LSAG requires n fewer scalar-element multiplications. However, zk-WIP$_{\vec{1}^n}$ performs in a time approximately equal to $2n/\log_2(n)$, and also there is a time needed to hash n left parts.

For a ring of 32 addresses, these times should nearly compensate each other, making the verification complexities roughly equal. Although, we have not conducted such tests at the moment. As n increases, verification of LS-LSAG is likely to become comparatively faster.

7 Design summary

In a nutshell, the proposed scheme is made up of two parts. The first of them, which follows the LSAG concept, is a set of n Schnorr-like systems (1) where the left-hand side can be guessed ahead of time only by knowing the private key. The second part contains hash images $\boldsymbol{A}$ of all n left-hand sides and a commitment A to one of them, specifically the one for which the preimage is known in advance. For its main role, the second part provides a zero-knowledge proof that an opening of A belongs to $\boldsymbol{A}$.

In this design, the second part can be varied by changing the hash function and alternating the zero-knowledge proof. Beside this, the linearly independent generators $\boldsymbol{B}, D, H$ can obviously be generated on the fly, in which case the scheme does not require its own setup.

8 Additional notes

8.1 On quantum-resistance

Another feature of this design is that the mentioned above second part is connected with the first one only through the hashes. This allows the parts to be implemented under different cryptographic assumptions, stronger ones for the first and weaker for the second.

For example, imagine a setting where the DL problem on a curve is solvable by a quantum computer in a time less than lifetime of a blockchain, e.g., in a decade. Suppose that for each newly sampled point there is still a decent amount of time, say a week, to believe that its logarithm is unknown.

In this setting, the first part inevitable needs to be quantum-resistant. Let us assume that it is implemented using some of the existing (or future) post-quantum homomorhic (or partially homomorhic) commitment schemes, e.g., on a lattice. Thus, the systems (1) will be different, however the property connecting knowledge of private key with the ability to predict the left-hand side will hold for them.

The second part, which contains the hashes, can be left on the curve as is, provided that there is a guarantee that prior to submitting the proof to the blockchain the signer has not discovered any linear dependency between the generators in it. That is, between the $2n+2$ sampled and one predefined points in the set $\boldsymbol{A} \cup \boldsymbol{B} \cup \{D, H, G\}$.

Such a guarantee can be obtained by letting all of the sampled points expire, say, in a week. This can be done in the blockchain by concatenating the point preimages with the block height at the moment of sampling.

8.2 On set membership proofs with 'rerandomization'

Essentially, if there is an efficient set membership argument that supports 'rerandomization' in the sense of Curve Trees [1], then a solution using LSAG/CLSAG can be migrated onto it the following way.

Each of the Schnorr equations in the system (1), including the omitted equations of the 'hinged equipment', can be regarded as containing a point from anonymity set. For instance, in Monero blockchain there are three equations in (1) containing the address P, address hash $\mathcal{H}_p(P)$, and address balance commitment C, respectively. The anonymity set in this case is a set of n triplets $(P_i, \mathcal{H}_p(P_i), C_i)$.

Using the efficient set membership argument with 'rerandomization' a signer proves that the 'rerandomized' triplet $(P'_s, \mathcal{H}_p(P_s)', C'_s)$ belongs to the anonymity set. Then, the signer inserts $P'_s, \mathcal{H}_p(P_s)', C'_s$ into the corresponding equations in (1) and thus proves correctness of the signature. That is, for the key image I, output balance commitment B, balance blinding generator H, and 'rerandomization' generator T, the signer proves knowledge of the scalars $x, \alpha, \beta, \gamma, \delta$ in the equalities

$$\begin{cases} P'_s = xG + \alpha T \\ I = x\mathcal{H}_p(P_s)' + \beta T \\ (C'_s - B) = \gamma H + \delta T \end{cases} \tag{3}$$

Although, here is a couple of tricks. First, to prove the equalities (3), due to the presence of 'rerandomization' generator T, the Schnorr equations can no longer be used in (1). They are to be replaced with Okamoto equations [9]. Second, since it is necessary to prove that the key image I has zero 'randomization', the 'rerandomization' generator T can be sampled using I in pre-image. Alternatively, this can be proved with a call to the inner-product argument, as proposed in the Monero FCMP++ specification document [10].

