An airdrop that preserves recipient privacy[⋆]

Riad S. Wahby[⋆] Dan Boneh[⋆] Christopher Jeffrey[∘] Joseph Poon[ϟ]

[⋆]Stanford University [∘]Purse.io [ϟ]Lightning Network

Abstract. A common approach to bootstrapping a new cryptocurrency is an *airdrop*, an arrangement in which existing users give away currency to entice new users to join. But current airdrops offer no recipient privacy: they leak which recipients have claimed the funds, and this information is easily linked to off-chain identities.
In this work, we address this issue by defining a *private airdrop* and describing concrete schemes for widely-used user credentials, such as those based on ECDSA and RSA. Our private airdrop for RSA builds upon a new zero-knowledge argument of knowledge of the factorization of a committed secret integer, which may be of independent interest. We also design a *private genesis airdrop* that efficiently sends private airdrops to millions of users at once. Finally, we implement and evaluate. Our fastest implementation takes 40–180 ms to generate and 3.7–10 ms to verify an RSA private airdrop signature. Signatures are 1.8–3.3 kiB depending on the security parameter.

Keywords: Cryptocurrency · Airdrop · User privacy · Zero-knowledge proof of knowledge of factorization of an RSA modulus

1 Introduction

Newly-created cryptocurrencies face a chicken-and-egg problem: users appear to prefer currencies that already have a thriving ecosystem [39]. For general-purpose cryptocurrencies, this might entail a healthy transaction volume. For currencies supporting distributed applications, it could mean having a critical mass of clients already using the provided functionality. In both cases, the bottom line is: to attract users, you must already have some.

This problem is well known in practice. One response is an *airdrop*, an arrangement in which the existing users of a cryptocurrency give value in their currency to non-users, at no cost, to entice them to become users. Airdrops have become increasingly popular [2,13,15,49], with recent high-profile examples including Stellar [74] and OmiseGO [61].

As the name implies, an airdrop is designed to transfer value to *passive* recipients. To be most effective at recruiting new users, an airdrop should not require recipients to enroll ahead of time—or, in the best case, even to know about the airdrop in advance. This is effected by leveraging existing cryptographic infrastructure. Commonly, recipients claim their airdropped value on a new blockchain by reusing their identities from some other, well-established blockchain.

[⋆] Extended abstract. The full paper is available from `https://goosig.crypto.fyi`.

While airdrops to existing blockchains are convenient, using other cryptographic infrastructure may be more effective at recruiting desirable users. A very interesting example is GitHub, since it has tens of millions of users [42], many of whom use SSH keys to access repositories and PGP keys to sign commits. GitHub publishes users' public keys [43,44], which allows cryptocurrencies to design airdrops intended for developers by allowing them to claim airdropped funds using keys from GitHub. The PGP web of trust [64], Keybase [51], GitLab [45], and the X.509 PKI [29] are interesting for similar reasons.

Yet, no matter the infrastructure they target, airdrops have a serious flaw: they offer no privacy to their recipients. This means that an observer can easily learn whether or not any given recipient has claimed her airdropped value. Even cryptocurrencies that provide anonymity mechanisms for on-chain transactions (e.g., [19,10]; §5) do not prevent this leakage, because a recipient must first use her existing identity to claim the airdropped funds. And using cryptographic infrastructure like GitHub exacerbates this privacy leak since GitHub accounts, PGP keys, etc., are often tied to software projects and professional activities. All told, these issues act as a *disincentive* for privacy-conscious recipients to redeem their awards, which reduces the airdrop's effectiveness in recruiting new users.

Existing solutions fall short of addressing this issue. The simplest possible approach—sending each recipient a fresh secret key for claiming her funds—carries an even stronger disincentive: it requires recipients to trust the sender. Both the sender and recipient know the secret key, so either can take the funds, but neither can prove who did. Meanwhile, a dishonest sender might garner free publicity with an airdrop, only to claw back the funds; or an incompetent one might accidentally disclose the secret keys. To avoid this trust requirement, a workable solution must allow *only* the recipient to withdraw the funds.

A more plausible approach is to have recipients claim airdrop funds by proving their identities in zero knowledge. Concretely, a recipient proves that she knows the secret key for some pre-existing public key (say, the RSA public key of her GitHub credential), and that no prior airdrop claim has used this public key. To preserve her privacy, she must do so without revealing which public key she is using. But proving knowledge of one secret key among a large list of RSA keys using general-purpose zero-knowledge proof systems [24,78,3,26,41,20,11,62,9] is too expensive: infeasible computational cost, enormous proofs, and/or a setup phase whose incorrect execution allows proving false statements (see §5).

Meanwhile, infrastructures like GitHub are primarily based on RSA because it is, anecdotally, the most widely-supported key type for both SSH [73] and PGP [47]. This means that taking advantage of these infrastructures effectively requires support for airdrops to RSA keys.

Our contributions. This work builds an efficient and practical private airdrop system using special-purpose zero-knowledge proofs designed for this task.

First, we define precisely the required functionality and security properties for a *private airdrop scheme* (§2.1). Second, we exhibit practical private airdrop schemes designed to work with ECDSA (§3) and RSA (§4) credentials. Our ECDSA scheme extends in a straightforward way to Schnorr [71], EdDSA [12], and similar credentials. To construct our RSA scheme, we devise a new succinct zero-knowledge proof of knowledge (ZKPK) of the factorization of a committed secret integer, which we prove secure in the generic group model for groups of unknown order [72,31]. This new ZKPK may be of independent interest.

In the full paper, we carefully describe how to use private airdrops to bootstrap a new cryptocurrency, a scheme we call a *private genesis airdrop* [77, §5]. This scheme is designed to handle millions of recipients, each of whom has hundreds of keys of mixed types (some RSA, some ECDSA, etc.) and who may potentially have lost one or more of their keys. The scheme lets the airdrop's sender prove the total value of the airdrop, while enabling airdrop recipients to prove non-payment in case the sender was dishonest.

We have also implemented and evaluated our schemes [77, §6]. Our evaluation focuses on the private airdrop scheme for RSA (which is more costly than the one for ECDSA) and the private genesis airdrop. Depending on the security parameter, our fastest implementation takes 40–180 ms for an airdrop recipient to generate an RSA-based private airdrop signature comprising 1.8–3.3 kiB. The signature takes miners 3.7–10 ms to verify. The scheme requires a trusted setup to generate one global RSA modulus with an unknown factorization. Eliminating trusted setup, by using class groups of unknown order, increases signing and verifying times by 9–13$\times$ in our reference implementation. Compared with a private airdrop to one recipient, a private genesis airdrop to one million users, each with one thousand public keys, increases signature size by less than 1.8$\times$ in the worst case. Our implementations are available under open-source licenses [46,48].

2 Background and definitions

$[\ell]$ denotes the set of integers $\{0, 1, \ldots, \ell - 1\}$. λ is a security parameter (e.g., $\lambda = 128$); we generally leave λ implicit. $\mathsf{Primes}(2\lambda)$ is the set of the smallest $2^{2\lambda}$ odd primes; this is roughly the primes up to $2\lambda + \log(2\lambda)$ bits in length.

Detailed knowledge of blockchains and cryptocurrencies is not required to understand this work. For now, we regard a blockchain simply as an append-only log of transactions. We give slightly more detail in the full paper [77, §5]; curious readers can also consult the survey of Bonneau et al. [18].

