Threshold Scriptless Scripts

Omer Shломовיץ

KZEN Research
“Magicking digital signatures so that they can only be created by faithful execution of a smart contract”.

Andrew Poelstra
In This Talk...

- Intro to Schnorr Scriptless Scripts (SSS)
- The road to ECDSA Scriptless Scripts (ESS)
  - Rebuttal ECDSA known security issues
  - Discussing 2P-ECDSA as main tool
  - Expending the Tool Box with Threshold ECDSA
- Experimenting with ESS
Schnorr Signature

EC public parameters : \( q,G \)

- Choose random \( k \)
- Compute \( R = k \cdot G \)
- Compute \( s = k + H(R, P, m) \cdot x \mod q \) where \( x \) is the private key, \( R = x \cdot G \)
- Output \((R,s)\)
The Shtik

* Adaptor Signature: Main building block is a tweak to Schnorr signature \((R, s)\)

\[ s' = s + t \]

- Schnorr Sig
- Adaptor \((t, T)\)
SS Atomic Swap

Wallet A

$PK_A + PK_B$

Wallet B

$PK_A + PK_B$
SS Atomic Swap

Wallet A

PK_A + PK_B

T, s'_A, s'_A

Wallet B

PK_A + PK_B
SS Atomic Swap

Wallet A

PK_A + PK_B

T, s'_A, s'_A

SB

Wallet B

PK_A + PK_B
SS Atomic Swap

Wallet A

PK_A + PK_B

T, s'_A, s'_A

SB

Publish S_A

Wallet B

PK_A + PK_B
SS Atomic Swap

Wallet A

\[ PK_A + PK_B \]

Wallet B

\[ PK_A + PK_B \]

\[ T, s'_A, s'_A \]

\[ S_B \]

Publish \( s_A \)

\[ t = s'_A - s_A \]

\[ s_A = s'_A - t \]
Scriptless Scripts

Open Problems

- Preserving scriptless scripts in multisig
- ECDSA support
- Locktimes and other extrospection
- Formalizing/understanding limits of scriptless scripts
ECDSA SS - The Hard Questions

• Why we needed Schnorr in the first place?
• What are the challenges in using ECDSA for SS
## Why Schnorr?

EC public parameters: $q, G$

### ECDSA
- Choose random $k$
- Compute $R = k \cdot G$
- Compute $r = r_x \mod q$ where $R = (r_x, r_y)$
- Compute $s = k^{-1} \cdot (H(m) + r \cdot x) \mod q$
- Output $(r, s)$

### Schnorr
- Choose random $k$
- Compute $R = k \cdot G$
- Compute $s = k + H(R, P, m) \cdot x \mod q$
- Output $(R, s)$
Why Schnorr #2

ECDSA:
- No security proof
- Malleable
- Not linear

EC-Schnorr:
- Provably secure under ROMDL
- Provably non-malleable
- Linearity!

CIW19 Keynote

By Pieter Wuille
March 20, 2019
Why Schnorr #2

ECDSA:
- No security proof
- Malleable
- Not linear

EC-Schnorr:
- Provably secure under ROMDL
- Provably non-malleable
- Linearity!

ECDSA is a NO-GO?
ECDSA: No security proof?
ECDSA: No security proof?

The Security of DSA and ECDSA
Bypassing the Standard Elliptic Curve Certification Scheme

Serge Vaudenay
Swiss Federal Institute of Technology (EPFL)
Serge.Vaudenay@epfl.ch

• 2003, Generic Group Model
ECDSA: No security proof?

On the Provable Security of (EC)DSA Signatures

Manuel Fersch
manuel.fersch@rub.de

Eike Kiltz
eike.kiltz@rub.de

Bertram Poettering
bertram.poettering@rub.de

Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany

• 2016, Bijective Random Oracle (BRO) model
ECDSA: No security proof?

- ECDSA is one of the widely used signature schemes: TLS, PGP, S/MIME, multiple cryptocurrencies etc..
- ECDSA is widely standardized: IEEE P1363, ANSI X9.62, FIPS 186-4
- It was subject to massive cryptanalytic efforts with zero known attacks

**Implementations**

Below is a list of cryptographic libraries that provide support for ECDSA:

- Botan
- Bouncy Castle
- cryptlib
- Crypto++
- libgcrypt
- OpenSSL
- wolfCrypt
- mbed TLS
ECDSA: Malleable?

• Known malleability:
  • \((r,s)\) and \((r, q - s)\) are both valid signatures on message \(m\)
ECDSA: Malleable?

