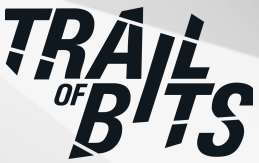

Citizen Technologies: Practical Stealth Addresses

Security Assessment

March 7, 2023

Prepared for:
Ryan Shea
Citizen Technologies

Prepared by: **Opal Wright and Joop van de Pol**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Executive Summary

Engagement Overview

Citizen Technologies engaged Trail of Bits to review the security of its stealth addresses protocol. From January 23 to January 27, 2023, a team of two consultants conducted a security review of the client-provided source code, with two person-weeks of effort. Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report.

Project Scope

Our testing efforts were focused on the identification of flaws that could result in a compromise of confidentiality, integrity, or availability of the target system. We conducted this audit with partial knowledge of the system. We had access to a high-level description of the protocol, a proof-of-concept implementation, and reference links to several related protocols. We performed a thorough cryptographic analysis of the proposed protocol, including checking relevant professional literature, assessment of full and partial key compromise situations, investigation of common implementation errors, and other mathematical analyses.

Summary of Findings

The audit uncovered significant findings that could impact system confidentiality, integrity, or availability. A summary of the findings and details on notable findings are provided below.

EXPOSURE ANALYSIS

Severity	*Count*
Medium	3
Low	1

CATEGORY BREAKDOWN

Category	*Count*
Cryptography	4

Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

- **TOB-CTSA-003**
  Pairwise transactions (from Alice to Bob and from Bob to Alice) can be deanonymized because the shared secret is equal. As a result, the derived root public keys for pairwise transactions with the same index have a constant difference that can be detected by an attacker.

- **TOB-CTSA-004**
  Allowing invalid public keys may enable DH private key recovery. An attacker can use the invalid public keys to cause the recipient to leak information on their DH private key according to well-known invalid point attacks. Once enough information is obtained using different invalid public keys, it can be combined to recover the full DH private key.

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Jeff Braswell, Project Manager
jeff.braswell@trailofbits.com

The following engineers were associated with this project:

Joop van de Pol, Consultant
joop.vandepol@trailofbits.com

Opal Wright, Consultant
opal.wright@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
Jan 20, 2023	Pre-project kickoff call
Jan 27, 2023	Status update meeting #1
Feb 3, 2023	Report readout meeting
March 7, 2023	Delivery of final report

Project Goals

The engagement was scoped to provide a security assessment of the Citizen Technologies Stealth Addresses protocol. Specifically, we sought to answer the following non-exhaustive list of questions:

- Is there a way for a participant, malicious party, or outside observer to steal funds or track funds?
- Are there any mathematical attacks that allow recovery of protected key material or signature forgeries?
- Are there any attacks based on common implementation errors that could compromise the security of the system?
- For any issues discovered, are there suitable mitigations, and what are the advantages and disadvantages of each mitigation?

Project Targets

The engagement involved a review and analysis of the Stealth Address protocol.

Stealth address protocol

Protocol Spec	Practical stealth addresses
Implementation	Example implementation
Platform	N/A
Version	N/A

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches include the following:

- Independent cryptographic analysis
- Literature review
- Comparison to related and prior protocols
- Implementation consideration review
- Project scale consideration

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review:

- Not every implementation error can be anticipated or considered; our implementation error analysis focused on common errors seen in previous audits (e.g., failure to reject invalid points during elliptic curve operations).
- The protocol documentation received was very high-level. While the general mechanisms used for the protocol were clear, specific algorithms were not specified in most places. For example, it was clear that elliptic-curve Diffie-Hellman (ECDH) would be used, but specific hash functions were left out of the specification (the example implementation used SHA-256). Even within an otherwise-secure protocol, the use of inappropriate cryptographic primitives can cause security problems.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Related-nonce attacks across keys allow root key recovery	Cryptography	**Medium**
2	Limited forgeries for related keys	Cryptography	**Low**
3	Mutual transactions can be completely deanonymized	Cryptography	**Medium**
4	Allowing invalid public keys may enable DH private key recovery	Cryptography	**Medium**

Detailed Findings

1. Related-nonce attacks across keys allow root key recovery	
Severity: **Medium**	Difficulty: **Undetermined**
Type: Cryptography	Finding ID: TOB-CTSA-1
Target: Root key	

Description
Given multiple addresses generated by the same sender, if any two signatures with the associated private keys use the same nonce, then the recipient's private root key can be recovered.