2.1 Private airdrop scheme

High-level description. In a private airdrop, a *sender* $\mathcal{S}$ creates a *token* and a *secret* for a *recipient* $\mathcal{R}$ whose public key is pk. The sender sends the secret to $\mathcal{R}$[1] and records the token in a blockchain transaction. To claim the airdrop, $\mathcal{R}$ uses the token, the secret, and her secret key sk (i.e., corresponding to pk) to sign a new transaction. Any *verifier* $\mathcal{V}$ (i.e., other blockchain stakeholders) can verify this signature using the token, and *does not* learn the recipient's pk.

[1] This is usually accomplished by encrypting the secret to the recipient's pk and publishing the resulting ciphertext, so no explicit private channel is necessary.

Syntax. Let $\mathsf{SIG} := (\text{gen}^{\mathsf{SIG}}, \text{sign}^{\mathsf{SIG}}, \text{verify}^{\mathsf{SIG}})$ be a signature scheme secure against existential forgery under a chosen message attack. The derived private airdrop scheme PAD with implicit security parameter λ is a tuple of four algorithms:

setup$(1^\lambda) \xrightarrow{\text{R}} \mathsf{pp}$**:** Output pp, which is an implicit input to the other algorithms.

send$(pk) \xrightarrow{\text{R}} (c, s)$**:** Compute and output token c and secret s for public key pk, where $(pk, sk) \xleftarrow{\text{R}} \text{gen}^{\mathsf{SIG}}()$. Here c is a public airdrop token that can later be claimed by a recipient whose public key is pk. The element s is a secret that the recipient will use, along with sk, to claim the token c.

sign$(sk, (c, s), msg) \xrightarrow{\text{R}} sig$**:** Sign message $msg \in \{0,1\}^\star$ under token-secret pair (c, s) using secret key sk, where $(pk, sk) \xleftarrow{\text{R}} \text{gen}^{\mathsf{SIG}}()$ and $(c, s) \xleftarrow{\text{R}} \text{send}(pk)$. An airdrop recipient uses this algorithm to claim the airdrop token c.

verify$(c, msg, sig) \to \{\mathsf{OK}, \perp\}$**:** OK if sig is valid for msg and token c, else $\perp$. This algorithm is used to verify a claim for the token c.

PAD may also be *validatable*, in which case it has an additional algorithm:

validate$(pk, (c, s)) \to \{\mathsf{OK}, \perp\}$**:** This algorithm outputs OK if token c with secret s granted to public key pk is valid, else it outputs $\perp$.

For schemes that are not validatable, we let $\text{validate}(\cdot, \cdot) := \mathsf{OK}$.

Functionality. We require that, for all messages $msg \in \{0,1\}^\star$,

$$\Pr\left[\begin{array}{lll} & \multicolumn{2}{c}{\text{verify}(c, msg, sig) = \mathsf{OK} \wedge \text{validate}(pk, (c, s)) = \mathsf{OK}} \\ \text{where} & \mathsf{pp} \xleftarrow{\text{R}} \text{setup}(1^\lambda) & (pk, sk) \xleftarrow{\text{R}} \text{gen}^{\mathsf{SIG}}() \\ & (c, s) \xleftarrow{\text{R}} \text{send}(pk) & sig \xleftarrow{\text{R}} \text{sign}(sk, (c, s), msg) \end{array}\right] = 1$$

Security. PAD is secure if it is *anonymous*, *unforgeable*, and *orthogonal* to SIG. *Anonymity* means, informally, that c and sig reveal nothing about pk or sk, other than a well-defined leakage given by a function Λ. This ensures that claiming a token c does not reveal the claimant's identity, as required for privacy.

Definition 1. PAD *is* ***Λ-anonymous*** *if there is a leakage function* Λ *such that for all PPT adversaries* $\mathcal{A}$ *there exists a simulator* Sim *such that the following two distributions are statistically indistinguishable, letting* $\mathsf{pp} \xleftarrow{\text{R}}$ *setup*(1^λ)*:*

$$D_{\text{r}} = \left\{\begin{array}{r} (pk, sk) \xleftarrow{\text{R}} \text{gen}^{\mathsf{SIG}}() \\ (c, s) \xleftarrow{\text{R}} \textit{send}(pk) \\ (msg, \mathsf{st}) \xleftarrow{\text{R}} \mathcal{A}(c) \\ sig \xleftarrow{\text{R}} \textit{sign}(sk,\ (c, s), msg) \\ \text{output } (pk, c, msg, sig, \mathsf{st}) \end{array}\right\} \ ; \ D_{\text{s}} = \left\{\begin{array}{r} (pk, sk) \xleftarrow{\text{R}} \text{gen}^{\mathsf{SIG}}() \\ \mathsf{H} \xleftarrow{\text{R}} \Lambda(pk, sk) \\ (c, msg, sig, \mathsf{st}) \xleftarrow{\text{R}} \mathsf{Sim}(\mathsf{H}) \\ \text{output } (pk, c, msg, sig, \mathsf{st}) \end{array}\right\}$$

Remark 1. Sim sees only H (and not pk), yet it simulates $(c, msg, sig, \mathsf{st})$. This shows that this 4-tuple reveals nothing about pk except the leakage $\mathsf{H} = \Lambda(pk, sk)$. $\mathcal{A}$ does not learn s because in an airdrop only the sender and recipient do so, and the goal is to prevent any *other* entity from learning the recipient's identity.

Remark 2. A slightly stronger definition of anonymity also includes sk in the output of both distributions. Anonymity under this definition implies, roughly speaking, that even knowledge of the key sk corresponding to a token c is not sufficient to connect sig to pk. The schemes in the following sections meet this stronger notion, but it does not appear necessary in practice.

Unforgeability means, roughly speaking, that without sk one cannot generate a valid PAD signature for any message, even given valid PAD signatures for other messages and valid signatures in the underlying SIG for arbitrary messages. Consider Forge, a game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

Setup: $\mathcal{C}$ sets $\mathsf{pp} \xleftarrow{R} \text{setup}(1^\lambda)$, $(pk, sk) \xleftarrow{R} \text{gen}^{\mathsf{SIG}}()$, and $(c, s) \xleftarrow{R} \text{send}(pk)$, then sends pk, (c, s) to $\mathcal{A}$.

Query: $\mathcal{A}$ makes any number of queries of type Q1 and Q2, in any interleaving.
Q1: $\mathcal{A}$ sends msg_i^{SIG} to $\mathcal{C}$, who replies with $sig_i^{\mathsf{SIG}} \xleftarrow{R} \text{sign}^{\mathsf{SIG}}(sk, msg_i^{\mathsf{SIG}})$.
Q2: $\mathcal{A}$ sends msg_j to $\mathcal{C}$, who replies with $sig_j \xleftarrow{R} \text{sign}(sk, (c, s), msg_j)$.

Forge: $\mathcal{A}$ outputs $(\hat{m}, \hat{s})$, winning if $\text{verify}(c, \hat{m}, \hat{s}) = \mathsf{OK} \wedge \bigwedge_j \hat{m} \neq msg_j$.

Definition 2. *Let adversary $\mathcal{A}$'s advantage in* Forge *be* $\text{Adv}_{\mathcal{A}}^{\mathsf{Forge}} = \Pr[\mathcal{A} \text{ wins}]$. PAD *is **unforgeable** if, for any PPT $\mathcal{A}$,* $\text{Adv}_{\mathcal{A}}^{\mathsf{Forge}} \leq \mathsf{negl}(\lambda)$.