- Known malleability:
  - \((r, s)\) and \((r, q - s)\) are both valid signatures on message \(m\)

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable
ECDSA: Malleable?

- Known malleability:
  - \((r, s)\) and \((r, \ q - s)\) are both valid signatures on message \(m\)

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable

- BIP62 (WIP):
  - Low S values in signatures

    The value \(S\) in signatures must be between \(0x1\) and \(0x7FFFFFFF\) \(FFFFFFFF\) \(FFFFFFFF\) \(FFFFFFFF\) \(5D576E73\) \(57A4501D\) \(DFE92F46\) \(681B20A0\) (inclusive). If \(S\) is too high, simply replace it by \(S' = 0xFFFFFFFF\) \(FFFFFFFF\) \(FFFFFFFF\) \(FFFFFFFF\) \(FFFFFFFF\) \(FFFFFE\) \(BAAEDCE6\) \(AF48A03B\) \(BFD25E8C\) \(D0364141\) - \(S\).
ECDSA: Malleable?

- Known malleability:
  - \((r, s)\) and \((r, q - s)\) are both valid signatures on message \(m\)

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable

- BIP62 (WIP):
  - Low S values in signatures
    - The value \(S\) in signatures must be between \(0x1\) and \(0x7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 5D576E73 57A4501D DFE92F46 681B20A0\) (inclusive). If \(S\) is too high, simply replace it by \(S' = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E5BFAEDCE6 AF48A03B BFD25E8C D0364141 - S\).

- Segwit
**Schnorr: Malleable?**

**EC Schnorr signature: multiple standard?**

- **Asked:** 3 years, 3 months ago
- **Active:** 4 months ago
- **Viewed:** 2k times

<table>
<thead>
<tr>
<th>scheme</th>
<th>public key</th>
<th>first component</th>
<th>second component</th>
<th>sign. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Sc91]</td>
<td>$-d \times G$</td>
<td>$H(Q, M)$</td>
<td>$k + d \cdot h$</td>
<td>$b + 2b$</td>
</tr>
<tr>
<td>EC-SDSA</td>
<td>$-d \times G$</td>
<td>$H(Q_x \parallel Q_y \parallel M)$</td>
<td>$k + d \cdot h$</td>
<td>$2b + 2b$</td>
</tr>
<tr>
<td>EC-SDSA-opt</td>
<td>$-d \times G$</td>
<td>$H(Q_x \parallel M)$</td>
<td>$k + d \cdot h$</td>
<td>$2b + 2b$</td>
</tr>
<tr>
<td>EC-FSDSA</td>
<td>$-d \times G$</td>
<td>$Q_x \parallel Q_y$</td>
<td>$k + d \cdot H(Q_x \parallel Q_y \parallel M)$</td>
<td>$4b + 2b$</td>
</tr>
<tr>
<td>EC-Schnorr</td>
<td>$d \times G$</td>
<td>$H(M \parallel Q_x)$</td>
<td>$k - d \cdot h$</td>
<td>$2b + 2b$</td>
</tr>
<tr>
<td>libsecp256k1</td>
<td>$d \times G$</td>
<td>$Q_x$</td>
<td>$k - d \cdot H(Q_x \parallel M)$</td>
<td>$2b + 2b$</td>
</tr>
</tbody>
</table>
Linearity

- Valid point!
- Deal breaker?
Linearity

• Valid point!

• Deal breaker?

Given the non-linearity of ECDSA, Are ECDSA Scriptless Scripts

• Possible ?
• Possible but Inefficient ?
• Possible but with compromise on Security ?
Linearity: Observation

- **Threshold ECDSA** has struggled with the same problem.
- Not surprisingly
  - Existing works for **ECDSA-SS** are based on threshold ECDSA protocol
Scaling Bitcoin 2018 #1

Instantiating Scriptless 2P-ECDSA
Fungible 2-of-2 Multisigs for Today’s Bitcoin

Conner Fromknecht
Head of Cryptographic Engineering, Lightning Labs
Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

Giulio Malavolta*, Pedro Moreno-Sanchez*, Clara Schneidewind†, Aniket Kate‡, Matteo Maffei‡

*§Friedrich-Alexander-University Erlangen-Nürnberg, †TU Wien, ‡Purdue University
Scaling Bitcoin 2018 #2
Anonymous Multi-Hop Locks for Blockchain
Scalability and Interoperability

Giulio Malavolta*, Pedro Moreno-Sanchez*, Clara Schneidewind†, Aniket Kate†, Matteo Maffei†
§Friedrich-Alexander-University Erlangen-Nürnberg, †TU Wien, ‡Purdue University

