TALER

Taxable Anonymous Libre Electronic Reserves

Project number: Horizon Europe 101135475

D5.3

Impact of Quantum Computers on GNU Taler for WP5

Due date of deliverable: 1. December 2024
Actual submission date: 28. November 2024

WP contributing to the deliverable: WP5

Start date of project: 1. December 2023 Duration: 3 years

Coordinator:
Eindhoven University of Technology
`www.taler.net/eurotaler`

Revision 1.0

Project co-funded by the European Commission within Horizon Europe		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

HISTORY OF CHANGES		
VERSION	PUBLICATION DATE	CHANGE
0.1	25. November 2024	First circulated version
0.2	26. November 2024	Minor edits as part of consortium review
1.0	27. November 2024	Minor typo fixes. Updated analysis of timing constraints in Spend Protocol vulnerabilities. Added new Spend Protocol vulnerability. Clarification of linkability de-anonymization attack in Refresh Protocol

Impact of Quantum Computers on GNU Taler for WP5

Tanja Lange, Jonathan Levin

28. November 2024
Revision 1.0

The work described in this report has been funded (in part) by the European Union in the HORIZON-CL4-2023-HUMAN-01-CNECT call in project 101135475 TALER. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Abstract

GNU Taler is an auditable, privacy-preserving electronic cash system that allows customers to make anonymous payments, while also protecting against money-laundering and allowing for easy reporting and taxation. GNU Taler consists of several cryptographic protocols, each of whose security relies on the security of various underlying cryptosystems, including public key cryptosystems vulnerable to attacks by quantum computers. At the time of this report, quantum computers powerful enough to run Shor's algorithm on real-world cryptosystems are not known to exist, but there is a consensus within the field of cryptography that pre-quantum protocols should be updated to use post-quantum cryptography as soon as possible to minimize the potential damage that future adversaries with a powerful quantum computer can cause. This document catalogs the existing cryptography used in GNU Taler and evaluates the threats to security and usability that a quantum attacker poses. It is the first of two reports on the impact of quantum computers on the GNU Taler protocol.

Keywords: GNU Taler, quantum impact, post-quantum cryptography, security assumptions.

Chapter 1

Introduction

GNU Taler (Taler) is a new privacy-preserving electronic payment system, which provides strong privacy guarantees for users, while also allowing for easy auditing and enforcement of existing financial regulations. Taler makes extensive use of public key cryptography to achieve a its security guarantees. With the widely expected advent of quantum computers in the coming years, we wish to evaluate the potential impact that these will have on the security goals of GNU Taler. This chapter introduces the high level issues presented by quantum computers and also gives a brief introduction to GNU Taler.

1.1 Threats to Cryptography from Quantum Computers

Quantum computers, machines which can exploit quantum physical phenomena in algorithmic computations, are widely expected to be built in the coming decades. A major consequence of a future quantum computer is that it can execute Shor's Algorithm [13], a polynomial-time quantum algorithm that can factor large integers and compute discrete logarithms, two hard problems that underpin the vast majority of deployed public key cryptography and digital signatures on the Internet. A less catastrophic but still important quantum impact on cryptography is Grover's Algorithm [9], which can compute a secret $2n$-bit AES key in $O(2^n)$ steps instead of $O(2^{2n})$ steps, a quadratic speedup over pre-quantum search algorithms, although the steps in Grover's algorithm are significantly more expensive than in the pre-quantum one and must be executed sequentially. While the impacts of Grover's algorithm on symmetric cryptography can be mitigated completely by doubling key lengths, Shor's algorithm cannot be outpaced by choosing larger parameters and requires that affected public key schemes be completely replaced by ones that resist attacks by quantum computers. The speedup from Grover's algorithm improves all attacks which include a preimage search, which is typically the best type of attack in symmetric cryptography, but it can also be used to speed up subroutines in attacks on public-key cryptography. The latter is relevant for attacks on post-quantum systems.

