Full lightwallet functionality for .NET

Primary Contact:	Andrew Arnott
Amount requested:	$18,000 USD (upon completion of grant)
Grant Description:	By funding my last minor grant proposal (Unified Address parsing and construction library for .NET (zfnd.org)), the ZF sponsored entry of Zcash onto the .NET platform, providing 7-8 million .NET developers worldwide with easy access to the very beginnings of Zcash functionality. I propose to take this Zcash library for .NET much further by expanding the scope of the library to all the basic functionality required to write a wallet app on .NET or an application for merchants to accept Zcash, whether the merchant is online or in a brick-and-mortar store. According to stackshare.io, ASP.NET is over 6X more popular than rust on web servers (https://stackshare.io/stackups/asp-net-vs-rust). By making Zcash accessible to the .NET community, and in particular by exposing functionality that would be uniquely interesting to merchants, we lower the barrier to entry for many more merchants to accept Zcash. ## Deliverables The Nerdbank.Zcash library at https://github.com/nerdcash/Nerdbank.Cryptocurrencies will be expanded upon to add the following functionality:

	1. BIP-32 and BIP-44 key and address derivation for transparent pools. 2. BIP-39 mnemonics for seeding HD wallets. 3. ZIP-32: Shielded Hierarchical Deterministic Wallets, including generation and encoding of spending, full and incoming viewing keys for Orchard and Sapling. 4. ZIP-316 Unified (full and incoming) Viewing Keys. 5. ZIP-321 payment URI creation and parsing. 6. ZIP-302 standardize memo field format. 7. ZIP-315 Best Practices for Wallet Implementations, where applicable. 8. Diversified address generation for sapling and orchard addresses. This includes support for time-based diversified addresses to avoid reusing diversified addresses in each importing wallet. 9. Download transactions from a lightwallet server. These transactions will include details of the diversified address recipient so that merchants can correlate payments back to a specific invoice or customer. 10. Create and broadcast transactions, following the ZIP-317 fee schedule. 11. Get account balance, in its various forms (spendable, insufficient confirmations, mempool). 12. A developer-friendly high-level API that brings all these things together. All this functionality will come with automated tests and API docs to ensure the quality and readiness for consumption by the .NET community. This library will be uploaded to nuget.org and MIT licensed for general availability. I never stopped working on the library since the last grant proposal, so some of the items in the above list will already be present by the time this proposal is reviewed. This work will have a strong emphasis on enabling a more user-friendly wallet than I've ever seen in the Zcash ecosystem. In particular:

	1. The multiple forms for account balances will allow a wallet app based on this library to not suddenly dip unexpectedly lower than the amount of an outbound transaction, which tends to alarm the user. 2. Shielding or other internal transactions should be described as such rather than just showing an unexplained -0.0001 balance change to the user. See Risks section for challenges in meeting this goal. ## Dependencies The ZK cryptography is both extremely complex and non-existent in the .NET ecosystem. Rather than re-implement it, I'll use FFI to expose higher level APIs from rust crates into the .NET ecosystem. All the data types and all the simpler operations that Zcash defines will be declared and implemented in native C# to ease future maintenance of the library by .NET developers. The rust crates the Nerdbank.Zcash will depend on include zcash_client_backend, zcash_primitives and orchard. Honorable mention goes to Zingo (a ZCF grant beneficiary) for their excellent ZingoLib rust crate that brings many other Zcash-related crates together in a more usable form, which is critical to this work. In cases where these crates do not expose the functionality required for the .NET library, I can fork these crates to add the required functionality and send PRs to try to contribute back to the broader community. But work will not be blocked on these PRs being accepted. ## Risks I aspire to offer higher quality transaction history than Zcash wallets expose today. For example, shielding transactions are only reported to the user as a -0.0001 withdrawal with no further explanation. In researching this area, I'm finding that the sub-par transaction history is coming from the rust crates themselves. Deep transaction insights may require dropping to lower-level dependencies and rewriting some of the transaction assembly stack. I'll have to timebox this effort and may have to fallback to

	reporting the same quality transactions in .NET that the other wallets do, for the scope of this proposal. ## The road ahead This grant is one more step along the road to making Zcash far more accessible to all people. Particularly people who aren't as technically minded as many wallets out there seem to take for granted. I anticipate the next steps will include writing the user-facing wallet apps that specialize in usability for either the end user or the merchant use cases.
Team:	I'm a team of one at this point. I have been an open-source developer for over 20 years. I am a Principal Software Engineer at Microsoft (17 years there). I have a passion for developing OSS software to raise the privacy level for end users. Some of my OSS work includes: 1. The IronPigeon protocol and library (https://github.com/aarnott/ironpigeon) that features E2E encryption and message routing over a trustless network *without disclosing anything about a person's social network*, and a POC desktop and mobile app to demonstrate it. 2. A modern personal financial tracking client application (ala Quicken) that I hope can one day consume a .NET Zcash library so that Zcash wallet operations can be done directly from within a full financial tracking application instead of a mere software wallet with the typical functionality that they contain. I own dozens of other popular repos, and have contributed to over a hundred others. Check out my GitHub profile at https://github.com/AArnott.