Nonce reuse attacks are a known risk for single ECDSA keys, but this attack extends the vulnerability to *all* keys generated by a given sender.

Exploit Scenario
Alice uses Bob's public key to generate addresses $B_1 = Hash(s_{ab}||1) * G + B_{root}$ and $B_2 = Hash(s_{ab}||2) * G + B_{root}$ and deposits funds in each. Bob's corresponding private keys will be $b_1 = Hash(s_{ab}||1) + b_{root}$ and $b_2 = Hash(s_{ab}||2) + b_{root}$. Note that, while Alice does not know b_1 or b_2, she does know the difference of the two: $b_{diff} = b_2 - b_1 = Hash(s_{ab}||2) - Hash(s_{ab}||1)$. As a result, she can write $b_2 = b_1 + b_{diff}$.

Suppose Bob signs messages with hashes m_1 and m_2 to transfer the funds out of B_1 and B_2 (respectively), and he uses the same nonce k in both signatures. He will output signatures (r, s_1) and (r, s_2), where $r = (k * G)_x$, $s_1 = k^{-1}(m_1 + rb_1)$, and $s_2 = k^{-1}(m_2 + rb_1 + rb_{diff})$.

Subtracting the s-values gives us $s_1 - s_2 = k^{-1}(m_1 - m_2 - rb_{diff})$. Because all the terms except k are known, Alice can recover k and thus b_1, b_2, and $b_{root} = b_2 - Hash(s_{ab}||2)$.

Recommendations
Consider using deterministic nonce generation in any stealth-enabled wallets. This is an approach used in multiple elliptic curve digital signature schemes, and can be adapted to ECDSA relatively easily; see RFC 6979.

Also consider root key blinding. Set $B_i = Hash(s_{ab}||i) * G + Hash(s_{ab}||i||"root") * B_{root}$. With blinding, private keys take the form $b_i = Hash(s_{ab}||i) + b_{root} \cdot Hash(s_{ab}||i||"root")$. Since the b_{root} terms no longer cancel out, Alice cannot find b_{diff}, and the attack falls apart.

Finally, consider using homogeneous key derivation. Set $B_i = Hash(s_{ab}||i) * B_{DH} + B_{root}$. The private key for Bob is then $b_i = Hash(s_{ab}||i) \cdot b_{DH} + b_{root}$. Because Alice does not know b_{dh}, she cannot find b_{diff}, and the attack falls apart.

References

- ECDSA: Handle with Care
- RFC 6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)

2. Limited forgeries for related keys	
Severity: **Low**	Difficulty: **Low**
Type: Cryptography	Finding ID: TOB-CTSA-2
Target: Stealth addresses generated by the same sender	

Description

If Bob signs a message for an address generated by Alice, Alice can convert it into a valid signature for another address. She cannot, however, control the hash of the message being signed, so this attack is of limited value.

As with the related-nonce attack, this attack relies on Alice knowing the difference in discrete logarithms between two addresses.

Exploit Scenario

Alice generates addresses $B_1 = Hash(s_{ab}||1) * G + B_{root}$ and $B_2 = Hash(s_{ab}||2) * G + B_{root}$ and deposits funds in each account. As before, Alice knows b_{diff}, the difference of the discrete logs for B_1 and B_2, and $B_{diff} = B_2 - B_1$.

Bob transfers money out of B_2, generating signature (r, s) of a message m with hash e, where r is the x-coordinate of $k * G$ (where k is the nonce). The signature is validated by computing $P = es^{-1} * G + rs^{-1} * B_2$ and verifying that the x-coordinate of P matches r.