Orthogonality means, informally, that PAD signatures do not help to create a SIG forgery. In other words, the airdrop scheme does not weaken the user's credential (e.g., for authenticating to GitHub). Consider Ortho, a game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

Setup: $\mathcal{C}$ sets $\mathsf{pp} \xleftarrow{R} \text{setup}(1^\lambda)$ and $(pk, sk) \xleftarrow{R} \text{gen}^{\mathsf{SIG}}()$, then sends pk to $\mathcal{A}$, who chooses (c, s) and sends them to $\mathcal{C}$. Finally, $\mathcal{C}$ aborts if $\text{validate}(pk, (c, s)) = \perp$.

Query: $\mathcal{A}$ makes any number of queries of type Q1 and Q2, in any interleaving.
Q1: $\mathcal{A}$ sends msg_i to $\mathcal{C}$, who replies with $sig_i \xleftarrow{R} \text{sign}^{\mathsf{SIG}}(sk, msg_i)$.
Q2: $\mathcal{A}$ sends msg_j^{PAD} to $\mathcal{C}$, who replies with $sig_j^{\mathsf{PAD}} \xleftarrow{R} \text{sign}(sk, (c, s), msg_j^{\mathsf{PAD}})$.

Forge: $\mathcal{A}$ outputs $(\hat{m}, \hat{s})$, winning if $\text{verify}^{\mathsf{SIG}}(pk, \hat{m}, \hat{s}) = \mathsf{OK} \wedge \bigwedge_i \hat{m} \neq msg_i$.

The game wkOrtho is similar, but further requires $\bigwedge_j \hat{m} \neq msg_j^{\mathsf{PAD}}$ for $\mathcal{A}$ to win.

Definition 3. *Let adversary $\mathcal{A}$'s advantage in* Ortho *be* $\text{Adv}_{\mathcal{A}}^{\mathsf{Ortho}} = \Pr[\mathcal{A} \text{ wins}]$. PAD *is **orthogonal** to **SIG** if, for any PPT adversary $\mathcal{A}$,* $\text{Adv}_{\mathcal{A}}^{\mathsf{Ortho}} \leq \mathsf{negl}(\lambda)$. PAD *is **weakly orthogonal** if* Ortho *is replaced with* wkOrtho *in this definition.*

Remark 3. The PAD scheme of Section 4 gives orthogonality, while the scheme of Section 3 gives only weak orthogonality. In practice, weak orthogonality suffices as long as messages signed in the PAD scheme cannot be confused with messages signed in the SIG scheme; this appears to be true in our applications.

3 Warm-up: A private airdrop to ECDSA keys

Let $\mathbb{H}$ with generator $\hat{g}$ be a cyclic group of prime order $\hat{q}$. Let the ECDSA signature scheme in $\mathbb{H}$ be the triple $(\text{gen}_{\mathbb{H}}^{\mathsf{DSA}}() \xrightarrow{R} (pk, sk),\ \text{sign}_{\mathbb{H}}^{\mathsf{DSA}}(sk, msg) \xrightarrow{R} sig$, $\text{verify}_{\mathbb{H}}^{\mathsf{DSA}}(pk, msg, sig) \rightarrow \{\mathsf{OK}, \perp\})$; $(pk, sk) = (\hat{g}^x,\ x)$ is an ECDSA key pair.

We now define PAD-DSA, a private airdrop scheme to ECDSA keys. Intuitively, the token c in this scheme is a fresh ECDSA public key derived from an existing key, such that only that key's owner can compute the corresponding secret. In particular, PAD-DSA leverages the fact that $c = pk^s = \hat{g}^{x \cdot s} \in \mathbb{H}$ is an ECDSA public key whose corresponding secret key is $sk \cdot s = x \cdot s \in \mathbb{Z}_{\hat{q}}$. Further, if s is chosen at random, pk^s is independent of pk, so c reveals nothing about pk.

Thus, PAD-DSA is the validatable private airdrop scheme given by:

setup$(1^\lambda) \to \mathsf{pp}$**:** Output $\perp$; this scheme uses no public parameters.

send$(pk) \xrightarrow{\mathrm{R}} (c, s)$**:** Choose $s \xleftarrow{\mathrm{R}} [\hat{q}] \setminus \{0\}$, set $c \leftarrow pk^s \in \mathbb{H}$, and output (c, s).

sign$(sk, (c, s), msg) \xrightarrow{\mathrm{R}} sig$**:** Output $\mathrm{sign}_{\mathbb{H}}^{\mathsf{DSA}}(sk \cdot s \in \mathbb{Z}_{\hat{q}}, (c, msg))$.

verify$(c, msg, sig) \to \{\mathsf{OK}, \perp\}$ **:** Output $\mathrm{verify}_{\mathbb{H}}^{\mathsf{DSA}}(c, (c, msg), sig)$.

validate$(pk, (c, s)) \to \{\mathsf{OK}, \perp\}$**:** OK if $s \in [\hat{q}] \setminus \{0\} \wedge c = pk^s \in \mathbb{H}$, else $\perp$.

Theorem 1. PAD-DSA *is anonymous (Def. 1), with no leakage.*

We prove Theorem 1 in the full paper [77, §3].

Definition 4 (Idealized ECDSA [22,32]). *The triple* $(\mathrm{gen}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{sign}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{verify}_{\mathbb{H}}^{\mathit{DSA}})$ *is the **idealized ECDSA** algorithm if the two hash functions called as subroutines by* $\mathrm{sign}_{\mathbb{H}}^{\mathit{DSA}}$ *and* $\mathrm{verify}_{\mathbb{H}}^{\mathit{DSA}}$ *are modeled as random oracles.*

Theorem 2. PAD-DSA *is unforgeable (Def. 2) when* $(\mathrm{gen}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{sign}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{verify}_{\mathbb{H}}^{\mathit{DSA}})$ *is modeled as the idealized ECDSA algorithm.*

Theorem 3. PAD-DSA *is weakly orthogonal to ECDSA in* $\mathbb{H}$ *(Def. 3) when* $(\mathrm{gen}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{sign}_{\mathbb{H}}^{\mathit{DSA}}, \mathrm{verify}_{\mathbb{H}}^{\mathit{DSA}})$ *is modeled as the idealized ECDSA algorithm.*

Dauterman et al. [32, Thm. 5, Appx. C] prove a statement equivalent to Theorem 2. PAD-DSA is, in effect, a signature under a related key; Theorem 3 captures the required security against related-key attacks. Morita et al. [57, Thm. 2] prove a statement equivalent to this theorem, and also suggest a tweak to DSA whose use would give full (rather than weak) orthogonality for PAD-DSA.

An alternative to the above scheme is to use $c = pk \cdot \hat{g}^s = \hat{g}^{x+s}$, with signing key $x + s \in \mathbb{Z}_{\hat{q}}$, similarly to hierarchical deterministic wallets [79]. PAD-DSA also extends naturally to Schnorr [71], EdDSA [12], and related schemes.

4 A private airdrop to RSA keys

Let $\mathbb{G}$ be a group of unknown order [77, §2.2] with generators g, h having unknown discrete-log relation. Let $\mathbb{H}$ be an auxiliary cyclic group of known prime order $\hat{q}$ with generators $\hat{g}$, $\hat{h}$ having unknown discrete-log relation. Let $n \in [N]$ be a secret integer where N is a public upper bound on n and $N > |\mathbb{G}| \cdot 2^{2\lambda}$. Let $c := g^n \cdot h^s \in \mathbb{G}$ be a Pedersen commitment [63] to n with opening $s \xleftarrow{\mathrm{R}} [N]$.