While many post-quantum public-key cryptographic schemes have been proposed, widespread deployment is still a work in progress, and most work has focused on the simple building blocks of public-key encryption, Key-Encapsulation Mechanisms (KEMs), and signatures. Many advanced cryptographic systems do not have matching post-quantum alternative or at least not one matching the performance of the pre-quantum one.

For example, David Chaum outlined a one-RTT protocol for blind signatures [4] in 1982,

which can be implemented using RSA [12], but the fastest current proposal [3] for a lattice-based blind signature scheme requires two round trips. This means that protocols using Chaum-style blind signatures will likely need alterations beyond a simple replacement of RSA with a new post-quantum primitive. They' will also be affected by the size and performance impacts coming with lattice-based systems.

1.2 Overview of the GNU Taler Electronic Payment System

Chaum's blind signatures were developed as a component for his untraceable payment system [4]. In Chaum's scheme, a bank would sign a blinded coin from a customer, rendering the coin spendable. The customer would then unblind the signature and send the coin to the payee as payment in a transaction. The payee would forward this coin (signature) to the bank for verification. Because the bank signed a blinded value, it cannot match the submitted signature to the one it issued originally to the customer. However, it can still verify the unblinded signature, and credit payment to the payee who submitted it.

GNU Taler is a digital payment system that significantly builds upon Chaum's original system [7]. Like Chaum's digital cash, coins in GNU Taler are untraceable to the customers who spend them, but they allow for income transparency on the side of the payee. This prevents money laundering/tax evasion by merchants who may like to accept payments without reporting the income to tax authorities. GNU Taler supports unlinkable change, refunds, and more recently, anonymous donations [10] and age restrictions on coins [11],

GNU Taler currently makes heavy use of public key cryptography; Almost all the protocols described in the thesis by Florian Dold [7], which introduced the Taler protocol, use (regular and/or blind) digital signatures, and one protocol — the Refresh Protocol — uses Diffie-Hellman key exchange [6] on elliptic curves. Because of the threat posed by quantum computers on public key cryptosystems, this report aims to evaluate the potential impact an attacker with a quantum computer would be able to have on GNU Taler. In Chapter 2, we describe the various functions of public key cryptography in each of the main protocols of GNU Taler. In Chapter 3, we provide an analysis of the impact of a quantum attacker on these protocols.

We note that while a quantum computer can break the cryptographic schemes used in the existing GNU Taler system, the impact of a quantum attacker is not consistently severe. Some protocols would require an active quantum attacker who can modify the messages being sent between honest parties in real time, while others are susceptible to attackers who store protocol transcripts or steal an exchange's database and attack them much later. We highlight the potential severity of a particular quantum attacker for each protocol, and also mention the timing requirements for an attack to be beneficial. The latter determines the level of urgency there is for rolling out the replacements.

Chapter 2

Cryptography Used in GNU Taler

This chapter gives an overview of the main protocols of GNU Taler, focusing on the cryptographic components of each protocol, and highlighting, when relevant, the relationship between those components and the security or correctness of the protocol. Details such as algorithms used internally by a single protocol party, or database operations, are omitted. Interested readers may wish to refer to [7] for more detailed specifications.

2.1 Exchanges, Master Signing Key, and Denomination Keys

The value of each coin in GNU Taler is derived from a cryptographic signature applied to it by an exchange, which blindly signs each coin with an RSA signing key associated with a specified denomination value.[1] Because a denomination key's signing key is used to create new coins, its compromise must be avoided to prevent third parties from generating their own valid coins. To limit potential damage from a compromised denomination key, these keys have a pre-determined validity period and are replaced regularly by new denomination keys. The exchange publishes announcements of its denomination keys along with the validity period for each key, and signs these announcements with a separate, long-term Ed25519 [2] master signing key. These signatures bind the announcement to the exchange, and allow other users to verify that the coins signed by denomination keys indeed come from the exchange.