Scriptless Scripts (SS-Schnorr)

- Technique originally proposed by A. Poelstra
- “Encode” payment condition within the Schnorr signatures
- Unfortunately, Schnorr is not used yet in many cryptocurrencies
- In our work:
  - formal description and security analysis
  - scriptless scripts based on ECDSA
Previous Work

- Common to both is 2P-ECDSA:

Fast Secure Two-Party ECDSA Signing*

Yehuda Lindell**

Dept. of Computer Science
Bar-Ilan University, ISRAEL
lindell@biu.ac.il
Threshold Signatures

(t,n)-threshold signature scheme distributes signing power to n parties such that any group of at least t parties can generate a signature.
Threshold Signatures

(t,n)-threshold signature scheme distributes signing power to n parties such that any group of at least t parties can generate a signature.

(2,3) - Signing

\[ x_s, Q \]

\[ x_i, Q, tx \]

\[ x_c, Q, tx \]
2P-Keygen [L17]

\[ Q_1 = x_1 \cdot G \]

\[ Q = x_1 \cdot Q_2 \]

\[ Q_2 = x_2 \cdot G \]

\[ Q = x_2 \cdot Q_1 \]

The protocol promises: (1) Privacy, (2) Correctness
2P-Signing [L17]

Signining message $m$: $m' = \text{Hash}(m)$

Party 1:
- $R_1 = k_1 \cdot G$
- $R = k_1 \cdot R$
- $s = \text{Dec}_d(s^*)/k_1$

Party 2:
- $R_2 = k_2 \cdot G$
- $R = k_2 \cdot R_1$
- $s^* = \text{Enc}_e(m'/k_2) \boxplus \text{Enc}_e(x_1) \odot \text{Enc}_e(x_2 \cdot r/k_2)$

Output: $\sigma = (s, r)$, s.t. $\text{Verify}(\sigma, Q, m') = 1$

The protocol promises: Unforgeability

KZen-networks/multi-party-ecdsa
2P-ECDSA Lock
[MMSKM18]

Party1

\[ R_1 = k_1 \cdot T \]
\[ R = k_1 \cdot R \]
\[ s' = \text{Dec}_d(s^*)/k_1 \]

message \( m' = \text{Hash}(m), \text{Adaptor } (t, \ T) \)

\[ R_1 \]

Party2

\[ R_2 = k_2 \cdot T \]
\[ R = k_2 \cdot R_1 \]
\[ s^* = \text{Enc}_e(m' / k_2) \oplus \text{Enc}_e(x_1) \odot \text{Enc}_e(x_2 \cdot r / k_2) \]

Output: \( \sigma' = (s' = s \cdot t, r), \text{ s.t. } \text{Verify}(s' \cdot t^{-1}, r, Q, m') = 1 \)
Possible Issues

- Given the non-linearity of ECDSA, Are ECDSA Scriptless Scripts Possible?
  - Possible? **YES, using Lindell 2P-ECDSA**
  - Possible but Inefficient? **MAYBE**
  - Possible but with compromise on Security? **YES**
Possible Issues #2

Digital signing by utilizing multiple distinct signing keys, distributed between two parties

Abstract

Described herein is a method and system for digital signing by utilizing Elliptic Curve Digital Signature Algorithm (ECDSA) with a group generator of an elliptic-curve group of order and an elliptic curve point Q. The method may be configured to receive a digital message and associated with a request from a third-party in order to sign the digital message. The system designed to sign such messages may comprise two parties denoted P1 and P2 configured to conduct a multiparty signing procedure by utilizing ECDSA. The digital signing procedure may follow preliminary steps configured to set the system with the necessary conditions for the multiparty signing procedure. Such preliminary steps may set the parties P1, and P2, in accordance with the configuration defined herein.
Bad News?
## Threshold ECDSA Papers Circa 2017-Today

<table>
<thead>
<tr>
<th>Paper</th>
<th>Params</th>
<th>Assumptions</th>
<th>Signing Rounds</th>
<th>Signing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[L17]</td>
<td>2/2</td>
<td>ECDSA, Paillier</td>
<td>4</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[GG18]</td>
<td>t/n</td>
<td>ECDSA, Strong RSA</td>
<td>9</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[LNR18]</td>
<td>t/n</td>
<td>ECDSA, DDH</td>
<td>8</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[DKLS18]</td>
<td>2/n</td>
<td>ECDSA</td>
<td>2</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[DKLS19]</td>
<td>t/n</td>
<td>ECDSA</td>
<td>$\log(t) + 6$</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[CCLST19]</td>
<td>2/2</td>
<td>ECDSA, Class groups</td>
<td>4</td>
<td>milliseconds</td>
</tr>
<tr>
<td>[SA19]</td>
<td>n/n</td>
<td>ECDSA</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>[DKOSS19]</td>
<td>t/n</td>
<td>ECDSA</td>
<td>1</td>
<td>sub millisecond</td>
</tr>
</tbody>
</table>
## Threshold ECDSA Papers Circa 2017-Today