In this section we construct a private airdrop to RSA keys. We proceed in two steps: we first construct an interactive zero-knowledge proof of knowledge

(ZKPK) of the factorization of an RSA modulus $n \in \mathbb{Z}$ given a public Pedersen commitment [63] to this n (see §4.1 and §4.2). We then make this protocol non-interactive via the Fiat-Shamir heuristic [34], yielding a private airdrop (§4.3).

One way to prove knowledge of the factorization of a committed n is for the prover to commit to integers p and q, and then prove that they are nontrivial factors of n. We instantiate this approach in Section 4.1, but verifying the proof is costly: it requires an exponentiation by a several thousand–bit exponent.

To address this, in Section 4.2 we describe a second ZKPK that reduces the verifier's work by roughly $5\times$ and gives $\approx$14–50% shorter proofs. The resulting protocol leaks a small amount of information about n: at most two bits, This can be reduced to just one leaked bit under a mild assumption (Cor. 1, §4.3).

Remark 4. The protocols of this section are insecure if the group $\mathbb{G}$ contains a non-identity element of known order. In the group $\mathbb{Z}_m^\times$ the element -1 has order 2, and hence this group is unsuitable for our protocols. Instead, we work in the quotient group $\mathbb{G} := \mathbb{Z}_m^\times/\{\pm 1\}$, where elements are represented as integers in the interval $[1, m/2]$ and the product of x and y is defined as $x \cdot y = \min(z, m-z)$ where $z = (x \cdot y \bmod m)$. In this group -1 is the same as 1, and presumably there are no other known elements of known order other than the identity. We discuss the group $\mathbb{G}$ further in the full paper [77, §7].

4.1 $\mathsf{PoKF_1}$: ZKPK of factorization of a committed integer

To prove knowledge of the factorization of n, the prover establishes the relation

$$\mathfrak{R}'_{g,h} := \left\{ \begin{matrix} \Big(c \in \mathbb{G}, & (n, p, q, s) \in [N] \times \mathbb{Z}^3\Big), & \text{where} \\ c = g^n \cdot h^s, & p \cdot q = n, & p \notin \{\pm 1, \pm n\} \end{matrix} \right\} \tag{1}$$

where c is the statement and (n, p, q, s) is the witness. At a high level, the proof works as follows: the prover $\mathcal{P}$ sends the verifier $\mathcal{V}$ two Pedersen commitments c_p and c_q to p and q, respectively, then proves that $p \cdot q = n$ and $p \notin \{\pm 1, \pm n\}$. For this purpose, we combine folklore sigma protocols [71,27,60,30,6,52] with recent work extending such protocols to generic groups of unknown order [17].

To efficiently prove that $p \notin \{\pm 1, \pm n\}$ we make use of the auxiliary group $\mathbb{H}$. Recall that $\mathcal{V}$ has commitments to p and n, and could therefore prove that $p \notin \{\pm 1, \pm n\}$ by proving that $(p^2 - 1)(p^2 - n^2) \neq 0$. However, this requires a relatively large proof containing multiple elements of $\mathbb{G}$.

To sidestep this issue, we take a different approach: rather than execute the proof in $\mathbb{G}$, our P and V execute it in a much smaller group $\mathbb{H}$ of known prime order (say, an elliptic curve group). For RSA moduli at practical security levels the order of $\mathbb{H}$ is all but certainly coprime to p, $p \pm 1$, and $p \pm n$, so this suffices to convince V that $p \notin \{\pm 1, \pm n\}$ in $\mathbb{Z}$.

The prover $\mathcal{P}$ provides a commitment $\hat{c}_{p^2} \in \mathbb{H}$ to p^2, from which $\mathcal{V}$ can compute a commitment to $p^2 - 1$ as $\hat{c}_{p^2}/\hat{g} \in \mathbb{H}$. To do the same for $p^2 - n^2$ the verifier $\mathcal{V}$ needs a commitment $\hat{c}_{n^2} \in \mathbb{H}$ to n^2. Fortunately, in the airdrop context this is easy to arrange, by requiring the sender $\mathcal{S}$ to compute the token as $(c, \hat{c}_{n^2})$ with corresponding secret (s, s_2). This gives the modified relation

$$\mathfrak{R}''_{g,h,\hat{g},\hat{h}} := \left\{ \begin{array}{lll} \Big((c, \hat{c}_{n^2}) \in \mathbb{G} \times \mathbb{H}, & & (n,p,q,s,s_2) \in [N] \times \mathbb{Z}^3 \times [\hat{q}]\Big), \\ \text{where} & c = g^n \cdot h^s, & \hat{c}_{n^2} = \hat{g}^{(n^2)} \cdot \hat{h}^{s_2}, \\ & p \cdot q = n, & p \notin \{\pm 1, \pm n\} \mod \hat{q} \end{array} \right\} \tag{2}$$

for statement $(c, \hat{c}_{n^2})$ and witness (n, p, q, s, s_2).

We leave details of $\mathsf{PoKF_1}$ to the full paper [77, §4.1, Appx. B].

4.2 $\mathsf{PoKF_2}$: reducing costs by allowing (1-bit) leakage

As mentioned previously, $\mathsf{PoKF_1}$ suffers from high verification cost [77, §4.1, Appx B.4]. In this section, we give a protocol that reduces both verification and communication cost compared to $\mathsf{PoKF_1}$, but leaks one bit about n. This leakage appears to be acceptable in private airdrop applications.

To prove knowledge of factorization of n, the prover establishes the following relation for $w \in [N]$ where $w^2 \equiv t \pmod{n}$ and $t \in \mathbb{Z}$ is prime, $2 \leq t < \lambda$. (Recall that computing square roots modulo n is equivalent to factoring n.)

$$\mathfrak{R}_{g,h} := \left\{ \begin{array}{c} \Big((c,t) \in \mathbb{G} \times [\lambda], \quad (n,s,w,a) \in [N]^4\Big), \quad \text{where} \\ c = g^n \cdot h^s \in \mathbb{G}, \quad w^2 = t + a \cdot n \in \mathbb{Z}, \quad 2 \leq t < \lambda \text{ a prime} \end{array} \right\} \tag{3}$$

Here (c, t) is the statement and (n, s, w, a) is the witness. The integer relation $w^2 = t + a \cdot n$ proves that $w^2 \equiv t \pmod{n}$, as required.

Remark 5. Common hardware security tokens for RSA keys (e.g., [80]) implement a signing oracle abstraction. This means that the device's owner has access to (at best) an e^{th} root in $\mathbb{Z}_n$ for $(n, e) = pk$—and *not* to the factorization of n. Furthermore, these security tokens often fix $e = 65537$. In principle, it is possible to adapt our ZKPK to a relation analogous to (3) for $w^\star$ a 65537^{th} root of t. This proof would be an order of magnitude longer, but would eliminate the leakage about n. We leave to future work the problem of devising a concretely small ZKPK supporting these security tokens.

We now give an interactive ZKPK for Relation (3), building on the results of Boneh et al. [17]. This relation leaks that $t \in \mathbb{Z}$ is a quadratic residue modulo the committed n. As discussed below (Cor. 1, §4.3), this leakage amounts to one bit under a standard cryptographic assumption.