Because a compromised denomination signing key allows an unauthorized party to sign its own coins, effectively printing its own money, there is an incentive to limit the validity period of each denomination key to bound potential losses. However, a validity period that is too short can impact the anonymity of customers, as too few customers may possess coins signed by the same denomination key. There is thus a trade-off between longer validity periods and better customer anonymity and shorter validity periods and increased protection from key compromises.

2.2 Withdrawal Protocol

The Withdrawal Protocol enables a customer to transfer currency from their bank account into their GNU Taler wallet. It consists of three phases: (1) Create Reserve, where funds are

[1] GNU Taler also supports the use of the Clause Blind Schnorr Signature scheme [8, 5]; however, RSA is the default so we focus on this variant in this report.

transferred from the customer's bank to a GNU Taler reserve at an exchange, (2) Prepare Withdraw, where the customer generates coins (Ed25519 keypairs), along with a random blinding factor for each coin, and (3) Execute Withdraw, where the customer sends the blinded public key of each coin to the exchange to be signed by the exchange's appropriate denomination key. The exchange performs non-cryptographic operations to check that the customer is not overdrawing coins, for example, but we ignore these details here.

The Withdrawal Protocol uses both Ed25519 signatures and blind Full Domain Hash (FDH)-RSA signatures in several ways. During the Create Reserve phase, the customer generates an Ed25519 keypair that identifies a new reserve at the exchange. The customer sends the public key for this reserve to the exchange with their bank transfer. When the customer executes a withdrawal for that reserve, they sign their request with the corresponding private key, proving that the coin request is coming from the owner of the reserve.

During the Execute Withdraw phase, the customer hashes and blinds the public key portion of each freshly generated coin. The exchange signs this blinded value using the appropriate denomination private key, conferring to the coin its value and authenticity.

While each signed GNU Taler coin that the customer obtains from the exchange is itself the public key of an Ed25519 keypair, this key is not used to perform any cryptographic operations in the Withdrawal Protocol itself.

2.3 Payment: Spend Protocol and Deposit Protocol

Payment consists of two protocols performed in succession: (1) the Spend Protocol, where the customer and merchant agree on a payment contract, and the customer "spends" coins by signing them to indicate permission for the merchant to deposit them at the exchange, and (2) the Deposit Protocol, where the merchant forwards the spent coins to the exchange and receives proof of deposit for those coins.

Both of these protocols exclusively make use of Ed25519 signatures for cryptographic operations. In the Spend Protocol, the customer makes use of two different Ed25519 signing keys. One of them is an Ed25519 keypair which is newly created by the customer, who sends the public key to merchant; this public key is then included in the transaction offer from the merchant. The merchant uses their long-term Ed25519 key to sign the transaction offer. The customer keeps the private key of their fresh key pair to be able to later prove to a third party that they received the offer from the merchant. The customer also has the Ed25519 private key(s) belonging to the coin(s) spent during the transaction. Each coin's private key is used to sign a tuple that binds the coin to the transaction offer and the merchant. This signature serves as a proof that the customer gives permission to the merchant to deposit that coin into the merchant's bank account.

In the deposit protocol, the signed coins (permissions of deposit) are sent to the exchange, where the exchange checks for coin validity and double spending. If the coins have a valid denomination signature from the exchange, and the signature on the deposit permission verifies against the coin's public key, then the exchange also signs the deposit permission as a certificate of deposit and returns this to the merchant.

2.4 Refresh and Linking Protocol

The Refresh Protocol allows customers to exchange a *dirty* coin, that is coins which were typically either (1) partially spent, (2) nearly expired, (3) potentially disclosed in an aborted transaction, or (4) previously refreshed into a subsequently revoked denomination, for new or *fresh* coins. The fresh coins are then unlinkable to the dirty coin by anyone other than the customer. In order to protect against the potential loophole of letting someone else obtain the refreshed coins, effectively an untraceable peer-to-peer payment, the Linking Protocol makes it possible for the owner of the dirty coins to re-obtain the fresh coins, meaning that the fresh coins can be shared, but cannot completely change hands to a new owner. In terms of the cryptography used, the Refresh Protocol is perhaps the most complicated, as it takes advantage of the coincidence that Ed25519 keys can also be used to perform Curve25519 [1] Diffie-Hellman key exchange.