Overall, the above migration way is inspired by reading the specification [10], and can be viewed as an attempt to generalize the corresponding part of its realization idea. Not claiming that this way applies for every LSAG-based solution, we see that it does for a rather non-trivial one. Hence, we say that it applies to any 'not too complex' ring signature design.

8.3 On compatibility

Thus, altogether, an LSAG-based solution is compatible with LS-LSAG, i.e., can be smoothly migrated to it. The same LSAG-based solution is compatible with the full-chain membership proof [1, 10], to the exclusion of too complex designs. Therefore, with the same caveat, LS-LSAG is compatible with the full-chain membership proof.

To sum up, this compatibility allows for incremental development of a ring signature subsystem in a cryptosystem. It can be started with LSAG/CLSAG, which is simple and can operate over any underlying curve of choice that satisfies security demands, however with a highly restricted ring size. Then, it can be migrated to LS-LSAG, which works over the same undrlying curve and increases the ring size by an order of magnitude, at the price of including the inner-product argument module if it wasn't there. Finally, it can be migrated to a full set membership proof like [1, 10], increasing the ring size by a couple of orders of magnitude at the price of introducing a new helper curve into the system.

References

[1] Matteo Campanelli, Mathias Hall-Andersen, and Simon Holmgaard Kamp. *Curve Trees: Practical and Transparent Zero-Knowledge Accumulators.* Cryptology ePrint Archive, Paper 2022/756. https://eprint.iacr.org/2022/756. 2022. URL: https://eprint.iacr.org/2022/756.

[2] Heewon Chung et al. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger.* Cryptology ePrint Archive, Paper 2020/735. https://eprint.iacr.org/2020/735. 2020. URL: https://eprint.iacr.org/2020/735.

[3] Brandon Goodell, Sarang Noether, and Arthur Blue. *Concise Linkable Ring Signatures and Forgery Against Adversarial Keys.* Cryptology ePrint Archive, Paper 2019/654. https://eprint.iacr.org/2019/654. 2019. URL: https://eprint.iacr.org/2019/654.

[4] Jens Groth and Markulf Kohlweiss. *One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin.* Cryptology ePrint Archive, Paper 2014/764. https://eprint.iacr.org/2014/764. 2014. URL: https://eprint.iacr.org/2014/764.

[5] Thomas Icart. *How to Hash into Elliptic Curves.* Cryptology ePrint Archive, Paper 2009/226. https://eprint.iacr.org/2009/226. 2009. URL: https://eprint.iacr.org/2009/226.

[6] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. *Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups.* Cryptology ePrint Archive, Paper 2004/027. https://eprint.iacr.org/2004/027. 2004. URL: https://eprint.iacr.org/2004/027.

[7] *Monero.* URL: https://www.getmonero.org.

[8] Sarang Noether and Brandon Goodell. *Triptych: logarithmic-sized linkable ring signatures with applications.* Cryptology ePrint Archive, Paper 2020/018. https://eprint.iacr.org/2020/018. 2020. URL: https://eprint.iacr.org/2020/018.

[9] Tatsuaki Okamoto. "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes". In: *Advances in Cryptology — CRYPTO' 92.* Ed. by Ernest F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 31–53. ISBN: 978-3-540-48071-6.

[10] Luke "Kayaba" Parker. *FCMP++.* Github. 2024. URL: https://github.com/kayabaNerve/fcmp-ringct/blob/develop/fcmp%2B%2B.pdf.

[11] Nicolas van Saberhagen. *CryptoNote v 2.0.* Internet. 2013. URL: https://en.wikipedia.org/wiki/CryptoNote.

[12] Anton A. Sokolov. *Efficient Linkable Ring Signature from Vector Commitment inexplicably named Multratug.* Cryptology ePrint Archive, Paper 2022/1322. https://eprint.iacr.org/2022/1322. 2022. URL: https://eprint.iacr.org/2022/1322.