Protocol $\mathsf{PoKF_2}$ for relation (3) between prover $\mathcal{P}$ and verifier $\mathcal{V}$ works as follows. $\mathcal{V}$'s input is $(c, t) \in \mathbb{G} \times [\lambda]$ with t prime, and $\mathcal{P}$'s input is $(c, t, n, s, w, a) \in \mathbb{G} \times [N]^5$. To start, $\mathcal{P}$ chooses two random integers $s_1, s_2 \xleftarrow{\text{R}} [N]$ and computes $c_1 \leftarrow g^w \cdot h^{s_1} \in \mathbb{G}$ and $c_2 \leftarrow g^a \cdot h^{s_2} \in \mathbb{G}$. Next, define a homomorphism $\phi : \mathbb{Z}^8 \to \mathbb{G}^4 \times \mathbb{Z}$ parameterized by g, h, c, c_1, c_2:

$$\phi \begin{pmatrix} w, \mathit{w2}, \mathit{s1}, a, \\ na, \mathit{s1w}, sa, \mathit{s2} \end{pmatrix} := \begin{pmatrix} g^w \cdot h^{\mathit{s1}}, \quad g^a \cdot h^{\mathit{s2}}, \quad g^{\mathit{w2}} \cdot h^{\mathit{s1w}} / c_1^w, \\ g^{na} \cdot h^{sa} / c^a, \quad \mathit{w2} - na \end{pmatrix} \tag{4}$$

It is easy to see that ϕ is a group homomorphism whose range is the group $\mathbb{G}^4 \times \mathbb{Z}$. We will write the group operation in this group multiplicatively. That is, if $(a_i, b_i, c_i, d_i, e_i) \in \mathbb{G}^4 \times \mathbb{Z}$ for $i \in \{1, 2\}$, then

$$(a_1, b_1, c_1, d_1, e_1) \cdot (a_2, b_2, c_2, d_2, e_2) := (a_1 a_2,\ b_1 b_2,\ c_1 c_2,\ d_1 d_2,\ e_1 + e_2).$$

To prove knowledge of a witness for relation (3), it suffices for $\mathcal{P}$ to prove that it knows a ϕ-preimage of $T := (c_1, c_2, 1, 1, t) \in \mathbb{G}^4 \times \mathbb{Z}$. In other words, we need a ZKPK for a vector $\mathbf{v}' = (w', \mathit{w2}', \mathit{s1}', a', na', \mathit{s1w}', sa', \mathit{s2}') \in \mathbb{Z}^8$ such that

$$\phi(\mathbf{v}') = T = (c_1, c_2, 1, 1, t) \in \mathbb{G}^4 \times \mathbb{Z}. \tag{5}$$

This proves that c_1 is a commitment to $w' \in \mathbb{Z}$, c_2 is a commitment to $a' \in \mathbb{Z}$, $\mathit{w2}' = (w')^2$, and $na' = a' \cdot n$ for some integer a'. The fifth term in (5) proves that $(w')^2 - a' \cdot n = t \in \mathbb{Z}$, as required.

We design a ZKPK for a ϕ-preimage using a zero-knowledge protocol due to Boneh et al. [17, Appx. A]. Here, the verifier $\mathcal{V}$ is given $T \in \mathbb{G}^4 \times \mathbb{Z}$ and the prover $\mathcal{P}$ is given T and $\mathbf{v} \in \mathbb{Z}^8$ where $\phi(\mathbf{v}) = T$. The protocol works as follows:

(1) $\mathcal{P}$ sets $\mathbf{r} := (r_w, r_{\mathit{w2}}, r_{\mathit{s1}}, r_a, r_{na}, r_{\mathit{s1w}}, r_{sa}, r_{\mathit{s2}}) \in \mathbb{Z}^8$ where $r_w, r_{\mathit{w2}}, r_{na}, r_a \xleftarrow{\text{R}} [2^{2\lambda}]$ and $r_{\mathit{s1}}, r_{\mathit{s1w}}, r_{sa}, r_{\mathit{s2}} \xleftarrow{\text{R}} [N]$.
$\mathcal{P}$ then computes $\mathbf{R} \leftarrow \phi(\mathbf{r}) \in \mathbb{G}^4 \times \mathbb{Z}$ and sends $(c_1, c_2, \mathbf{R})$ to $\mathcal{V}$.

(2) $\mathcal{V}$ chooses challenges $ch \xleftarrow{\text{R}} [2^\lambda]$ and $\ell \xleftarrow{\text{R}} \mathsf{Primes}(2\lambda)$,[2] and sends them to $\mathcal{P}$.

(3) $\mathcal{P}$ computes $\mathbf{z} \leftarrow (ch \cdot \mathbf{v} + \mathbf{r}) \in \mathbb{Z}^8$, $\mathbf{z}_\ell \leftarrow (\mathbf{z} \bmod \ell) \in [\ell]^8$, $\mathbf{z}_q \leftarrow \lfloor \mathbf{z}/\ell \rfloor \in \mathbb{Z}^8$, and $\mathbf{Z}_q \leftarrow \phi(\mathbf{z}_q)$; and sends $(\mathbf{Z}_q, \mathbf{z}_\ell) \in (\mathbb{G}^4 \times \mathbb{Z}) \times [\ell]^8$ to $\mathcal{V}$.

(4) $\mathcal{V}$ accepts if $\mathbf{Z}_q^\ell \cdot \phi(\mathbf{z}_\ell) = T^{ch} \cdot \mathbf{R}$ in $\mathbb{G}^4 \times \mathbb{Z}$.

Verification cost is dominated by evaluation of $\mathbf{Z}_q^\ell \cdot \phi(\mathbf{z}_\ell)$, which entails four multi-exponentiations with exponents of size at most $2\lambda + \log(2\lambda)$ bits (i.e., the bit length of ℓ; §2). For $\lambda = 128$ and $N \approx 2^{4096}$, this is roughly $5\times$ less expensive than the verification cost of protocol PoKF_1 from the prior section. As we discuss in the full paper [77, Appx. B.4], PoKF_2 also gives ≈14–50% smaller proofs.

Remark 6. The commitment c_2 to the integer a is necessary for soundness, and in particular to ensure that a is an integer. If c_2 along with s_2 and the second coordinate of ϕ are eliminated then there is an attack where an adversarial prover can prove knowledge of $(\sqrt{3} \bmod n)$ using $a = 1/n$ and $w = 2$.

Theorem 4. *Protocol* PoKF_2 *is a zero-knowledge protocol for* $\mathfrak{R}_{g,h}$ *from* (3).

Definition 5. *Algorithm* $\mathcal{G}$ *is an* ***honest instance generator*** *for* $\mathfrak{R}_{g,h}$ *(eq.* (3)*) if it chooses integers* n, s, t*, and outputs* (c, t) *where* $c := g^n \cdot h^s \in \mathbb{G}$ *and* $t \in [\lambda]$.

Theorem 5. *Protocol* PoKF_2 *is an argument of knowledge for the relation* $\mathfrak{R}_{g,h}$ *in* (3) *for instances* (c, t) *generated by an honest instance generator* $\mathcal{G}$*, when the group* $\mathbb{G}$ *is a modeled as a generic group of unknown order.*

We prove Theorems 4 and 5 in the full paper [77, Appx. C].

[2] In an interactive protocol, $\ell \xleftarrow{\text{R}} \mathsf{Primes}(\lambda)$ would suffice for soundness. Applying the Fiat-Shamir heuristic causes a loss in security, thus requiring a larger ℓ [16, §3.3].