Alice can convert this into a signature under B_1 for a message with hash $e' = e + rb_{diff}$. Verifying this signature under B_1, computing P becomes:

$$P = (e + rb_{diff})s^{-1} * G + rs^{-1} * B_1$$

$$= es^{-1} * G + rb_{diff}s^{-1} * G + rb_1s^{-1} * G$$

$$= es^{-1} * G + rs^{-1}(b_1 + b_{diff}) * G$$

$$= es^{-1} * G + rs^{-1} * B_2$$

This is the same relation that makes (r, s) a valid signature on a message with hash e, so P will be correct.

Note that Alice has no control over the value of e', so to make an effective exploit, she would have to find a preimage m' of e' under the given hash function. Computing preimages is, to date, a hard problem for SHA-256 and related functions.

Recommendations

Consider root key blinding, as above. The attack relies on Alice knowing $b_{diff'}$, and root key blinding prevents her from learning it.

Consider homogeneous key derivation, as above. Once again, depriving Alice of b_{diff} obviates the attack completely.

3. Mutual transactions can be completely deanonymized

Severity: **Medium**	Difficulty: **Medium**
Type: Cryptography	Finding ID: TOB-CTSA-3
Target: Anonymity of mutual transactions	

Description

When Alice and Bob both make stealth payments to each other, they generate the same Shared Secret #i for transaction i, which is used to derive destination keys for Bob and Alice:

Symbol	Description	Alice's derivation	Bob's derivation
s_i	Shared secret #i	hash(a_{dh} * B_{dh}, i)	hash(b_{dh} * A_{dh}, i)
B_i	Bob's destination key #i	$B_r + s_i * G$	
A_i	Alice's destination key #i	$A_r + s_i * G$	

However, as a result, the destination keys of Bob and Alice for the same transaction number #i will always have the constant difference $B_i - A_i = B_r - A_r$.

Exploit Scenario

An attacker records all root public keys from the database, computes the pairwise differences $B_i - A_i$ for every pair of users A and B they want to monitor, and stores the differences in a list.

The attacker subsequently reviews all pairs of public keys associated with transactions on the blockchain, computes the differences between the public keys, and compares the x coordinates to the stored list of pairwise differences.

If a match is found, the attacker knows that those transactions correspond to stealth transactions between A and B with some index i. The attacker can also determine which transaction went from A to B and vice versa by considering the y coordinates of the public keys and pairwise differences.

Because the attack compromises the anonymity of pairs of users without giving access to funds, we consider this issue to have medium severity. As the attack requires both users in the pair to make transactions to each other, and as it requires a potentially significant

calculation effort (proportional to the square of the number of transactions), we consider this issue to have medium difficulty.

Recommendations

There are several ways to mitigate this issue:

- Root key blinding, as described in the recommendations for TOB-CTSA-1.
- Diversify the shared secret #i, e.g., by adding the DH key of the receiver to the hash $s_{i,B} = Hash\,(a_{dh} \cdot B_{dh} \,||\, i \,||\, B_{dh}) = Hash\,(b_{dh} \cdot A_{dh} \,||\, i \,||\, B_{dh})$, or to any other known identifier for the receiver (e.g., public root key B_r, a unique predetermined string that is stored in the public key database).
- Homogenize the destination keys by including both the root key and the DH key of the receiver, i.e., $B_i = B_r + s_i * B_{dh}$, with corresponding private key $b_i = b_r + s_i \cdot b_{dh}$.

The advantages and disadvantages of each of these mitigation methods is discussed in detail in Appendix B.

4. Allowing invalid public keys may enable DH private key recovery	
Severity: **Medium**	Difficulty: **High**
Type: Cryptography	Finding ID: TOB-CTSA-4
Target: `DH private key`	

Description
Consider the following three assumptions:

1. Alice can add points that are not on the elliptic curve to the public key database,
2. Bob does not verify the public key points, and
3. Bob's scalar multiplication implementation has some specific characteristics.

Assumptions 1 and 2 are currently not specified in the specification, which motivates this finding.