<table>
<thead>
<tr>
<th>Params</th>
<th>Assumptions</th>
<th>Signing Rounds</th>
<th>Signing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[L17]</td>
<td>2/2 ECDSA, Paillier</td>
<td>4 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[GG18]</td>
<td>t/n ECDSA, Strong RSA</td>
<td>9 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[LNR18]</td>
<td>t/n ECDSA, DDH</td>
<td>8 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[DKLS18]</td>
<td>2/n ECDSA</td>
<td>2 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[DKLS19]</td>
<td>t/n ECDSA</td>
<td>Log(t) + 6 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[CCLST19]</td>
<td>2/2 ECDSA, Class groups</td>
<td>4 milliseconds</td>
<td></td>
</tr>
<tr>
<td>[SA19]</td>
<td>n/n ECDSA</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>[DKOSS19]</td>
<td>t/n ECDSA</td>
<td>1</td>
<td>sub millisecond</td>
</tr>
</tbody>
</table>
Experiments
Use Case #1: Scriptless Script MultiSig

- Access policy privacy
- Cost: one standard transaction
- Max number of parties

KZen-networks/multi-party-ecdsa
Use Case #2: Threshold Wallet

- Distributed key generation (DKG)
- Distributed Signing
- Secret Share Recovery
- Deterministic Child Address Derivation
- Rotation
Use Case #3: Coin Join Mixer

- Pubkey is equal to a sum of locally generated public keys:
  - \( Pk = Pk_1 + Pk_2 + \ldots + Pk_n \)
- Basically a constructed way for parties to reach off-chain consensus on output addresses like Chaumian coin-join
- No need for central coordinator
Use Case #4: Atomic Swaps

- **Option 1**: Use the scriptless script construction
  - Locking using Adaptor signatures.
- **Option 2**: Depends on access structure secret shares can be swapped using “gradual release of secrets”

[scriptless-scripts/atomic-swap]
[KZen-networks/centipede/]
Wallet A

Wallet B

Wallet A

Wallet B

sk_{B1}  sk_{B2}

sk_{L1}  sk_{L2}
Use Case #6: Payment Channel Network

Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

Giulio Malavolta*, Pedro Moreno-Sanchez*, Clara Schneidewind†, Aniket Kate‡, Matteo Maffei‡
§Friedrich-Alexander-University Erlangen-Nürnberg, †TU Wien, ‡Purdue University

KZen-networks/multi-hop-locks/
Use Case #7: Zero Knowledge
Contingent Payments

Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts

Wacław Banasik, Stefan Dziembowski, and Daniel Malinowski

University of Warsaw
ZKCP

Prover / Seller
Knows \( x \)

Run threshold KeyGen

Verifier / Buyer

Generate TX\(_1\), TX\(_2\):
- TX\(_1\),\_in : Buyer
- TX\(_1\),\_out : threshold-Sig
- TX\(_2\),\_in : TX\(_1\)
- TX\(_2\),\_out : Seller

Sign TX\(_2\)

ZK Prove \( x \) is encrypted by \( \sigma \)

Broadcast TX\(_2\)

Extract \( x \)

Broadcast TX\(_1\)
R & D

Still long way to go!
R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)

(*)https://csrc.nist.gov/Projects/Threshold-Cryptography
R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)

- Network layer: authenticated secure p2p communication, Broadcast channel
  - Idea: use a blockchain/consensus layer for the communication
R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)
- Network layer : authenticated secure p2p communication, Broadcast channel
  - Idea: use a blockchain/ consensus layer for the communication
- Improvements: Accountability , Batch signing and verification
Summary

- Threshold ECDSA based Scriptless Scripts / systems
  - Are possible now
  - Are practical to use in real life
  - Hold strong security guarantees and the focus of active research in the cryptography community
Summary

- Threshold ECDSA based Scriptless Scripts / systems
  - Are possible now
  - Are practical to use in real life
  - Hold strong security guarantees and the focus of active research in the cryptography community

Questions?

Special thanks: Oded Leiba, Jonas Nick, Elichai Turkel, Pedro Moreno Sanchez

ZenGo/research

https://t.me/kzen_research

https://github.com/KZen-networks