4.3 PAD-RSA: a private airdrop for RSA keys

We construct PAD-RSA by applying the Fiat-Shamir heuristic [34] to the interactive ZKPK PoKF_2 from Section 4.2. We optimize further in [77, §4.4].

Let $(\text{gen}^{\mathsf{RSA}}() \xrightarrow{\text{R}} (pk, sk), \text{sign}^{\mathsf{RSA}}(sk, msg) \xrightarrow{\text{R}} sig, \text{verify}^{\mathsf{RSA}}(pk, msg, sig) \to \{\mathsf{OK}, \perp\})$ be an RSA signature scheme, e.g., RSA-FDH [8]. Then PAD-RSA is given by:

setup$(1^\lambda) \xrightarrow{\text{R}} \mathsf{pp}$**:** Select a group $\mathbb{G}$ generated by g and h, and $N > |\mathbb{G}| \cdot 2^{2\lambda}$ an upper bound on the size of RSA moduli that can be used with these public parameters. Output $\mathsf{pp} = (\mathbb{G}, g, h, N, \lambda)$. We discuss candidate groups $\mathbb{G}$ below.

send$(pk) \xrightarrow{\text{R}} (c, s)$**:** For $(n, e) = pk$, $s \xleftarrow{\text{R}} [N]$, $c \leftarrow g^n \cdot h^s \in \mathbb{G}$, output (c, s).

sign$(sk, (c, s), msg) \xrightarrow{\text{R}} sig$**:** For $(n, p, q) = sk$, do:

(1) choose a random prime $2 \leq t < \lambda$ such that t is a quadratic residue in $\mathbb{Z}_n$,

(2) find integers (w, a) such that $w^2 = t + an$ in $\mathbb{Z}$ (i.e. $w^2 \equiv t \bmod n$),

(3) choose a random $s_1 \xleftarrow{\text{R}} [N]$ and compute $c_1 \leftarrow g^w \cdot h^{s_1} \in \mathbb{G}$,

(4) choose a random $s_2 \xleftarrow{\text{R}} [N]$ and compute $c_2 \leftarrow g^a \cdot h^{s_2} \in \mathbb{G}$,

(5) compute $\mathbf{v} \leftarrow (w, w^2, s_1, a, n \cdot a, s_1 \cdot w, s \cdot a, s_2)$,

(6) set $\mathbf{r} := (r_w, r_{w2}, r_{s1}, r_a, r_{na}, r_{s1w}, r_{sa}, r_{s2}) \in \mathbb{Z}^8$ where $r_w, r_{w2}, r_{na}, r_a \xleftarrow{\text{R}} [2^{2\lambda}]$ and $r_{s1}, r_{s1w}, r_{sa}, r_{s2} \xleftarrow{\text{R}} [N]$,

(7) compute $\mathbf{R} \leftarrow \phi(\mathbf{r}) \in \mathbb{G}^4 \times \mathbb{Z}$, where ϕ is the homomorphism defined in (4),

(8) compute $(ch, \ell) \leftarrow \text{Hash}(msg, \mathbb{G}, g, h, c, c_1, c_2, t, \mathbf{R})$, where $ch \in [2^\lambda]$ and $\ell \in \mathsf{Primes}(2\lambda)$ (e.g., by treating the hash output as a PRG seed),

(9) compute $\mathbf{z} \leftarrow (ch \cdot \mathbf{v} + \mathbf{r}) \in \mathbb{Z}^8$, $\mathbf{z}_\ell \leftarrow (\mathbf{z} \bmod \ell) \in [\ell]^8$, $\mathbf{z}_q \leftarrow \lfloor \mathbf{z}/\ell \rfloor \in \mathbb{Z}^8$, $\mathbf{Z}_q \leftarrow \phi(\mathbf{z}_q) \in \mathbb{G}^4 \times \mathbb{Z}$,

(10) output the signature $sig = (c_1, c_2, t, ch, \ell, \mathbf{Z}_q, \mathbf{z}_\ell)$.

verify$(c, msg, sig) \to \{\mathsf{OK}, \perp\}$ **:** For $(c_1, c_2, t, ch, \ell, \mathbf{Z}_q, \mathbf{z}_\ell) = sig$,

(1) output $\perp$ if $t \notin [\lambda]$ or not prime, $c1, c2 \notin \mathbb{G}$, $\mathbf{Z}_q \notin \mathbb{G}^4 \times \mathbb{Z}$, or $\mathbf{z}_\ell \notin [\ell]^8$.

(2) with $T := (c_1, c_2, 1, 1, t) \in \mathbb{G}^4 \times \mathbb{Z}$, compute $\mathbf{R}' \leftarrow \mathbf{Z}_q^\ell \cdot \phi(\mathbf{z}_\ell)/T^{ch} \in \mathbb{G}^4 \times \mathbb{Z}$,

(3) compute $(ch', \ell') \leftarrow \text{Hash}(msg, \mathbb{G}, g, h, c, c_1, c_2, t, \mathbf{R}')$, where $ch' \in [2^\lambda]$ and $\ell' \in \mathsf{Primes}(2\lambda)$,

(4) output OK if $ch' = ch$ and $\ell' = \ell$, else output $\perp$.

validate$(pk, (c, s)) \to \{\mathsf{OK}, \perp\}$**:** Output OK if $s \in [N] \wedge c = g^n \cdot h^s \in \mathbb{G}$, else $\perp$.

As discussed in Remark 4, the security of PAD-RSA relies crucially on $\mathbb{G}$ containing no elements of known order other than the identity. $\mathbb{Z}_m^\times/\{\pm 1\}$ for m an RSA modulus with unknown factorization is a convenient choice, but it requires a trusted setup (to generate m without leaking its factorization). A candidate $\mathbb{G}$ that does not require trusted setup is the class group of imaginary quadratic order [23]. We discuss further in the full paper [77, §7].

Since the ZKPK of Section 4.2 is complete, PAD-RSA is a valid scheme. The following theorems establish the security properties of PAD-RSA. Corollary 1 and

Theorem 8 rely on the quadratic residuosity assumption (QRA) [14]: informally, for RSA modulus m with unknown factorization, distinguishing between a square modulo m and a non-square with Jacobi symbol $+1$ is infeasible.

Theorem 6. PAD-RSA *is Λ^{RSA}-anonymous (Def. 1) in the ROM. Λ^{RSA} reveals two bits about $(n, e) = (pk, sk)$, namely, a small prime quadratic residue mod n.*

Corollary 1. *Under QRA, $\Lambda^{\mathsf{RSA}}(pk, sk)$ leaks one bit about pk with respect to any RSA modulus of unknown factorization, to any PPT observer.*

Theorem 7. PAD-RSA *is unforgeable in the random oracle model if computing $\sqrt{t} \in \mathbb{Z}_n$ from RSA public key $(n, e) = pk$ is hard, $2 \leq t < \lambda$ a prime.*

Theorem 8. PAD-RSA *is orthogonal to RSA under QRA in the ROM.*

We prove Theorems 6–8 and Corollary 1 in the full paper [77, §4.3].

5 Related work

Anonymity and privacy for cryptocurrencies. Our work relates broadly to privacy for cryptocurrency users, but it attacks a different problem than prior work. We very briefly rehearse that work for context. Following Bünz et al. [24], we separate prior work into anonymity, hiding associations between identities and transactions, and confidentiality, hiding contents of transactions.