The Refresh Protocol is an interactive protocol between the customer and the exchange. Given a dirty coin (an Ed25519/Curve25519 keypair), the customer first generates $k = 3$ random seeds, each of which is fed through HKDF to create a new Curve25519 keypair. The customer then computes the shared Diffie-Hellman secret between each private key of these new keys and the public key of the dirty coin. Each shared secret is fed again through HKDF to create a new 256-bit value, which becomes a private key of a new Ed25519 keypair. The shared secret is also mapped to an element of $\mathbb{Z}_N^*$, where N is the modulus of the denomination key used to sign the fresh coin. These new Ed25519 keypairs are planchets for new coins from the exchange, and each is paired with a blinding factor as in the Withdrawal Protocol, except that in this case the blinding factor is not random, but is instead deterministically derived from the Diffie-Hellman secret.

The customer creates a hashed commitment of the k Curve25519 public keys and k planchets, signs this commitment with the private key of the dirty coin, and sends it to the exchange. The exchange verifies the signature and then picks a random challenge index γ, and sends this to the customer. The customer then reveals the seeds for all the new Curve25519 keys it generated, except for the one indexed by γ. They also send the public key of the Curve25519 key indexed by γ and the corresponding planchet. Because the planchets and blinding factors are generated deterministically from the random seeds, the exchange can recompute all the Curve25519 keys and planchets, except for γ, and can also recompute the commitment sent by the customer. If the recomputed commitment matches what was sent by the customer, the exchange assumes that the new planchet$_\gamma$ was also generated honestly and signs it, yielding a fresh coin. The exchange stores the challenge Curve25519 and blinded planchet in its database.

In the Linking Protocol, a customer can request the transfer Curve25519 public key and signed planchet values from the exchange by providing the original dirty coin public key. Because the customer knows the private key of the original coin, it can perform the Curve25519 operations using the private key of the old coin and the public transfer key used to derive the keys for the fresh coin, obtaining the blinding factor for planchet and using that to compute the signature over the fresh coin and its private key.

2.5 Refunds

The Refund Protocol occurs between the merchant, exchange, and the customer, and only makes use of Ed25519 signatures. The customer requests a refund from the merchant. If granted, the merchant signs a refund message that includes the transaction details with its long-term Ed25519 key and send this to the exchange. The exchange verifies the signature, updates its database to make the customer's coins spendable (and thus also refreshable) again, and signs a confirmation message for the refund with its long term signing key. The exchange sends this message to the merchant, who can then forward it to the client.

Chapter 3

Impact of Quantum Computers on GNU Taler

Shor's Algorithm makes it possible for someone with a quantum computer to factor large integers, like those used as RSA moduli, and compute discrete logarithms, like those used in Ed25519 and Curve25519, in polynomial time. The protocols in GNU Taler exclusively use factoring- and discrete logarithm-based public key cryptography. This means that the cryptographic protocols in GNU Taler are very susceptible to attacks with a quantum computer.

There are mitigating factors that depend on implementation details, however. For example, computing the discrete logarithm of an Ed25519 public key requires that the actor with a quantum computer has access to this public key. If the public key is not visible to the attacker, because the attacker does not have access to traffic on the network at the moment required to use the information effectively, then this may in practice provide some protection. However, for the purpose of this report, we assume that any quantum attacker is also in control of the network, and can actively view and alter messages between participants in real time.

We also ignore economic considerations for the most part. Any attack that is more expensive to execute than the monetary value it produces is unlikely to occur in the real world. Still, attack costs change over time, and so in order not to underestimate the likelihood of a possible attack, we assume here that the cost to an attacker of breaking pre-quantum schemes with a quantum computer is essentially zero. That is, an economic analysis of attacks is outside the scope of this report.