If these assumptions hold, then Alice can recover Bob's DH key using a complicated attack, based on the CRYPTO 2000 paper by Biehl et al. and the DCC 2005 paper by Ciet et al. What follows is a rough sketch of the attack. For more details, see the reference publications, which also detail the specific characteristics for Assumption 3.

Exploit Scenario
Alice roughly follows the following steps:

1. Find a point P' which
    a. is not on the curve used for ECDH, and
    b. when used in Bob's scalar multiplication, is effectively on a different curve E' with (a subgroup of) small prime order p'.
2. Brute-force all possible values of $x \cdot P'$ for $0 \leq x < p'$, and sends funds to all addresses with shared secret $Hash(x \cdot P' \,||\, 0)$, i.e., $B_x = B_r + Hash(x \cdot P' || 0) * G$.
3. Monitor all resulting addresses associated with B_x until Bob withdraws funds from the unique stealth address associated with $x' = b_{dh} \bmod p'$. This happens because $b_{dh} \cdot P' = (b_{dh} \bmod p') \cdot P'$.
4. Repeat steps 1–3 for new points P_j' with different small prime orders p_j' to recover $b_{dh} \bmod p_j'$.
5. Use the Chinese Remainder Theorem to recover b_{dh} from $b_{dh} \bmod p_j'$.

As a result, Alice can now track all stealth payments made to Bob (but cannot steal funds). To understand the complexity of this attack, it is sufficient for Alice to repeat steps 1–3 for the first 44 primes (numbers between 2 and 193). This requires Alice to make 3,831 payments in total (corresponding to the sum of the first 44 primes).

There is a tradeoff where Alice uses fewer primes, which means that fewer transactions are needed. However, it means that Alice does not recover the full b_{dh}. To compensate for this, Alice can brute-force the discrete logarithm of B_{dh} guided by the partial information on b_{dh}.

Because the attack compromises anonymity for a particular user without giving access to funds, we consider this issue to have medium severity. As this is a complicated attack with various assumptions that requires Bob to access the funds from all his stealth addresses, we consider this issue to have high difficulty.

Recommendations

The specification should enforce that public keys are validated for correctness, both when they are added to the public database and when they are used by senders and receivers. These validations should include point-on-curve checks, small-order-subgroup checks (if applicable), and point-at-infinity checks.

References

- Differential Fault Attacks on Elliptic Curve Cryptosystems, Biehl et al., 2000
- Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults, Ciet et al., 2005

Summary of Recommendations

The Citizen Technologies Practical Stealth Addresses proposal is a work in progress with multiple planned iterations. Trail of Bits recommends that Citizen Technologies address the findings detailed in this report and take the following additional steps prior to deployment:

- Update the specification to ensure that the public key database enforces the validity of the elliptic curve points corresponding to the public keys. Additionally, specify that both sender and receiver should perform their own input validation of elliptic curve points.
- Update the specification to recommend deterministic nonce generation for signatures to avoid nonce re-use, if this is not yet specified for the used signature scheme.
- Consider the various proposed mitigation methods that are summarized and analyzed in more detail in Appendix B and implement the mitigations that provide the desired trade-offs in terms of performance and security.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Mitigation Strategy Tradeoffs

The following table provides a comparison of the different proposals linked in the protocol document. Please note that this merely comprises a comparison based on the analysis of the target proposal. It should not be interpreted as a full analysis of all proposals.

The following notation is used:

- M is the number of elliptic curve multiplications.
- A is the number of elliptic curve additions.
- H is the number of hash function calls.
- N_{pk} is the number of users (i.e., sets of public keys) in the database.
- N_{rtxs} is the number of stealth transactions received.
- N_{I} is the number of input keys.
- Tx is Transaction.