While Bitcoin was intended to provide anonymity [58], in practice it does not [55,4]. Early responses to this issue hide transaction history by shuffling together unrelated transactions [53,69]. More recent work uses cryptographic machinery to give stronger guarantees [10,59,70]. CryptoNote stealth addresses [70] are similar to a PAD in that they allow a sender to derive an anonymous identity from a recipient's public key. But this scheme requires a special public key format, is incompatible with RSA keys, and has no formal security statement.

A related line of work deals with confidentiality. Maxwell showed how to construct transactions whose inputs and outputs are hidden in cryptographic commitments, and which include zero-knowledge proofs attesting to validity [54]. Later work built upon and refined this approach [66,50,65,36]. Most recently, Bünz et al. [24] showed how to significantly improve the costs of the zero-knowledge proofs on which confidential transactions are built.

Efficient airdrops. MerkleMine [56] and pooled payments [67] are methods for compressing airdrops using Merkle trees. These are similar to our private genesis airdrop (described in the full paper [77, §5]), but our design entails more complexity because it aims to preserve the privacy of recipients, supports multiple keys per recipient, and allows recipients to accuse the sender of dishonesty.

A recent survey of airdrops [35] discusses the cost of these and other methods.

General-purpose zero-knowledge proofs and private smart contracts. Several lines of work have produced frameworks for constructing zero-knowledge proofs for general NP statements; other work has applied these ideas to constructing smart contracts. For space reasons we defer this discussion to the full paper [77, §8]. In sum, these works pay a high cost for their generality, and are far more expensive than the special-purpose ZKPK of Section 4.

Group signatures, ring signatures, etc. In a group signature scheme [28,7], users join a group by registering with an administrator; thereafter, any user can sign for the group. This signature does not reveal which user signed, just that one member of the group did. Private airdrops are vaguely similar to group signatures, but they disconnect the anonymity set (*all* users who own a certain key type) from the signing set (exactly one user, designated by the sender). Our private genesis airdrop (described in the full paper [77, §5]) is roughly a "one-time-per-user" group signature with extra properties tailored to our application.

Ad-hoc anonymous identification schemes [33] and ring signatures [68], unlike group signatures, have no administrator. Instead, users create ad-hoc anonymity sets out of existing keys, then create signatures which reveal only that one user in the anonymity set was the signer. Private airdrops are similar to ring signatures in that they do not require users to register with an administrator, but an administrator (the sender) is nevertheless required.

The ring signature scheme of Abe et al. [1] admits signatures whose ad-hoc anonymity sets mix keys of different types. In this scheme, signing and verifying time and signature size are all linear in the size of the anonymity set. Our private genesis airdrop scheme also allows signatures with anonymity sets having mixed key types; it has logarithmic and concretely small cost in the size of the anonymity set, but requires a sender to set up the scheme.

Anonymous proxy signatures [37] let a delegator give signing privileges to a proxy. The delegator's role is faintly reminiscent of the sender's in a private airdrop; and like the recipient, the proxy's identity is kept secret. But the delegator retains signing privileges after designating a proxy, whereas the private airdrop sender permanently transfers signing privileges for a given token to its recipient.

Proving knowledge of factorization of an RSA modulus. A large body of work deals with proving knowledge of factorization of RSA moduli. Much of this is in the setting where the modulus n is public (e.g., [76,21,38,40]) and is thus unsuitable for our application, since revealing n would violate anonymity.

Camenisch and Michels [25] give a protocol for proving that $a \cdot b \equiv d \bmod n$ for committed values a, b, d, and n, that is secure under the discrete log assumption. This is considerably milder than our modeling $\mathbb{G}$ as a generic group of unknown order (§4.2; [77, §7]). On the other hand, as a consequence of impossibility results for Σ-protocols in groups of unknown order [5,75], the protocol requires k repetitions for soundness 2^{-k}, wherein each repetition requires five range proofs and five proofs of knowledge of a commitment's opening. This means that proofs are orders of magnitude larger and costlier to verify than in our scheme.

6 Conclusion

We have defined private airdrops, which allow users to create signatures using their cryptographic credentials *without* revealing those credentials, and we have described concrete private airdrop schemes for ECDSA and RSA keys. To construct private airdrops for RSA, we defined a new zero-knowledge argument of knowledge of the factorization of a committed integer, in generic groups of unknown order.

In the full paper [77, §5] we describe how to use these private airdrops to bootstrap a new cryptocurrency, using a design we call a private genesis airdrop. Private genesis airdrops handle millions of recipients, each having hundreds of public keys, potentially of different types. The creator of a private genesis airdrop can prove the total value he has airdropped; if he created the airdrop dishonestly, recipients can prove that they did not receive the promised funds.

Finally, we have implemented and evaluated our schemes [77, §6]. In our fastest implementation, private airdrop signatures for RSA keys take tens to hundreds of milliseconds to create and milliseconds to verify, and they comprise at most a few kilobytes. The private genesis airdrop scheme increases signature size by about a kilobyte for an airdrop to millions of users, each having hundreds of keys; its computational overhead is negligible. While these costs are expensive compared to plain RSA signatures, we believe that may be justified, in the airdrop setting, by the improvement in recipient privacy.

Our implementations are available under open-source licenses [46,48].

Acknowledgments

This work was supported in part by the NSF, the ONR, the Simons Foundation, the Stanford Center for Blockchain Research, and the Ripple Foundation. The authors thank Fraser Brown, Henry Corrigan-Gibbs, and Dmitry Kogan for helpful conversations, and David Mazières for pointing out the need for the orthogonality property.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (Dec 2002). 10.1007/3-540-36178-2˙26
2. Airdrop Alert. `https://airdropalert.com/`
3. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press (Oct / Nov 2017). 10.1145/3133956.3134104
4. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in Bitcoin. In: Proc. Financial Crypto (Apr 2013)

5. Bangerter, E., Camenisch, J., Krenn, S.: Efficiency limitations for S-protocols for group homomorphisms. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 553–571. Springer, Heidelberg (Feb 2010). 10.1007/978-3-642-11799-2˙33
6. Bangerter, E., Camenisch, J., Maurer, U.: Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 154–171. Springer, Heidelberg (Jan 2005). 10.1007/978-3-540-30580-4˙11
7. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (May 2003). 10.1007/3-540-39200-9˙38
8. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). 10.1145/168588.168596
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 701–732. Springer, Heidelberg (Aug 2019). 10.1007/978-3-030-26954-8˙23
10. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014). 10.1109/SP.2014.36
11. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: Fu, K., Jung, J. (eds.) USENIX Security 2014. pp. 781–796. USENIX Association (Aug 2014)
12. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (Sep / Oct 2011). 10.1007/978-3-642-23951-9˙9
13. Bjorøy, T.V.: The latest crypto PR craze: ‘airdropping’ free coins into your wallet. VentureBeat (Sep 2017)
14. Blum, L., Blum, M., Shub, M.: Comparison of two pseudo-random number generators. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO’82. pp. 61–78. Plenum Press, New York, USA (1982)
15. Bogart, S.: The trend that is increasing the urgency of owning Bitcoin and Etherium. Forbes (Oct 2017)
16. Boneh, D., Bünz, B., Fisch, B.: A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712 (2018), `https://eprint.iacr.org/2018/712`
17. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586. Springer, Heidelberg (Aug 2019). 10.1007/978-3-030-26948-7˙20
18. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy. pp. 104–121. IEEE Computer Society Press (May 2015). 10.1109/SP.2015.14
19. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: Anonymity for bitcoin with accountable mixes. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 486–504. Springer, Heidelberg (Mar 2014). 10.1007/978-3-662-45472-5˙31

20. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016). 10.1007/978-3-662-49896-5˙12
21. Boyar, J., Friedl, K., Lund, C.: Practical zero-knowledge proofs: Giving hints and using deficiencies. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89. LNCS, vol. 434, pp. 155–172. Springer, Heidelberg (Apr 1990). 10.1007/3-540-46885-4˙18
22. Brickell, E.F., Pointcheval, D., Vaudenay, S., Yung, M.: Design validations for discrete logarithm based signature schemes. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 276–292. Springer, Heidelberg (Jan 2000). 10.1007/978-3-540-46588-1˙19
23. Buchmann, J., Hamdy, S.: A survey on IQ cryptography. In: Proc. Public-Key Cryptography and Computational Number Theory (Sep 2000)
24. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). 10.1109/SP.2018.00020
25. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (May 1999). 10.1007/3-540-48910-X˙8
26. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1825–1842. ACM Press (Oct / Nov 2017). 10.1145/3133956.3133997
27. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (Aug 1993). 10.1007/3-540-48071-4˙7
28. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (Apr 1991). 10.1007/3-540-46416-6˙22
29. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Tech. Rep. RFC5280, IETF (May 2008)
30. Cramer, R.J.F.: Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, Universiteit van Amsterdam (Jan 1997)
31. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (Apr / May 2002). 10.1007/3-540-46035-7˙17
32. Dauterman, E., Corrigan-Gibbs, H., Mazières, D., Boneh, D., Rizzo, D.: True2F: Backdoor-resistant authentication tokens. In: IEEE Symposium on Security and Privacy (May 2019), `https://arxiv.org/abs/1810.04660`
33. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (May 2004). 10.1007/978-3-540-24676-3˙36
34. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). 10.1007/3-540-47721-7˙12

35. Fröwis, M., Böhme, R.: The operational cost of Ethereum airdrops. arXiv:1907.12383 (2019), https://arxiv.org/abs/1907.12383
36. Fuchsbauer, G., Orrù, M., Seurin, Y.: Aggregate cash systems: A cryptographic investigation of Mimblewimble. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 657–689. Springer, Heidelberg (May 2019). 10.1007/978-3-030-17653-2˙22
37. Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., Prisco, R.D., Visconti, I. (eds.) SCN 08. LNCS, vol. 5229, pp. 201–217. Springer, Heidelberg (Sep 2008). 10.1007/978-3-540-85855-3˙14
38. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (Aug 1997). 10.1007/BFb0052225
39. Gandal, N., Halaburda, H.: Competition in the cryptocurrency market. Tech. Rep. DP10157, Center for Economic Policy Research (Sep 2014)
40. Gennaro, R., Micciancio, D., Rabin, T.: An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In: Gong, L., Reiter, M.K. (eds.) ACM CCS 98. pp. 67–72. ACM Press (Nov 1998). 10.1145/288090.288108
41. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 1069–1083. USENIX Association (Aug 2016)
42. GitHub: About. https://github.com/about
43. GitHub: User public keys. https://developer.github.com/v3/users/keys/
44. GitHub: User GPG keys. https://developer.github.com/v3/users/gpg_keys/
45. GitLab: Users API. https://docs.gitlab.com/ce/api/users.html
46. GooSig: short signatures from RSA that hide the signer's public key. https://github.com/kwantam/GooSig
47. GnuPG frequently asked questions. https://www.gnupg.org/faq/gnupg-faq.html#default_rsa2048
48. handshake-org/goosig: Anonymous RSA signatures. https://github.com/handshake-org/goosig/
49. ICO Drops. https://icodrops.com/
50. Jedusor, T.E.: Mimblewimble. Tech. rep. (Jul 2016)
51. Keybase.io. https://keybase.io/
52. Maurer, U.M.: Unifying zero-knowledge proofs of knowledge. In: Preneel, B. (ed.) AFRICACRYPT 09. LNCS, vol. 5580, pp. 272–286. Springer, Heidelberg (Jun 2009)
53. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world. https://bitcointalk.org/index.php?topic=279249 (Aug 2013)
54. Maxwell, G.: Confidential transactions. Tech. rep. (2016)
55. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of Bitcoins: Characterizing payments among men with no names. In: Proc. IMC (Oct 2013)
56. MerkleMine specification. https://github.com/livepeer/merkle-mine/blob/master/SPEC.md
57. Morita, H., Schuldt, J.C.N., Matsuda, T., Hanaoka, G., Iwata, T.: On the security of the schnorr signature scheme and DSA against related-key attacks. In: Kwon, S., Yun, A. (eds.) ICISC 15. LNCS, vol. 9558, pp. 20–35. Springer, Heidelberg (Nov 2016). 10.1007/978-3-319-30840-1˙2
58. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
59. Noether, S., Mackenzie, A.: Ring confidential transactions. Ledger **1** (2016)

60. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (Aug 1993). 10.1007/3-540-48071-4˙3
61. OmiseGO airdrop update. `https://www.omise.co/omisego-airdrop-update` (Aug 2017)
62. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press (May 2013). 10.1109/SP.2013.47
63. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). 10.1007/3-540-46766-1˙9
64. Penning, H.P.: Analysis of the strong set in the PGP web of trust. `https://pgp.cs.uu.nl/plot/` (Dec 2018)
65. Poelstra, A.: Mimblewimble. Tech. rep. (Oct 2016)
66. Poelstra, A., Back, A., Friedenbach, M., Maxwell, G., Wuille, P.: Confidential assets. Tech. rep. (Apr 2017)
67. Pooled payments (scaling solution for one-to-many transactions). `https://ethresear.ch/t/pooled-payments-scaling-solution-for-one-to-many-transactions/590`
68. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (Dec 2001). 10.1007/3-540-45682-1˙32
69. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: Practical decentralized coin mixing for Bitcoin. In: Proc. ESORICS (Sep 2014)
70. van Saberhagen, N.: CryptoNote v 2.0. Tech. rep. (Oct 2013)
71. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). 10.1007/0-387-34805-0˙22
72. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997). 10.1007/3-540-69053-0˙18
73. `ssh-keygen(1)`: OpenBSD manual pages. `https://man.openbsd.org/ssh-keygen`
74. We're distributing 16 billion Lumens to Bitcoin holders. `https://www.stellar.org/blog/bitcoin-claim-lumens-2/` (Mar 2017)
75. Terelius, B., Wikström, D.: Efficiency limitations of S-protocols for group homomorphisms revisited. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 461–476. Springer, Heidelberg (Sep 2012). 10.1007/978-3-642-32928-9˙26
76. van de Graaf, J., Peralta, R.: A simple and secure way to show the validity of your public key. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 128–134. Springer, Heidelberg (Aug 1988). 10.1007/3-540-48184-2˙9
77. Wahby, R.S., Boneh, D., Jeffrey, C., Poon, J.: An airdrop that preserves recipient privacy. `https://goosig.crypto.fyi` (Jan 2020)
78. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press (May 2018). 10.1109/SP.2018.00060
79. Wuille, P.: BIP 32: Hierarchical deterministic wallets. `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki` (Feb 2012)
80. The YubiKey. `https://www.yubico.com/products/yubikey-hardware/`