3.1 Exchange Master Signing Key

Exchanges use a long-term Ed25519 master signing key to sign their published denomination keys. A quantum attacker can compute the discrete logarithm of the master signing key's public key, allowing them to forge signatures from that key on its own sets of denomination keys. The attacker would then distribute these phony denomination keys to customers and sign coins with these phony keys during the Withdrawal Protocol. The customer will accept these signed coins because they are signed with denomination keys which in turn have signatures that verify against the exchange's master signing key. Each set of phony denomination keys could even be targeted to a specific customer, so every customer would potentially be uniquely identified by the phony public key used during the withdrawal protocol. When

the customer then spends these coins and the merchant deposits them, the deposit will fail, because the denomination key which signed the deposited coins is not known to the actual exchange. However, by observing deposits at the exchange, the attacker can see which customer participated in the transaction by matching the denomination keys used for the deposit protocol with the customer(s) who withdrew coins against that key.

This attack requires an active attacker with a quantum computer and is detectable by the customer when they try to spend their coins and the honest exchange rejects them. The attacker also needs to actively impersonate the exchange and convince customers to use a set of phony denomination keys. It is also worth noting that there are less detectable ways to deanonymize customers with a quantum computer, such as by attacking the Refresh Protocol. Because this vulnerability requires an active, live attack, we consider this a low severity vulnerability while quantum computers are not presumed to exist.

3.2 Exchange Denomination Keys

As RSA signing keys, all denomination keys are currently vulnerable to Shor's Algorithm. A quantum adversary can recover an exchange's denomination key signing keys and sign new coins for themself. Under normal circumstances, exchanges can automatically detect a denomination key compromise at the point that the number of deposits for a given denomination exceeds the total number of coins issued with that denomination key. If compromises are rare events, this naturally limits the financial losses of the exchange to a hopefully acceptable level. A quantum attacker can compromise every denomination key, however, meaning that exchanges may be unable to absorb the losses from these forged coins.

This vulnerability does not require the quantum attacker to interfere in any protocols, but does require the attacker to have a quantum computer and finish carrying out the attack during the validity period of the denomination keys it is attacking, as coins forged by an attacker with a revoked or expired key would not be spendable. Due to the requirement for a quantum computer at the time of the attack, this vulnerability is not currently considered particularly severe. As soon as a quantum computer exists that can execute this attack, however, this will have a severe impact on the GNU Taler system.

3.3 Withdrawal Protocol

In the Create Reserve phase of the Withdrawal Protocol, a customer generates an Ed25519 keypair which identifies a reserve with the exchange. The public key of this reserve keypair is sent along with the bank transfer from the customer's bank to the exchange. Because the attacker is able to compute the the discrete logarithm of this public key, they are able to forge a signature on their own blinded planchets that will verify under the customer's reserve public key. The attacker then simply performs the Prepare Withdraw and Execute Withdraw steps of the protocol with the exchange before the customer is able to do so, forging signatures on the blinded coins. The exchange will check that these signatures verify, and then return the signed coins to the attacker, and deducting from the customer's reserve balance accordingly.

This attack requires a live quantum attacker who can observe core banking transactions to learn the public key, can first break the discrete logarithm of the Ed25519 verification key and then execute the Withdrawal Protocol and still be faster than the legitimate customer (who is likely actively waiting for the exchange to notify them that their transaction has

arrived). The net effect is to steal coins belonging to a legitimate customer, who would likely detect this when they try to execute the Withdrawal Protocol with the exchange and learn their reserve balance, which they just made a transfer to create, is zero. Due to the limited damage to a customer/the exchange from this particular attack, and the narrow time window during which the attacker must execute it, we consider the impact of this attack to be low to medium severity.