Reference	Target proposal	Robin Linus, 2022	Ruben Somsen, 2022 (base only)	BIP351	EIP-5564
Key pairs	All users: 2	All users: 1	All users: 1	Receivers: 1 Senders: 1 or more for each receiver	1 for each send transaction (ephemeral) 2 for all receivers
Detection cost (operations)	(2M, A, H) per (N_{pk} + N_{rtxs})	(M, A) per (N_{pk})	(2M, A, H) per (N_{I})	(M, H) per notification (A, H) per Tx	(M, H) per announce, (M, A, H) per Tx
Detection cost (monitoring)	N_{pk} addresses	1 address	All transactions, N_{I} addresses	All blocks, 1 address per match	All announce, 1 address per match
Unique receive address	Yes (index)	No	Per input key	Yes (counter)	Yes (ephemeral sender key)
Separate	Yes	No	No	No	Yes

scan/spend keys					
Stateless	No	Yes	Yes	No	Yes
Resistant to TOB-CTSA-1	No	No	No	No	No
Resistant to TOB-CTSA-2	No	No	No	No	No
Resistant to TOB-CTSA-3	No	No	If same input key is not reused for multiple recipients	Yes	Yes
Resistant to TOB-CTSA-4	No	No	No	No	No

In general, the core concept of each of the stealth address protocol proposals seems cryptographically sound. The EUROCRYPT 2022 paper by Groth et al. gives a formal security proof for ECDSA with additive key derivation, which corresponds to the key tweaking that is used in most stealth address proposals. This shows that it is at least difficult to steal funds by forging signatures based on keys derived from the root key.

The paper does note that there is a security gap when using additive key derivation. A security gap in a formal security proof does not mean that there exists a practical attack against the scheme, but it theoretically "leaves room" for such an attack to exist. To close this gap, the authors propose homogeneous key derivation, which corresponds to one of the proposed mitigations in this report.

As a result, the findings in this report follow from subtle implementation flaws and reuse of DH shared secrets for distinct purposes. The following mitigations were proposed for these findings:

- Deterministic nonce generation (e.g., using RFC 6979 for ECDSA).
- Root key blinding: $B_i = Hash(s_{ab}||i) * G + Hash(s_{ab}||i||"root") * B_{root}$, with corresponding private key $b_i = Hash(s_{ab}||i) + b_{root} \cdot Hash(s_{ab}||i||"root")$.
- Diversification of shared secret with recipient data: $s_{i,B} = Hash(a_{dh} \cdot B_{dh} || i || B_{dh}) = Hash(b_{dh} \cdot A_{dh} || i || B_{dh})$.

- Homogenize the destination keys by adding DH: $B_i = B_r + s_i \cdot B_{dh}$, with corresponding private key $b_i = b_r + s_i \cdot b_{dh}$.

The following table lists these mitigations and their impact on the current proposal. Note that deterministic nonce generation is not included, as it does not directly affect the protocol and addresses only TOB-CTSA-1. It can be implemented along with any other proposed mitigation, and it is independently useful to avoid nonce reuse.

Reference	Target proposal	Diversify shared secret with recipient data	Root key blinding	Homogenize destination keys by adding DH
Key pairs	2	2	2	2
Detection cost (operations)	(2M, A, H) per (N_{pk} + N_{rtxs})	(2M, A, H) per (N_{pk} + N_{rtxs})	(2M, A, 2H) per (N_{pk} + N_{rtxs})	(2M, A, H) per (N_{pk} + N_{rtxs})
Detection cost (monitoring)	N_{pk} addresses	N_{pk} addresses	N_{pk} addresses	N_{pk} addresses
Unique receive address	Yes	Yes	Yes	Yes
Separate scan/spend keys	Yes	Yes	Yes	Yes
Stateless	No	No	No	No
Resistant to TOB-CTSA-1	No	No	Yes	Yes
Resistant to TOB-CTSA-2	No	No	Yes	Yes
Resistant to TOB-CTSA-3	No	Yes	Yes	Yes

This table does not list TOB-CTSA-4 because the corresponding mitigation is merely to update the specification. This can be done in conjunction with any other mitigation or scheme.

References

- Improved Stealth Addresses, Linus, 2022
- Silent Payments, Somsen, 2022
- BIP351: Private Payments, Hodler et al., 2022
- EIP-5564: Non-Interactive Stealth Address Generation, Wahrstätter et al., 2022
- On the security of ECDSA with additive key derivation and presignatures, Groth et al., EUROCRYPT 2022

C. Minor Considerations

The following considerations do not rise to the level of cryptographic vulnerabilities, but should be taken into account when refining the protocol or developing software to implement it.

Do not use raw hashes to derive key material.

When deriving key material from a shared secret, it is better to use a key derivation function such as HKDF. KDFs are more flexible, allowing multiple keys to be safely derived from the same shared secret.

Consider the implications of statefulness.

The transaction index used in the derivation of the base point multiplier needs to be tracked for all potential senders in the public key directory. Maintaining this state, recovering from a failure to maintain state, and dealing with other parties who fail to maintain the same state are complex technical problems.

Consider the overhead of the public directory.

Which responsibilities will the public directory take on? Will the public directory validate keys before publishing them? Will the public directory have DoS/DDoS protections in place? How are compromised/expired keys updated within the public directory? Is there a maximum size for the public directory?

D. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system.

On February 16 2023, Trail of Bits reviewed the fixes and mitigations implemented by the Citizen Technologies team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

Citizen Technologies weighed the pros and cons of several options to mitigate the related-key attacks identified in this report, and settled on using the key homogenization approach. Further, they integrated on-curve checks into the protocol, reducing the risk of off-curve attacks. Finally, they updated the protocol to use HKDF when deriving secret multipliers, and specified a multiplier size that should reduce mod bias.

In summary, Citizen Technologies has resolved all four of the issues described in this report. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Severity	Status
1	Related-nonce attacks across keys allow root key recovery	Medium	Resolved
2	Limited forgeries for related keys	Low	Resolved
3	Mutual transactions can be completely deanonymized	Medium	Resolved
4	Allowing invalid public keys may enable DH private key recovery	Medium	Resolved

Detailed Fix Review Results

TOB-CTSA-1: Related-nonce attacks across keys allow root key recovery
Resolved in 30 January 2023 protocol update.

The new key homogenization approach replaces $B_i = Hash(s_{ab}||i) * G + B_{root}$ with $B_i = Hash(s_{ab}||i) * B_{dh} + B_{root}$, preventing known relations between secret keys b_i and b_j for $i \neq j$.

Further, the implementation notes for the protocol state that implementers **must** use deterministic nonce derivation for ECDSA signatures. This reduces the risk of nonce reuse and related nonce attacks against individual keys.

TOB-CTSA-2: Limited forgeries for related keys
Resolved in the January 30, 2023 protocol update.

As with TOB-CTSA-1, key homogenization prevents known relations between secret keys, which prevents the attack.

TOB-CTSA-3: Mutual transactions can be completely deanonymized
Resolved in the January 30, 2023 protocol update.

As with TOB-CTSA-1 and TOB-CTSA-2, key homogenization prevents known relations between secret keys, which prevents the attack.

TOB-CTSA-4: Allowing invalid public keys may enable DH private key recovery
Resolved in the January 30, 2023 protocol update.

Implementation notes state that “clients **must** check the validity of public keys before doing calculations with them.” An implementation that fails to check the validity of public keys would thus be non-compliant with the standard.

Minor Considerations:

The January 30, 2023 protocol update states:

> Hashes should be derived using the key-derivation function HKDF and the hashing function SHA256, with no salt, with info set to the string Key-${i} where "i" is the index of the key, and with a key length of 40 bytes. The number should then be mapped onto the curve order using a modular operation.

Replacing direct use of SHA-256 for key derivation with HKDF addresses several minor risks.

In previous versions of the protocol, if $H_i = SHA256\left(s_{ab} || i\right)$ were leaked for some i, an attacker might be able to exploit length extension attacks to determine $H_{10^n i+k}$ for $0 \leq k < 10$ and $n > 0$, which would deanonymize those future transactions.

Additionally, the use of a 320-bit output reduces the effects of mod bias in the resulting multiplier of the Diffie-Hellman key.