3.4 Payment: Spend Protocol and Deposit Protocol

Each of the signatures in the payment protocols can be forged by an attacker with a quantum computer who can observe the communication between a customer and merchant on the network. If the merchant's signing key is targeted, the attacker would being able to generate fake transaction offers originating from that merchant. If the customer's ownership identifier key is targeted, the attacker would be able to generate a proof that they are the true recipient of the transaction offer. If the coin Ed25519 public keys are targeted, then the attacker could forge signatures using those coins' private keys. Also, because the client sends the merchant the exchange's signatures on those coins, the attacker can create fake deposit permissions for those coins.

There are several possible attacks originating from these compromises, each with different constraints. The attacker could try to spend the customer's coins themself, but this requires that they run Shor's algorithm on coins' Ed25519 public keys, use the corresponding private keys in another complete instance of the Spend Protocol, and complete that before the legitimate merchant has deposited the coins. This race is probably unwinnable by the attacker due to the short time window and many extra computations required. The attacker could instead try to deposit the coins directly with the exchange, posing as a merchant. This is also a (probably unwinnable) race against the legitimate merchant to deposit the coins, but requires slightly fewer steps for the attacker.

However, the proof of deposit signatures from the exchange in the Deposit Protocol are also forgeable, as the exchange uses a publicly announced Ed25519 keypair to sign them. An attacker who sees a customer's coin Ed25519 public keys (and accompanying denomination key signatures) can therefore forge an entire Spend/Deposit Protocol transcript for these coins, including forged deposit permissions from the customer and a forged proof of deposit from the exchange. This does not require the attacker to perform the attack before the original merchant has deposited the coins. It also does not require that all the coins come from a single transaction. Because the proof of deposit affirms, among other things, that the coins have not been double spent, the attacker could show this transcript to a judge and claim a right to the deposited amount, even if they do so after the exchange has marked the coins spent. Of course, the attack would be detected, because the exchange would see the same coins being spent in both transcripts, but a judge may still force the exchange to pay the attacker.

Due to the extremely short time window for the case where the attacker is racing to spend/deposit coins before a merchant engaging in a legitimate transaction has finished depositing them, we consider this attack to have a very low-severity impact on the system. For the case where an attacker collects coin public keys after they have been deposited, forges proofs of deposit for those coins, and then tries to legally enforce those deposits, we consider this to have low-to-medium-severity impact on the system. The severity is increased due to

the larger window of time for the attacker to collect and attack coins and a possibility of successfully enforcing payment. This attack's severity is mitigated, however, by its detectability by the exchange and by the fact that the attacker would likely be identified if they go to court.

3.5 Refresh and Linking Protocol

During the commitment phase of the Refresh Protocol, the customer sends their hashed commitment to the exchange along with the dirty coin's public key, the relevant denomination keys, and the unblinded signature on the dirty coin. An attacker with a quantum computer observing this message would see the dirty coin's public key and be able to compute its discrete logarithm. This allows an attacker to forge signatures with that coin, allowing the attacker to spend the residual value of the coin. The attacker learns nothing about the committed values at this point, because these are the output of a cryptographic hash.

Likewise, in the reveal phase, the customer reveals the public part of the challenge Curve25519 transfer key, whose discrete log can also be computed by an eavesdropping quantum attacker. Thus the attacker can compute the Diffie-Hellman secret in either direction, giving them the information they need to compute the refreshed coin and its blinding factor. If the attacker then obtains the blinded signature, they can easily unblind it and obtain the same new, spendable coin as the customer.

An important observation is that, because the blinding factor for refreshed coins is deterministic for a given transfer Curve25519 key, the resulting refreshed coin is not information-theoretically secure with respect to anonymity. In the Withdrawal Protocol, an exchange is unable to link a blinded coin withdrawn by a customer with its unblinded version when the customer spends it, even if the exchange has unlimited computing power. In the Refresh Protocol, an exchange (or eavesdropper) without a quantum computer could link a dirty coin with its refreshed derivative coin in exponential time, while an attacker with a quantum computer could do so in polynomial time. The original blinding factor from the initial Withdrawal Protocol instance would still remain unknown, of course, so this would not completely de-anonymize the customer. However, it would allow an attacker to connect two or more different payments of the customer to each other, which may be sufficient to uniquely identify them.

The Refresh Protocol also is rather unusual for taking advantage of a coin's ability to be used both for Curve25519 elliptic curve Diffie-Hellman and for Ed25519 signatures. Among the currently proposed post-quantum signature schemes, none innately has this dual-use property. This dual-use requirement of the Refresh Protocol thus poses a special challenge for replacing pre-quantum cryptography in GNU Taler with post-quantum schemes.

The impact of a quantum computer on the Linking protocol is similar to the impact on the Refresh Protocol. In the Linking Protocol, the exchange provides the customer with the linking record for a dirty coin $C_p^{(0)}$, which includes $C_p^{(0)}$, the transfer Curve25519 public key T_γ, and the signature on a derived blinded coin. A quantum attacker can compute the discrete logarithm of either the original coin public key or the transfer Curve25519 key, compute the Diffie-Hellman secret to re-obtain the blinding factor, and unblind the signed coin to recompute the customer's new refreshed coin. The attacker could repeat this process for all the linking records it is able to access.

The impact of a quantum attacker trying to obtain refreshed coins to spend them is limited

by the need to spend these coins before the legitimate owner does. We consider the impact of a quantum attack on the Refresh protocol to be moderate with respect to fund conservation.

However, recomputing refreshed coins has a much more serious impact on anonymity, and for this there is no temporal limit. An attacker who obtains the exchange database much later will be able to see all the instances of the refresh protocol, be able to compute the refreshed coins from any protocol instance, and see when they were spent elsewhere in the database. Identifying the customer in any of those transactions would thus break their anonymity for all transactions where a linked coin was used. Customers may have an interest in preserving the anonymity of transactions now for a long time into the future. Revealing a donation to a contentious organization, for example, may create problems for the customer, even if it is revealed many years later. Due to the ability for an attacker to store now and decrypt later, and because of the potentially serious damage that de-anonymizing transactions may have years into the future, we consider this to be a severe vulnerability in GNU Taler to a quantum attack that requires fixing long before quantum computers arrive.

3.6 Refunds

As Ed25519 signatures can be forged by a quantum attacker, the refund instruction from the merchant to the exchange can also be forged by a third party (the customer, say), to initiate a refund on coins previously spent by a customer. This would have the effect of allowing a customer to double (or more often) spend coins, as they could make purchases, generate fake refund instructions to the exchange, and then reuse the coins. While there is no theoretical limit to the number of times this could occur, this has a side effect of updating the exchange's database, and would thus likely be detectable by an auditor. Given that there are other, less detectable ways for a quantum attacker to forge their own coins, we consider this attack to have a low-to-medium impact on GNU Taler.

Bibliography

[1] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[2] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 124–142. Springer, 2011.

[3] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 16–29. ACM, 2023.

[4] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982.

[5] Gian Demarmels and Lucien Heuzeveldt. Adding schnorr's blind signature in taler. Master's thesis, Bern University of Applied Sciences, 2022.

[6] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[7] Florian Dold. The GNU Taler System: Practical and Provably Secure Electronic Payments. Doctoral Thesis, L'Université de Rennes, 2019. https://taler.net/papers/thesis-dold-phd-2019.pdf.

[8] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. Cryptology ePrint Archive, Paper 2019/877, 2019.

[9] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory*

of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, pages 212–219. ACM, 1996.

[10] Lukas Matyja Johannes Casaburi. Donau: Donation Authority. Tax-deductible Privacy-Preserving Donations. Bachelor Thesis, 2024. https://taler.net/papers/donau-thesis.pdf.

[11] Özgür Kesim, Christian Grothoff, Florian Dold, and Martin Schanzenbach. Zero-knowledge age restriction for GNU taler. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part I*, volume 13554 of *Lecture Notes in Computer Science*, pages 110–129. Springer, 2022.

[12] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[13] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.